Natural Language Assistant – A Dialog System for Online Product Recommendation

Joyce Chai, Veronika Horvath, Nicolas Nicolov,
Margo Stys, Nanda Kambhatla, Wlodek Zadrozny and Prem Melville

Abstract

With the emergence of e-commerce systems, successful information access on e-commerce websites becomes essential. Menu-driven navigation and keyword search currently provided by most commercial sites have considerable limitations, as they tend to overwhelm and frustrate users with lengthy, rigid and not very effective interactions. To provide an efficient solution for information access, we have built the Natural Language Assistant (NLA), a web-based natural language dialog system to help users find relevant products on e-commerce sites. The system brings together technologies in natural language processing and human computer interaction to create a faster and more intuitive way of interacting with web sites. By combining statistical parsing techniques with traditional AI rule-based technology, we have created a dialog system that accommodates both customer needs and business requirements. The system is currently embedded in an application for recommending laptops and was deployed as a pilot on IBM's website.

Introduction

For e-commerce web sites, enabling fast access to product information is crucial for generating sales. Users (customers) need to find products matching their interests and businesses need to organize product information to permit quick access. Menu-driven navigation provided by most commercial sites have tremendous limitations, as they tend to overwhelm and frustrate users with lengthy and rigid interactions. User interest in a particular site decreases exponentially with the increase in the number of mouse clicks (Huberman, Pirolli, and Pitkow 1998). Hence shortening the interaction path to provide useful information becomes important.

Many e-commerce sites attempt to solve the problem by providing keyword search capabilities. However, keyword search engines usually require that users know domain specific jargon so that the keywords could possibly match indexing terms used in the product catalog or documents. Keyword search does not allow users to precisely describe their intentions or specify relational operators (e.g. less than, cheapest, etc.) on product attributes. A search for "shirt" can reveal dozens or even hundreds of items, which are useless for somebody who has a specific style and pattern in mind. Moreover, keyword search systems lack an understanding of the semantic meaning of the search words and phrases. For example, keyword search systems usually can not understand that "summer dress" should be looked up in women's clothing under "dress", whereas "dress shirt" most likely in men's under "shirt". Finally, search engines do not accommodate business rules, e.g. a prohibition against displaying cheap earrings with more expensive ones.

A solution to these problems lies, in our opinion, in centering electronic commerce websites on natural language (and multimodal) dialog. Dialog allows the user and the machine to jointly arrive at the intended meaning of the query. Because it is a joint effort, the process is fast. Moreover, it is natural for the site owner to implement business rules as part of the dialog pragmatics. Based on these ideas, we have built the Natural Language Assistant (NLA), a web-based natural language dialog system to help users find relevant products on e-commerce sites.

Even though natural language dialog has been used in many domains, and different architectures are designed for supporting such systems (e.g., Allen et al. 2001), there is no general and practical theory of engineering such applications. Natural Language Assistant (NLA), is therefore another case study, following recent applications that include call-center routing (Chu-Carroll and Carpenter 1998), email routing (Walker, Fromer, and Narayanan 1998), information retrieval and database access (Androutsopoulos and Ritchie 1995), and telephony banking (Zadrozny et al. 1998).

NLA allows customers to make requests in natural language and directs them towards appropriate web pages that sell IBM laptops. The system applies natural language understanding to interpret user inputs, engages in a follow-up dialog with users to provide explanations and to ask for additional information, and finally makes recommendations. The required tight integration of natural language dialog with an e-commerce environment is a novel feature of our system. This involves engineering dialog for the purpose of recommending the merchandise to the user, using user interface studies to guide both the form and the content, and architecting the system to support business rules and business processes for updating the data (e.g. when offerings change). Natural Language Assistant was deployed in a pilot study at an IBM external web site. The data we

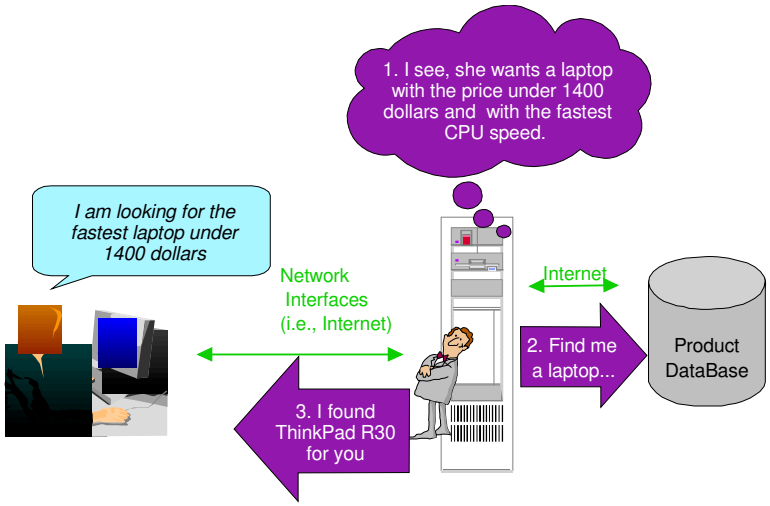

Figure 1: *Interacting with NLA*

collected, together with appropriate business requirements, will form a basis for a decision about its possible wider deployment. The goal of the paper is to describe the behavior and the architecture of the system together with the lessons learned.

In this paper, we start with a typical user session with NLA. Then we give a detailed description on the general architecture and NLA components. Finally, we present the evolution of the system, showing how results from the user studies shaped the development.

Interacting with NLA

When searching e-commerce sites, users often have target products in mind but do not know where to find information, or how to specify a request. Sometimes, users only have vague ideas about the products of interest (Saito and Ohmura 1998). Thus, users need to be able to formulate their requests in their own words as well as revise their request incrementally based on additional information, which can be provided through natural language dialog. Natural Language Assistant was built with that in mind.

Figure 1 shows a high level view of NLA. Users specify their needs to NLA in their own words over the Internet. NLA interprets the input, retrieves products, and gives its response to the user. For example, when the user specifies "the fastest computer under 1400 dollars", based on the understanding of this input, NLA retrieves the laptop (a ThinkPad R30 model) that has the fastest CPU speed among all laptops with price less than 1400 dollars. This example demonstrates the tremendous

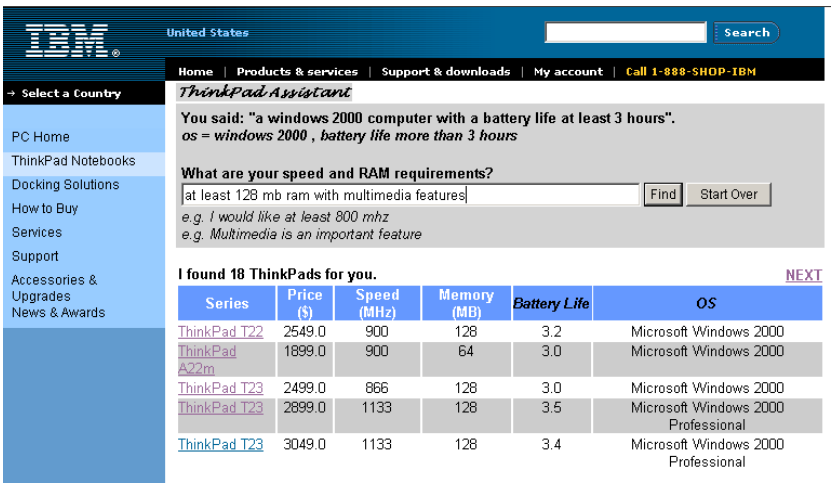

Figure 2: *A screenshot of NLA user interface for requesting more information.*

advantage of natural language search, since the user is able to obtain the desired product in one interaction, as opposed to navigating up and down several layers of menus (menu driven navigation) or browsing among several irrelevant pages (keyword search). A keyword search system may result in no hit or too many hits since the relational operators specified by "fastest" and "under 1400 dollars" are typically neither identified nor computed by search engines. Furthermore, in a menu driven navigation, the user may not be able to choose the product which has fastest CPU speed within the price range if she has no knowledge that it is the CPU speed that determines whether a computer is fast or not.

In addition to the capability of finding products in one click, NLA also engages users in a dialog to elicit their needs and provide guidance. Table 1 shows a typical user session with NLA. Note that the user can describe her requirements using either specifications on product attributes (Memory in utterance User[1], Operating System and Battery life in User[2]) or general concepts (User[4]). Furthermore, the user can express multiple constraints in one input (User[2], User[3]). In this example, NLA explains to the user why a certain requirement cannot be satisfied (NLA[2]), and helps the user narrow down the search space (NLA[3], NLA[4]). NLA does not force the user to only answer the specific questions put to them. For example, NLA asks the user about the price constraints (NLA[2] , NLA[4]), however, the user bypasses this question and provides additional criteria. This type of "mixed initiative" interaction provides flexibility to the user in specifying her needs and to NLA in guiding the user.

NLA [1]	What type of ThinkPad are you looking for?
User [1]:	***"something with at least 256 mb ram".***
NLA [2]:	ThinkPads ship with memory between 64 and 128 MB. I could not find any ThinkPads. Here are some popular modules. What are your price constraints? (Figure 2~~3~~)
User [2]:	**[presses Start Over button]** ***"a windows 2000 computer with a battery life at least 3 hours"***
NLA [3]:	I found 18 ThinkPads for you. What are your speed and RAM requirements?
User [3]:	***"at least 128 mb ram with multimedia features"***
NLA [4]:	I found 2 ThinkPads for you. What are your price constraints?
User [4]:	***"the lightest one"***
NLA [5]:	I recommend the ThinkPad model T23 for you. (Figure 3~~4~~)

Table 1: Example Scenario

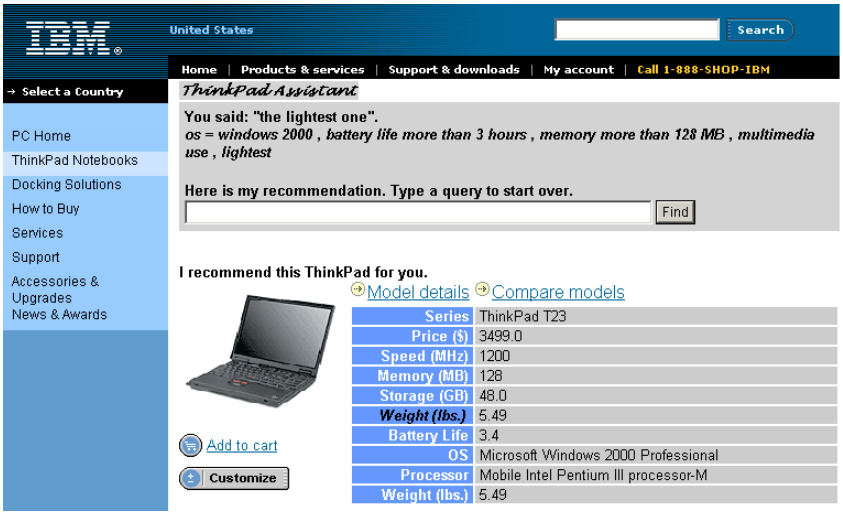

Figure 3: *A screenshot of NLA interface for final recommendation*

System Overview

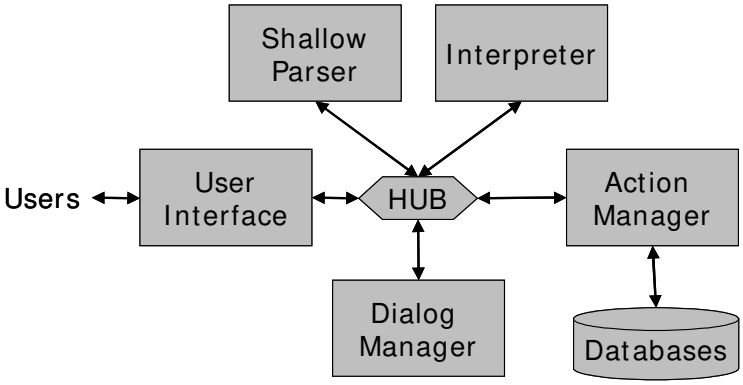

Figure 4: *General architecture*

Our architecture (Figure 4) is designed to support mixed initiative dialog with multiple modalities including typed-in text and speech. In NLA, we use a hub-and-spokes architecture with a central hub responsible for shuttling messages between all other components.

The user interface module is responsible for receiving user inputs and presenting system outputs. Once the input is received by the hub, the shallow parser parses it, and captures important expressions that are used to describe certain features of ThinkPads (e.g., hard disk size, CPU speed) or the usage patterns (e.g., for travel use). Based on these expressions and the session context maintained by the dialog manager, the interpreter constructs a set of constraints on attributes of ThinkPads. Those constraints are then translated to an SQL query by the action manager. The action manager executes the SQL query against a relational product database and retrieves a set of products matching user constraints. Based on the identified constraints and the retrieved products, the dialog manager constructs different responses such as requesting clarification, and soliciting more information to narrow down the recommendation list. Finally, the user interface module renders a screen presenting these responses and the retrieved products to the user. From this interface, the user can start another interaction with NLA. We now describe each of these components in details.

User Interface Module

The user interface module is responsible for receiving user inputs and displaying system outputs. In our architecture, we have a separate user interface for each modality of interaction. The dialog manager determines the content of what is to be presented and the specific user interface renders it using the unique capabilities of the channel and/or modality of interaction.

For the web-based interaction, we designed NLA interface to have a consistent look and feel in every screen. For example, the dialog box is positioned at exactly the same place on every screen. Furthermore, in every screen, NLA re-iterates the user input and provides feedback on what constraints have been understood so far. Such feedback is also reflected in the table of products, where NLA dynamically highlights the attributes in the column that correspond to the identified constraints.

Figure 2 shows a screenshot of the user interface for NLA[3] in Table 1. Note the follow-up question is shown to the user to solicit more information for the purpose of narrowing down the retrieved product list. Furthermore, both *Battery Life* and *OS* are highlighted in the product table to reflect the user specific requests. Figure 3 shows a screenshot of the user interface for the final turn (NLA[5]) of the dialog session in Table 1. Note the merged constraints from previous turns are shown at the top of the page as a feedback.

Parser

NLA uses a shallow, statistical parser to identify expressions in a user input referring to product specifications (e.g., CPU speed, hard disk capacity) or usage categories (e.g., use for multimedia applications). Using a statistical approach allows us to scale to multiple languages and geographies with minimal re-configuration. Thus, in order to create a French language version of NLA, we would only need to collect a corpus of French sentences and annotate them with the existing schemes, instead of recruiting French speaking linguists to create rules for French expressions.

Specifically, the statistical parser learns decision tree models using a corpus of sentences annotated with parse trees. Then the parser applies the learned models on user inputs to create semantic parse trees in a Bottom Up Left Most order (Magerman et al. 1994, Magerman 1995). The parse trees are relatively shallow in our domain given the brevity of user inputs. For example, given the input "*at least 128mb with multimedia features*', the parser will generate the most probable

parse tree as shown in Figure 5, together with the probability for this tree. In this parse tree, the non-terminals (e.g., RAM, MULTIMIDIA) are *labels* that capture the semantic categories of the user input, and the terminals (e.g., at least 128 mb) are the actual user expressions. This resulting parse tree is used by the interpreter to extract constraints. The parser is robust, fast and is not memory intensive. It is packaged as a separate module and receives parse requests via socket communication.

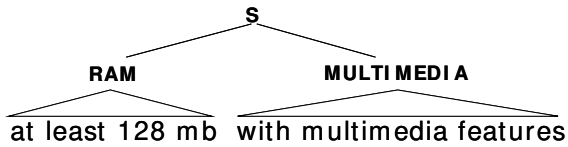

Figure 5: *Parse tree for the input 'at least 128mb with multimedia features'*

During the development, we collected 10069 user queries about ThinkPads to build the statistical parser model. We used 6804 queries as a training set for the parser, 2253 as a validation set and 1012 as a test set. The queries were collected from user interactions with a previous version of the system using a finite-state parser (Chai et al. 2001b). We use 32 labels to categorize different attributes of ThinkPads (e.g., PRICE, WEIGHT) and 6 labels to categorize usage patterns (e.g., TRAVEL-USE, MULTIMEDIA). We have gone through several cycles of revising the initial annotation, fine tuning the parser features and re-training the model.

Currently, the parser parses the test set with an average precision of 92% and an average recall of 94% for identifying the labels. In general, the parser works best for labels associated with well-defined crisp semantic meanings (e.g. PRICE, CPUSPEED, etc.). If we only consider the labels corresponding to product attributes, we obtain an average precision of 94% and an average recall of 98% for the test set. For labels corresponding to usage categories that tend to be more subjective in nature (e.g. CUTTING_EDGE), we obtain an average precision of 84% and an average recall of 80% for the test set. Thus, our parser is very good at identifying common product attributes, but works less well for identifying all possible interpretations of subjective usage categories. We believe that training with more data and modifying our label selection and annotation schemes will help with the latter.

Interpreter

The interpreter extracts a semantic representation (e.g. propositional formula of constraints over product attributes) from the parse tree returned by the parser. Specifically, from all the labeled chunks of text identified by the parser, the interpreter extracts constraints that specify relations and values for product attributes (e.g., PRICE < 2500, WEIGHT = min, CPUTYPE = 'Pentium', etc.). Furthermore, to keep the context of a dialog, the interpreter also integrates the constraints identified from the current input with the constraints captured previously in the session.

The interpreter first extracts constraints from the labeled chunks of text describing product specification. Depending on the (abstract data) types of the attributes, we distinguish between numerical constraints (PRICE < 2500), string constraints (CPUTYPE = 'Pentium') and constraints over pairs (RESOLUTION = 1600x1200). The values of numerical constraints are normalized to canonical units of measure (dollars for PRICE, MHz for CPUSPEED, etc.) using finite-state transducers. For example, given a user expression "*faster than 1.3 GHz*" which is categorized as CPUSPEED, the interpreter converts "*1.3 GHz*" into "*1300 MHz*".

We have explored two approaches for the treatment of string valued attributes. The first approach uses finite state patterns to produce a canonical string value that is directly matched (e.g. using substring matching in SQL) against string values in the product database. This approach requires us to pre-specify a canonical list of values for each string valued product attribute. Thus, this approach requires ongoing system maintenance costs as new products are released with either new values for existing attributes or with new attributes.

To avoid such dependencies on external (to our dialog system) resources, we have implemented an approach using Information Retrieval (IR) techniques. NLA matches the expression in a particular attribute category with values of that attribute in the product database, and chooses the most similar one(s) using similarity measurement. For example, for the query "*I want a machine with win xp*", the interpreter identifies the constraint (OS = "win xp"). If the values for the OS attribute in the database are: "Microsoft Windows XP Professional", "Windows NT", "Windows 2000 Professional", "Linux", the best match is "Microsoft Windows XP Professional".

For expressions in the usage category, the interpreter applies business rules to create constraints. Business rules provide a mechanism for bridging the gap between user vocabulary and business requirements. In other words, the parser provides the usage categories identified from the user input and the business rules specify how those categories relate to products (by providing constraints on product specifications). For example the MULTIMEDIA usage category is defined by the following business rule

```
MULTIMEDIA ::=      (DEVICE = dvd) &
                    (CPUSPEED: high) &
                    (HDSIZE: high) &
                    (DISPLAY ≥ 14.1) .
```

This rule indicates that a machine that can be used for multimedia purposes should have a DVD, high CPU speed, a large disk drive and a display with at least 14.1 inches. This example also shows the use of *qualitative constraints* (e.g. HDSIZE: high) that are "low" or "high" constraints on numerical product attributes. The qualitative constraints are further mapped to specific constraints like (HDSIZE > 20GB) based on automatic partitioning of the current range of values. For example, among all available values for the hard disk size, the top one fifth are considered as "high". Using qualitative constraints in business rules can reduce the maintenance effort. For example, the size of a hard disk that is considered to be large changes with time as larger disk spaces are available in new products. By using qualitative constraints, when such changes occur, the business rule can remain the same although the constraint (HDSIZE: high) will be interpreted differently through a dynamic mapping of "high" to a new range of values.

Constraints are grouped together with the usual propositional connectives to form formulae. Most often the connectives are conjunctions and elements in the formulae are either constraints or negated constraints. These formulae are passed on to the action manager to retrieve products.

Furthermore, to keep the dialog context, the interpreter merges the constraints identified from the current input with those captured previously in the session. It is possible to have multiple constraints on the same attribute. They could either have been specified directly by the user or occur due to the expansion of business rules. We use the following heuristics in the integration process. First, constraints directly specified by the user override other constraints. For example, if the user wants a MULTIMEDIA machine (which implies CPUSPEED: high which in turn is expanded as CPUSPEED > 1000) and the user explicitly requested (CPUSPEED > 900), then the resulting constraint from the CPU speed would be (CPUSPEED > 900). Second, the most recent constraint overrides previous constraints with the same attribute and relation. For example, if the user had previously specified (PRICE < 2000) and is now expressing a new constraint (PRICE < 1800), the most recently expressed constraint (PRICE < 1800) will prevail. Third, constraints on the same attribute with different compatible relations are preserved. For example, combining (PRICE < 2000) from a previous turn with (PRICE > 1500) will result in the range 1500-2000, i.e. both constraints will be kept.

Action Manager

The action manager is responsible for the back-end operations. In particular, the action manager translates constraints generated by the interpreter to a SQL statement. Based on this SQL, the dialog manager retrieves products from a relational database that contains product information. For example, if in consecutive turns the user specifies "*256 MB*" and "*fastest under 2000 dollars*", then the generated SQL is:

```
SELECT * FROM table WHERE
        ram = 256 AND
        price < 2000 AND
        cpuspeed = (SELECT max(speed) FROM table WHERE
                                price < 2000 AND
                                ram = 256).
```

To process the min/max constraints properly, the dialog manager considers the set of products satisfying the constraints from previous turns, and among these the dialog manager identifies products satisfying min/max constraints. Furthermore, when multiple min/max constraints are given, the dialog manager first applies all constraints other than the min/max constraints, and then applies the min/max constraints in reverse order of occurrence. This is necessary to ensure the retrieved products correspond to the most common linguistic interpretation of min/max constraints. For example, for the query "*fastest, lightest computer with 20 GBs*", the action manager first searches for "20 GB", then the "lightest", and finally the "fastest." i.e., of all machines with hard disk of 20 GB consider those with minimum weight and among those select the fastest. Any other order of processing constraints would correspond to a different interpretation of the user constraints and might result in unintended products being retrieved.

Dialog Manager

The dialog manager generates the system response based on the current user input, the prior dialog in the session, and the retrieved products. In particular, the dialog manager employs a mixed initiative strategy to interact with a user. At the beginning of each session, the dialog manager prompts users with a general question (e.g., NLA[1] in Table 1) to solicit specific requests. Moreover, at any point in the session, the dialog manager allows users to bypass questions put to them and describe

their needs directly. While giving the initiative to users, the dialog manager also takes the initiative by asking users very specific questions about different product attributes, thus directing the users to achieve their dialog goals.

NLA differentiates between two types of users. If the user initial query expresses requirements on any product attributes directly, the dialog manager classifies the user as a "technology savvy" user and, for the remainder of the session, the dialog manager only prompts her with questions concerning specific product attributes. Alternatively, if the user initial query expresses only usage patterns, for the remainder of the session, the dialog manager only prompts the user for information on general usages.

The dialog manager employs different strategies to deal with different situations. When no constraints are identified from a user input, the dialog manager presents a clarification screen suggesting possible queries and explaining the capabilities of NLA. When a user specifies an invalid constraint (e.g., User[1] in Table 1), the dialog manager presents the valid range of constraints for the attributes in question and prompts the user to reformulate her query. If the action manager retrieves more than one product based on constraints identified so far, the dialog manager prompts the user for constraints on product attributes or usage categories (depending upon the first query as explained above) that best discriminate among the retrieved products. If the action manager retrieves exactly one product based on constraints identified so far, the dialog manager recommends the product to the user, explains the reason for the recommendation and invites the user to start another search.

In a special situation where constraints identified result in no products being retrieved, the dialog manager employs the following strategy. The dialog manager (via the action manager) separately retrieves a pool of products for each constraint. If any of these product pools is empty, the dialog manager prompts the user with the range of values for the corresponding product attribute. Then the dialog manager merges (union of sets) all the non-empty product pools, and sorts them using a distance measure that measures the closeness of a product to the set of constraints. This merged product pool is presented to the user along with an alert about the conflicting nature of the identified constraints. For example, if the user inputs "*under 1000 dollars and at least 900 MHz*", the action manager will not retrieve any products since no laptop satisfies both of these constraints. In this case, the dialog manager instructs the action manager to separately retrieve the pool of laptops that are priced under $1000 and the pool of laptops that have at least 900 MHz CPU speed. These two product pools are merged and sorted with respect to closeness to both of the constraints. The sorted list is presented to the user. If all the product pools are empty, the user is prompted to reformulate her query.

The dialog manager maintains a dialog history that records the user input, the set of identified constraints, the list of products retrieved, and the system output at each turn of the dialog. Unlike other systems that have complex structures capturing user intentions and the focus of attention (e.g. LINLIN (Jonsson, 1997)), our dialog history is very simple. However, we found that this simple representation is sufficient for our application.

Data Management & Maintenance

We have developed various tools and processes to maintain the NLA system to ensure that updates to products and other resources are seamlessly reflected in user interactions. In a business setting, various databases are often pre-designed for other purposes and hence present problems for our system: e.g. the database might not have the right data types, multiple attributes might be represented in a single database column, etc. To address these issues, we maintain a local database that is populated directly from the original databases. We implemented an automated process to access the product databases to convert data types and extract product specification on a daily basis. Our script robustly copes with missing data values, multiple attributes merged into one attribute, etc. In addition, we have also explored the direct extraction from product web pages using a web-based tool that applies finite-state patterns to extract product specifications.

Furthermore, when new products or features are introduced, the business rules need to be updated accordingly. When more and more user inputs are collected, the statistical parser needs to be re-trained. Thus, we have implemented a tool for maintaining business rules and the statistical parser. The tool automatically extracts n-grams from logs of user queries and allows manual updates of business rules through an editing interface. A parts-of-speech tagger and a noun phrase grammar are used to select new input patterns. The new patterns are labeled through the interface and added to the training examples for the statistical parser. Figure 6 shows the interface where automatically identified bi-grams can be added to existing categories.

In addition to coping with the evolving data from the technology aspect, it is worth pointing out that human interaction is important in the data management process. In a business organization, different groups are responsible for different product parameters. Thus, interacting with different groups to understand the structure and the type of the data is important. Such interactions usually take a lot of effort and add the complexity of data management.

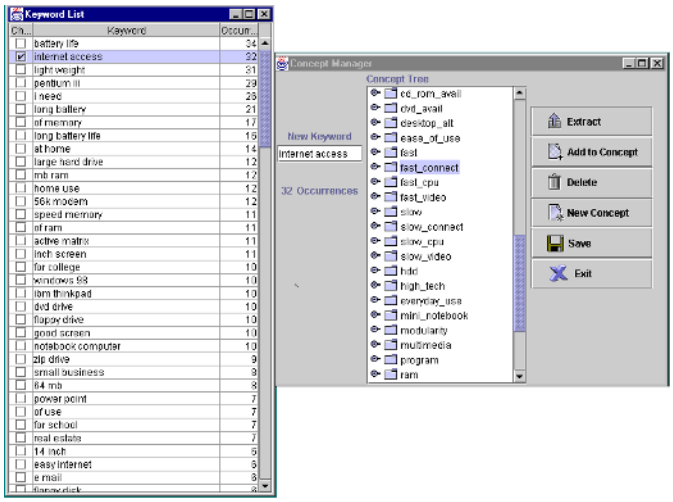

Figure 6: *Editing interface for concept management*

Implementation

NLA is implemented as a client-server system using Java servlets, WebSphere and DB2. We use HTTP to communicate between the client and the server. The system development was done under Visual Age for Java. The user interfaces were implemented using DHTML (HTML4.0, Cascading Style Sheets and JavaScript), JSPs and Java servlets. We have developed versions of NLA for different geographies as well as for different product lines.

For efficiency reasons the statistical parser is implemented in C. The NLA system connects to the parser via sockets. For training the parser, we pre-annotated data using a finite-state parser used in a previous version of the system (Chai et al., 2001b). These raw annotations were reviewed manually using a GUI annotation tool. We have also used examples artificially generated by using a Prolog Definite Clause Grammar (DCG) to cover more variations in user inputs.

System Evolution via Iterative Design

The present version of NLA has evolved through various cycles of iterative design. Specifically, we went through four stages of system development: concept proof, prototyping, pilot deployment, and post-pilot enhancement. During these stages, we incrementally designed and implemented different versions of NLA, and conducted user studies to evaluate the technology and improve the system. In this section, we share our experience and the results from the user studies carried out at separate stages of development.

Proof of Concept

For the proof of concept, we developed HappyAssistant, a simple rule-based system that provided limited language processing and dialog capabilities (Chai et al. 2001a). At this initial stage of development, it was important to learn users' reactions to this novel navigation approach as opposed to traditional approaches (e.g., menu driven navigation). Thus, we compared HappyAssistant with a menu driven system. We were particularly interested in finding answers to the following questions: Can natural language based navigation be more efficient (number of clicks, time spent searching, etc.) and easier to use than menu driven navigation? By how much? What are users' responses toward natural language based navigation as opposed to menu driven navigation? How do users with different levels of online experience react to the natural language dialog based navigation?

Seventeen subjects were recruited for the comparative study: four had advanced computer skills, eight were deemed to be at the intermediate level of proficiency and five had limited experience with the internet. Each participant was asked to use both the HappyAssistant and the menu driven system, following a set of pre-defined scenarios. The scenarios were designed to let the users experience the navigation of each web site in order to form an opinion of the tool's concept. They were then

asked to rank the tasks on a 1 to 10 scale (where 10 is easy) with regards to the ease of navigation and the series of events leading up to the result.

The results of this study showed that, to accomplish those tasks, HappyAssistant required less time and user movements (mouse clicks) than the menu driven system. Specifically, HappyAssistant reduced the average number of clicks by 63% and the average interaction time by 33% (compared with a menu-driven system). Furthermore, the less experienced users preferred the natural language enabled navigation much more than the experienced users. Table 2 shows the rating from different user groups in terms of the ease of use of the two systems. Overall, respondents preferred the natural language dialog based navigation (HappyAssistant) to the menu driven navigation two to one (2:1).

System	Novice	Intermediate	Experienced
HappyAssistant	9.4	8.5	8.3
Menu Driven System	6.3	8.1	8.9

Table 2: Ratings of ease-of-use of the two systems

In this study, we also found that users are accustomed to typing in keywords or simple phrases (e.g., "*moderately priced laptop*", "*computer with internet access + games*", "*a high-speed computer*"). Despite the moderator's assurance that the user could type "anything they wanted," complete sentences were seldom observed. The average length of a user query was 5.31 words long (with a standard deviation of 2.62). Analysis of the user queries reveals the brevity and relative linguistic simplicity of the input; hence, shallow parsing techniques seem adequate to extract the necessary meaning from the user input. Therefore, in such context, sophisticated dialog management is more important than the ability to handle complex natural language sentences. We also learned that in order to improve the functionality of an e-business site, the natural language dialog navigation and the menu-driven navigation should be combined to meet users' needs. While the menu-driven approach can provide choices for the user to browse around or learn some additional information, the natural language dialog provides the efficiency, flexibility and the natural touch to the users' online experience. Moreover, in designing natural language dialog based navigation, one of the key issues is to show users that the system understands their requests before giving any recommendation or relevant information.

Prototyping

Based on what we learned from the first user study, we developed the Natural Language Sales Assistant (NLSA). NLSA applied a shallow noun phrase parser to process user inputs. To enhance the dialog capability, NLSA used a mixed initiative, state-based dialog manager. Since the first user study highlighted a definite need for system acknowledgement and feedback, NLSA incorporated an explanation model that explained to the user what was understood and why a particular product was recommended. Furthermore, NLSA addressed the issue of real time data management and provided tools for managing data and knowledge used in online interaction. A detailed description of NLSA can be found in (Chai et al. 2001b).

Prior to the development of NLSA, we conducted a user survey to help understand specific user needs and collect user vocabulary. Users were given three sets of questions. The first set, in turn, contained three questions: "What kind of notebook computer are you looking for?", "What features are important to you?", and "What do you plan to use this notebook computer for?". By applying statistical n-gram models and a shallow noun phrase grammar to the user responses, we extracted keywords and phrases expressing users' needs and interests. In the second set of questions, users were asked to rank 10 randomly selected terms from 90 notebook related terms in order of familiarity to them. The third set of questions asked for demographic information about users such as their gender, years of experience with notebook computers, native language, etc. We derived correlations between vocabulary/terms and user demographic information. This study allowed us to group technical terms into different complexity groups and better formulate system responses to different user groups. Over a 30-day period, we received 705 survey responses. After about 400 responses, the number of extracted keywords and phrases started to converge. From this survey, we extracted 195 keywords and phrases. These keywords and phrases helped us jump-start the development of NLSA. We believe this kind of market survey could be one approach to help customizing our technology to a different domain.

We also conducted the second user study to test the usability of NLSA. In this study, we focused on evaluating the dialog flow and the ease of use. Thirty four subjects with "beginner" or "intermediate" computer skills were interviewed for the study. Again, they were asked to find laptops for a variety of scenarios using three different systems: the NLSA, a directed dialog system (through pre-designed questions and answers), and a menu driven navigation system. Participants were asked to rate each system for each task on a 1 to 10 scale (10 – easiest) with respect to the ease of navigation, clarity of terminology and their confidence in the system responses. The focus of the second study was to compare systems of similar functionality and to draw conclusions about the functionality of NLSA.

The results showed that the users clearly preferred dialog-based searches to non-dialog based searches (79% to 21% users). Furthermore, they liked the narrowing down of the product list based on identified constraints as the interaction

proceeded. Our analysis reveals statistical differences in terminology ratings among the three systems for the category of beginner users only. There were no statistical differences found in the other ratings of *navigation* and *confidence* across the three sites for different categories of users. The results suggest that asking questions relative to the right level of end user experience is crucial. Asking users questions about their lifestyle and how they were going to use a computer accounted for a slight preference of the directed dialog system over the NLSA that uses questions presented on the basis of understanding features and functions of computer terms.

Again, as in the first user study, we learned that, it is important to show users that the system understands them. Users remarked in our study that they appreciated the recommended results because the system offered some explanation. This feature allows the user to "trust the system." Good navigation aids can be provided by summarizing the user's requests by paraphrasing them using context history, or by engaging in conversations with the user. Our studies found that robust natural dialog had a very big influence on the user satisfaction – almost all of the respondents appreciated the additional questions prompted by their input and the summary of their previous queries.

The studies pointed towards improvements in the area of system responsiveness including tuning up of the follow up questions, prompts and explanations to the user's input. To a large extent, the success of a dialog system has been shown to depend on the kind of questions asked and the type of feedback provided. User's confidence in the system decreases if the system responses are not consistent with the user's input. The system feedback and the follow up questions should manage a delicate balance of exposing system limitations to the user but at the same time making sure the user understands the degree of flexibility and advantages of using a dialog system.

Pilot Deployment

We made further improvements to NLSA based on the results of the user studies and deployed NLSA on an external IBM website for a few months as a pilot. During the pilot, we collected valuable feedback from real users that greatly helped subsequent system improvements.

For the pilot data, the average user query was 6.1 words long. This is significantly higher than the roughly 2.2 words per query for search engines. We also found that users were open to typing in long natural language expressions to find ThinkPads. The maximum query length was over 150 words long.

Perhaps the most surprising finding of the pilot study was that a large proportion of user queries were technical in nature, expressing very specific needs about different product attributes. Users freely (and without any coaching or guidance) expressed relational operators (e.g. less than, at least, etc.) and conjunctions of multiple constraints. This suggests that NLSA is very useful for power users, enabling them to quickly get to the products of interest to them. Moreover, if a user has very specific technical requirements (e.g. "an xga computer with at least 20 gb, 128 mb ram and 15" tft display"), NLSA is often the best mechanism of finding the relevant products quickly (compared to keyword search or menu driven navigation).

Post-pilot Enhancement

Having carried out the two user studies and learned the lessons from the pilot deployment, we are now developing the third version of the system: the Natural Language Assistant (NLA).The system described in this paper is the result of this effort.

In particular, as described earlier, we re-designed the questions that NLA asks users to be simpler, and to focus on usage patterns rather than technical features. Subsequently, we added functionality that classified users into general versus technical categories. If the technical category of users was detected, a technical pool of questions would apply. We also integrated a statistical parser with NLA to more robustly handle varied user input. The statistical parser should enable NLA to scale to multiple languages and multiple domains in a more robust and reliable fashion. In addition, we have designed a more uniform, more compact and consistent UI.

While developing NLA, we iterated through various design phases as described above. This helped us learn more about user requirements and system limitations, and enabled us to incrementally improve the system in a systematic fashion. Our studies confirmed the hypothesis that a natural language dialog interface is a significant improvement over existing product retrieval mechanisms. In future studies, we would like to focus more on defining quantitative and objective measures of system's success.

Conclusion

This paper describes a natural language dialog system that helps users find products satisfying their needs on e-commerce sites. The system leverages technologies in natural language processing and human computer interaction to create a faster and more intuitive way of interacting with websites. By combining techniques in robust statistical parsing with traditional AI rule-based technology the system is able to accommodate both customer needs and business requirements.

Our studies show that dialog-based navigation is preferred over menu-driven navigation (79% to 21% users) and confirm the efficiency of using natural language dialog in terms of the number of clicks and the amount of time required to obtain the relevant information. Compared to a menu-driven system, the average number of clicks used in the natural language system was reduced by 63.2% and the average time was reduced by 33.3%. In a pilot study, we found that, when presented with the right interface, users do type long, technical queries (average of 6.10 words per turn), for example, expressing relational constraints on multiple product attributes or usage categories. Moreover, our pilot study revealed that technical users were able to use NLA successfully to quickly find products of interest to them. Thus, a shallow natural language layer on top of a relational database offers a powerful alternative to traditional keyword search or menu driven systems for e-commerce sites. Additionally, the use of a thin dialog layer makes the system accessible to all types of users and greatly enhances the user experience.

Natural language dialog interfaces offer a more "natural" mode of interaction than traditional user interaction mechanisms like command-driven interface, form-filling interface, question-answer sequences, menus, etc. However, natural language dialog also faces serious challenges. For novice users, a conversational system may be overwhelming and it may be quicker to use a menu-driven system. For experienced users, the amount of typing may be a drawback and browsing may be the best and quickest way to navigate. Ultimately, in order to satisfy different user needs, the natural language dialog navigation and the menu-driven navigation should be combined. While the menu-driven approach provides choices for the user to browse around or learn some additional information, the natural language dialog provides the efficiency, flexibility and natural touch to the user's online experience.

Furthermore, conversational interfaces offer the ultimate kind of personalization. Personalization can be defined as the process of presenting each user of an automated system with an interface uniquely tailored to his/her preference of content and style of interaction. Thus, mixed initiative conversational interfaces are highly personalized since they allow users to interact with systems using the words they want, to fetch the content they want in the style they want. Users can converse with such systems by phrasing their initial queries at a right level of comfort to them (e.g. "*I am looking for a gift for my wife*" or "*I am looking for a fast computer with DVD under 1500 dollars*").

In the next few years, natural language dialog should become the preferred mode of interaction with institutional knowledge as well. Hence our effort in building the Natural Language Assistant can be viewed as a prelude to such more advanced systems. While the existence of a product hierarchy and a limited number of product parameters makes e-commerce a natural domain for natural language dialog systems, our approach can be naturally expanded to industrial and enterprise domains outside e-commerce for which well-defined ontologies are available. Similarly, we expect that the lessons learned about engineering the interactions to be applicable there. Our work in knowledge update processes done as part of this project might eventually be expanded to address knowledge infrastructure in other domains (we see it as an interesting open problem). These three problems – ontology, HCI and knowledge architectures -- cover the basic pragmatics of engineering interactive knowledge systems.

Acknowledgements

We would like to thank John Karat and Catherine Wolf for input on user testing and user interface design, Jimmy Lin and Yiming Ye for contributions to the first version of the system and the first user study, Sunil Govindappa for contributions to the second version of the system, Xiaqiang Luo, Niyu Ge, Salim Roukos for help in the transitioning to the new statistical parser, Jose Menes for tools for user session analysis, and Tomek Czajka for maintenance tools as well as colleagues from the Conversational Systems group at IBM for illuminating discussions.

References

Allen, J.; Ferguson, G.; and Stent A.. 2001. An Architecture for More Realistic Conversational Systems. In *Proceedings of 2001 International Conference on Intelligent User Interfaces (IUI)*, pp. 1-8.

Androutsopoulos, I., and Ritchie, G. D. 1995. Natural Language Interfaces to Databases – an Introduction. *Natural Language Engineering* 1(1):29-81, Cambridge University Press.

Chai, J.; Lin, J.; Zadrozny, W.; Ye, Y.; Budzikowska, M.; Horvath, V.; Kambhatla, N.; and Wolf, C. 2001a. The Role of a Natural Language Conversational Interface in Online Sales: *A Case Study.* In *International Journal of Speech Technology Volume 4, Numbers 3/4*, pp. 285-295, Kluwer Academic Publishers.

Chai, J.; Horvath, V.; Nicolov, N.; Stys-Budzikowska, M.; Kambhatla, N.; and Zadrozny, W. 2001b. Natural Language Sales Assitant – A Web-based Dialog System for Online Sales. In *Proceedings of the Thirteenth Innovative Applications of Artificial Intelligence Conference,* pp. 19-26. The AAAI Press.

Chu-Carroll, J., and Carpenter, B. 1998. Dialog Management in Vector-based Call Routing. In *Proceedings of the 36th Annul Meeting of the Association for Computational Linguistics*, pp. 256-262

Huberman, B. A.; Pirolli, P. L. T.; Pitkow, J. E.; and Lukose, R. M. 1998. Strong Regularities in World Wide Web Surfing. *Science*, Vol. 280, pp. 95-97.

Jonsson, A. 1997. A model for habitable and efficient dialog management for natural language interaction. *Natural Language Engineering*, 3(2/3):103-122.

Magerman D., Jelinek F., Lafferty J., Mercer R., Ratnaparkhi A., and Roukos S., 1994. Decision tree parsing using a hidden derivation model, in Proceedings of ARPA Human Language Technology Workshop, pp. 272-277.

Magerman D., 1995. Statistical decision-tree models for parsing, in Proceedings of the ACL conference, June 1995, pp. 276-283.

Saito, M., and Ohmura, K. 1998. A Cognitive Model for Searching for Ill-defined Targets on the Web – The Relationship between Search Strategies and User Satisfaction. In *Proceedings of 21st International Conference on Research and Development in Information Retrieval*, Australia, pp. 155-163.

Walker, M.; Fromer, J.; and Narayanan, S. 1998. Learning Optimal Dialogue Strategies: A Case Study of a Spoken Dialogue Agent for Email. In *Proceedings of 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, Montreal, Canada, pp. 1345-1351.

Zadrozny, W.; Wolf, C.; Kambhatla, N.; and Ye, Y. 1998. Conversation Machines for Transaction Processing. In Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI) and Tenth Conference on Innovative Applications of Artificial Intelligence Conference (IAAI), Madison, Wisconsin, USA, pp. 1160-1166.

Autobiographies

Joyce Chai

Joyce Chai is a research staff member at IBM T. J. Watson Research Center. She received a B.A. in mathematics from Trinity College in 1993 and a Ph.D. in computer science from Duke University in 1998. Her thesis focused on learning and generalization for building trainable information extraction systems. Since joining IBM Research in 1998, she has worked on web-based dialog systems. Her research interests include natural language processing and multimodal user interfaces. Her current research centers on multimodal integration and discourse modelling. Her email address is jchai@us.ibm.com.

Veronika Horvath

Veronika Horvath is a software engineer at IBM T.J. Watson research center. She received a M.A. in linguistics from the Hungarian Academy of Sciences, a Ph.D. in applied linguistics from Ball State University, and a M.S. in computer science from Southern Illinois University. Her professional interests include Natural Language Processing, Dialog Systems, and Information Retrieval. Her e-mail address is veronica@us.ibm.com

Nicolas Nicolov

Nicolas Nicolov is a research staff member at IBM's T.J.Watson Research Center. His current research focuses on dialog systems, natural language generation and robust, efficient and scalable techniques for multimodal and multilingual language processing. He received an M.Sci. from the University of Sofia in 1992. He then joined the Department of Artificial Intelligence at the University of Edinburgh where he did his Ph.D. in the area of Natural Language Generation. He was the developer of the PROTECTOR NLG system. Between 1996-99 he was a postdoctoral fellow at the School of Cognitive Science, University of Sussex working on grammar engineering and wide-coverage parsing for English. He has worked for Apple and was a visiting scholar at LIMSI-CNRS (1992) and IMS, University of Stuttgart (1996). His email address is nicolas@us.ibm.com.

Margo Stys

Dr. Margo E. Stys is currently a Research Staff Member of the Conversational Dialog Systems Group at IBM T. J. Watson Research Center, which she joined shortly after receiving her PhD from the Computer Lab at the University of Cambridge (1998). Her research spans a diverse range of NLP topics including web-based dialog agents, discourse structure models,

machine translation and computer aided language learning. Her most recent research endeavors involve designing automated dialog evaluation paradigms.

Nanda Kambhatla

Nanda Kambhatla was born in Hyderabad, India in 1969. He received a B.Tech degree with first class honors in 1990 in Computer Science and Engineering from the Institute of Technology, Benaras Hindu University, India, and a Ph.D degree in Computer Science and Engineering from the Oregon Graduate Institute of Science & Technology, Oregon, USA, in 1996. Since 1996, Dr. Kambhatla has worked as a postdoctoral fellow under Prof. Simon Haykin at McMaster University, Canada and as a senior research scientist at WiseWire Corporation, Pittsburgh. He joined IBM as a research staff member at IBM's T.J.Watson Research Center in New York and is currently leading a team of researchers working on conversational dialog systems for Web and telephony applications. His research interests include all aspects of machine learning algorithms and their application to textual, speech and image data processing.

Wlodek Zadrozny

Wlodek Zadrozny is a senior researcher at IBM T.J. Watson Research Center. Currently, he is coordinating the work of Search and Text Analysis Institute. Previously, as a member of IBM Research technical strategy team, he investigated new technologies for their possible business impact. From 1995 until 2000, he led a team building natural language applications for the web and telephony. His interests focus on business impact of intelligent technologies, natural language semantics and interactivity.

Prem Melville

Prem Melville is a Ph.D. candidate in Computer Science at the University of Texas at Austin. He received his B.S. in Computer Science and Mathematics from Brandeis University. His research interests include learning for recommender systems, text categorization, ensemble methods and active learning. He was a summer intern (2000) in the Conversational Machines group at IBM T. J. Watson Research Center. His e-mail address is melville@cs.utexas.edu.