RC23787 (W0511-061) November 8, 2005
Computer Science

IBM Research Report

Enabling Context-Sensitive Information Seeking

Michelle X. Zhou, Keith Houck, Shimei Pan, James Shaw,
Vikram Aggarwal, Zhen Wen
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Enabling Context-Sensitive Information Seeking

Michelle X. Zhou Keith Houck Shimei Pan James Shaw Vikram Aggarwal Zhen Wen
IBM T. J. Watson Research Center
19 Skyline Dr.
Hawthorne, NY 10532

{mzhou, houck, shimei, shawjc, vikram, zhenwen}@us.ibm.com

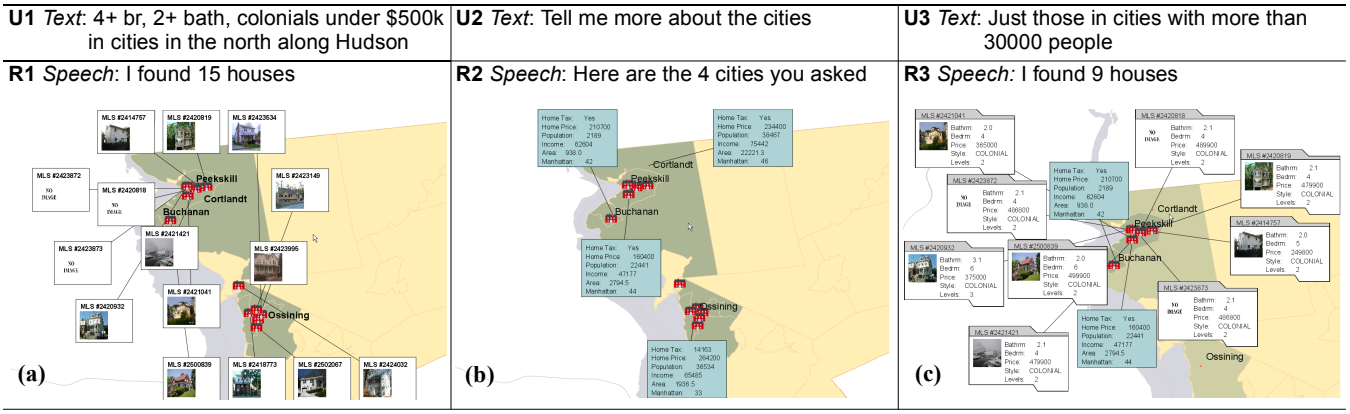

Figure 1. A recorded user-RIA interaction fragment: U1–U3 are user inputs, R1–R3 are corresponding RIA-generated responses.

Abstract

Information seeking is an important but often difficult task, especially when it involves large and complex data sets. We hypothesize that a context-sensitive interaction paradigm would greatly assist users in their information seeking. In particular, such a paradigm would allow users to both express their requests *and* receive requested information in context. Driven by this hypothesis, we have followed a rigorous process to design, develop, and evaluate a full-fledged, context-sensitive information system. We started with a Wizard-of-OZ (WOZ) study to verify the effectiveness of our envisioned system. We then built a fully automated system based on the findings from our WOZ study. We targeted the development and integration of two sets of technologies: context-sensitive multimodal input interpretation and multimedia output generation. Finally, we formally evaluated the usability of our system in real world conditions. The results show that our system greatly improves the users' ability to perform information-seeking tasks, thus confirming our initial hypothesis. Moreover, these results indicate that our approaches are capable of supporting context-sensitive information seeking in practical applications.

Keywords

Context-sensitive information seeking, intelligent input interpretation, automated output generation.

1. INTRODUCTION

People seek information daily, for example, exploring various options of vacation packages or examining multiple aspects of residential properties. However, information seeking can often be very difficult and time consuming for two main reasons. First, users often cannot directly specify their information desires using conventional WIMP-based interfaces. This becomes more evident when users do not know exactly what they are looking for at the beginning of the search (e.g., finding a dream home). In such cases, users must examine multiple data aspects and explore different but related data entities to determine their targets. To express a query like U1 in Figure 1, users may only be able to approximate the query by filling multiple forms plus manually stitching together information gathered at different steps (e.g., finding all desired cities first and then using the results to find houses). Moreover, most existing systems are context insensitive. Continuing the above example, most systems cannot directly respond to follow-on queries like U2/U3. Without being able to easily revise their queries in context, users must start over again (e.g., traversing multi-level menus to switch navigation targets or reformulating queries with new search foci or criteria). Second, users cannot easily distill information from often scattered, one-size-fits-all presentations of retrieved data. As a result, users have to manually integrate relevant information appearing at multiple places (e.g., one display on houses and another on schools). A context-insensitive output may also prevent users from easily identifying important information or comprehending the information as a whole.

To address the issues mentioned above, we envision *context-sensitive information systems* that allow users to express their requests and receive the requested information in context. Driven by this vision, we have built a full-fledged system, called Responsive Information Architect (RIA). RIA aids users in their information seeking from two aspects. First, RIA lets users specify and revise their requests in context using multiple modalities, including natural language and GUIs. Second, RIA dynamically creates a multimedia response that is tailored to the user's interaction context, including user interests and interaction history.

Figure 1 shows a fragment of user-RIA interaction in a real-estate domain. Starting with U1, the user asks for a set of houses. Accordingly, RIA dynamically synthesizes a multimedia response based on a number of factors at run time. For example, based on the retrieved data and the available space, RIA displays house images

and MLS numbers. Tailoring to the query expression, RIA also conveys the relevant city information to provide the spatial context for the houses. As a follow up, the user wants to examine the cities further (U2). Using the conversation context, RIA is able to supply the requested city information (R2). After knowing more about the cities, the user then shifts back to the houses and asks to filter them using an additional city population constraint. Again, RIA understands the input in context and returns the requested houses (R3).

Unlike our previously published work, which focuses on a specific RIA component [22, 31, 30, 21], here we report on the overall design, development, and evaluation of an end-to-end RIA system. More importantly, we present concrete evidence to support our two assertions: 1) RIA-like systems greatly improve user information seeking experience; and 2) our approaches are practical for building real-world applications. We organize our evidence around three fundamental questions:

1. Whether/how does our envisioned system help improve user information-seeking experience?
2. Whether/how can we implement the desired features of our envisioned system?
3. How well does the implemented system help users in their information seeking?

To address the first question, we conducted a Wizard-of-OZ (WOZ) study to validate the effectiveness of our envisioned system along with its functional requirements.

To address the second question, we evaluated which desired system features identified by our WOZ study are feasible to support. We then built a fully working RIA to support these features. In this process, we applied the same set of technologies to two different applications to validate the reusability and extensibility of our approaches.

To address the third question, we performed a formal user evaluation to determine whether the implemented RIA effectively aids users in their information seeking.

To the best of our knowledge, we believe that our work is the first to comprehensively address the entire process of designing, developing, and evaluating a practical, context-sensitive information system. As a result, it offers three unique contributions. First, we share our findings from the WOZ study to pinpoint the minimal functional requirements of context-sensitive information systems. Second, we have addressed practical issues in building context-sensitive input interpretation and output generation technologies for realistic applications. We have also exploited the synergy between the input and output technologies to support intelligent *and* robust user interaction, which is otherwise very difficult to achieve. Third, our evaluation results reveal the strength and limitations of our current work, and indicate future research directions.

Currently RIA is embodied in two applications: a residential real-estate application and a travel application. We use the real-estate application as our primary testbed for several reasons. First, we believe that house hunting exhibits the complexity of a typical information-seeking application. In such an application, users often need to cross-examine a wide range of interrelated data (e.g., houses, neighborhoods, and schools) before reaching a conclusion (e.g., houses worth visiting). Second, real-estate data is easily comprehensible by average users and is readily available, which allows us to experiment with realistic user scenarios. Third, there are online real-estate sites that we can directly use for comparison.

In the rest of paper, we start with a brief discussion of related work before reporting on our WOZ study. Based on the WOZ study results, we then explain key technical challenges, design rationale, and implementation of RIA. Last, we present a user evaluation of the working RIA system.

2. RELATED WORK

Our work is closely related to extensive efforts in both input analysis and output generation [16]. These works focus on specific techniques for understanding multimodal inputs (e.g., [29, 19, 12]) or for automating the creation of system outputs (e.g., [3, 7, 15, 13]). In contrast, our work described here is on the overall design, development, and evaluation of a full-fledged, intelligent information system that exploits the synergy between input analysis and output creation. Moreover, we emphasize the use of these technologies in support of practical applications that deal with large and complex data sets and diverse user interaction situations.

Another set of related work is information-seeking dialogue systems (e.g., [2, 11]) and natural language interfaces to databases [24]. However, none of them has gone to the extent as RIA does in developing and integrating context-sensitive multimodal input analysis with fully automated multimedia presentation generation.

Our work is also related to context-sensitive information retrieval systems (e.g., [27, 25]). Similar to these systems, RIA uses the interaction context (e.g., user interests) to aid users in information seeking. Unlike these systems, which focus on information retrieval from free text, RIA handles database retrieval. Accordingly, RIA is able to exploit a much finer-grained interaction context established by its input analysis and output creation engines. For example, RIA understands deeper semantics of a user request than other systems do.

Our work is built on previous studies on user information-seeking behaviors [18, 14, 26] and user interaction patterns [20]. While these studies guide the design of our system, we implement and validate these theories in practice.

3. WIZARD-OF-OZ (WOZ) STUDY

To observe realistic user interaction behaviors before creating a fully working system, we employed a WOZ technique that used a hidden human operator (wizard) to simulate envisioned RIA behaviors. The goals of our study were to 1) validate the effectiveness of the envisioned RIA, 2) identify RIA advantages and functional requirements, and 3) learn user interaction patterns and collect a corpus of actual user expressions for analysis and training. To achieve these goals, we first built a functional WOZ system to simulate our envisioned RIA features (e.g., supporting user queries in context). We then conducted a comparison study where we asked users to use both our WOZ system and an online real-estate site to complete similar information-seeking tasks.

3.1 WOZ System Setup

Given a user multimodal request, the wizard interprets the user's speech input and combines it with other inputs as needed (e.g., mouse pointing). The wizard then uses a GUI to compose a command (e.g., present <houses>) that drives the response creation. To optimize wizard response time, we use a limited set of parameterized commands to cover various user requests. We also perform run-time syntax checking on command parameters to minimize wizard input errors. Given a command, the system builds corresponding database queries, and then presents the retrieved data using pre-defined visual and verbal presentation templates.

3.2 A Comparison Study

To validate the effectiveness of our envisioned RIA and identify its advantages along with functional requirements, we compare the behaviors of our WOZ system with that of a conventional WIMP-based information system. To obtain meaningful results, we have designed our study scope and task scenarios such that they are realistic enough to emulate a practical application.

Part A: *Find houses satisfy the following criteria, and record their MLS, town name, school district, and tax.*

Location: White Plains, Mt. Kisco, and Briarcliff Manor
Price: 350K–550K; **Num of Bedrms**: 3+; **Style:** ranch
School district: have the highest graduation rate
Tax: lowest tax when all above criteria met

Part B: *Compare the median home price in Chappaqua with that of the town of your choice from Part A; then find houses satisfying*:

Location: Chappaqua; **Price**: 350K–550K; **Num of Bedrms**: 3+;
Square footage: biggest square footage when above criteria met

Figure 2. A sample task tested in our WOZ study.

3.2.1 Study Scope

We chose to compare our WOZ system with www.realtor.com, the official site of the National Association of Realtors. There are two types of typical information-seeking task: target search and browsing [14]. A *target-search task* involves finding a target that meets a set of well-defined search criteria (e.g., house price range and style). In contrast, a *browsing task* involves exploring a data space using fuzzy criteria (e.g., finding an ideal home). In this study, we chose to focus on target search tasks for three reasons. First, target search is well supported by both our WOZ system and by realtor.com. Second, target search facilitates an objective comparison (e.g., comparing task completion rate and time). Third, with a careful design, target search can cover typical user interaction behaviors exhibited in browsing, such as orienteering [26].

3.2.2 Task Scenarios

We recruited 18 users (including 2 pilot users) for our study. Each user session consisted of three parts. First, we profiled the user using a 9-item questionnaire, such as asking one's familiarity with the real-estate domain and with online real-estate sites. Second, we asked the user to perform two sets of house hunting tasks, one using our WOZ system, and the other using realtor.com. The goal was to find the target house and related information (e.g., the school district data) in the shortest time. Third, we asked the user 6 feedback questions. For each task, we recorded the correct target identification rate, task completion time, and the number of turns taken to complete the task. The maximal time allowed per task was 30 minutes. We videotaped every session for later analysis.

To test the usage of context, each task had two parts (A and B), where part B required certain information obtained from part A (Figure 2). To make task scenarios realistic, we used house hunting criteria acquired from several sources: 1) from a dozen eager home buyers in a separate study, 2) from local real-estate experts, and 3) from market surveys done by the National Association of Realtors. To eliminate potential biases like learning effects, we designed two similar but not identical sets of tasks. We also permuted the task order and the system usage order to test all combinations.

3.3 Study Results

Figure 3(a–e) show both objective and subjective evaluations, indicating that our WOZ system outperforms realtor.com in every category. An ANOVA test also identifies that the usage of system (using RIA or realtor.com) is the *only* statistically significant factor ($p < 1e\text{-}7$) that impacts user performance. To verify the practicality of our tasks, we asked users to rate how likely they would use the similar set of house hunting criteria in a real world. We obtained an average score of 6.41 on a 1 to 7 scale, with 7 being most likely.

Our comparison study not only confirmed the effectiveness of our envisioned RIA, but it also helped to identify two key advantages of such a system. First, the WOZ system allowed users to express their requests flexibly in context, which in turn helped them to easily express their search targets, update their search criteria, and switch their data foci. Second, the WOZ system tailored its multimedia responses to the interaction context, which helped users to easily distill important information and comprehend all relevant information as a coherent whole.

Our study also identified several additional user needs. First, it is desirable for RIA to provide explicit system feedback whenever possible. For example, since our WOZ system allowed users to revise their search criteria in context, the users preferred explicit confirmation of the complete search criteria in effect. Second, it is desirable for RIA to provide system guidance, since users may not know what they could ask or how to express their requests. Due to the diverse user interaction patterns, pre-defined system help may be insufficient. Third, besides delivering requested information, the system may need to explain the relation between what is being asked and what is being retrieved. Figure 4 shows an example of a simple system response that has confused some users. In this case, Mt. Kisco school district does not exist, and the city of Mt. Kisco is covered by the Bedford school district. Without explicitly explaining such data relationships, there is an information gap between what a user asks and what she receives. In addition, we found the template-based output design used in our WOZ system inadequate for handling diverse interaction situations. For example, we could not integrate retrieved houses with a spatial map due to a lack of dynamic layout and visual context management [28]. The wizard also had to improvise his verbal responses when a suitable template-based response was unavailable.

Based on our analysis, we summarize desired RIA functional

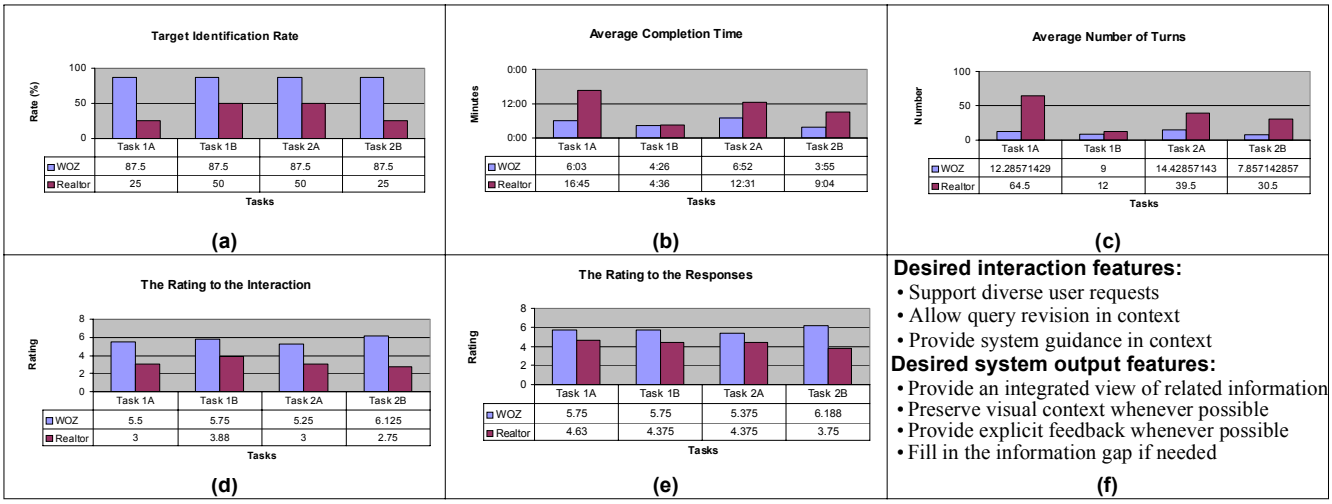

Figure 3. (a–e) are objective and subjective evaluations for two sets of 2-part (parts A and B) tasks. Subjective evaluations are rated on the scale of 1 (least favorable) to 7 (most favorable). (f) lists user-desired features collected from the study.

U3: Tell me about Mt. Kisco school district

WOZ: I found 1 school district, Bedford school district

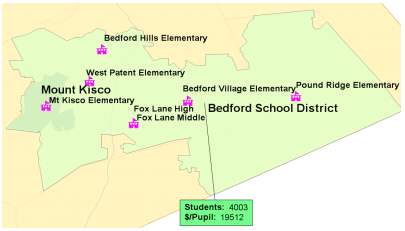

Figure 4. A potentially confusing system response.

requirements in Figure 3(f).

4. CONTEXT-SENSITIVE INFORMATION SYSTEM

We have built RIA to fulfill all the requirements listed in Figure 3(f) except the last one, since complex data inference may be required to detect the situation and is beyond our current scope of work. Based on our user feedback, we have also decided to focus on supporting text input instead of speech input to achieve the desired system robustness. We first describe RIA's overall architecture and main user-RIA interaction flow. We then explain RIA key technologies, emphasizing the design rationale that we have used to enable a robust and extensible RIA system.

4.1 RIA Architecture and Main Flow

RIA is a multi-threaded, distributed client-server system (Figure 5a). Starting from the client side, a user uses multiple modalities, including GUI, natural language, and deictic gestures, to express a data request. Such a request is then sent to the *process dispatcher* on the server side. The dispatcher manages the RIA component work flow by invoking each component in an order that is defined in a system configuration file. As each component is called, it updates the conversation history, which serves as a blackboard for all the components to exchange and share information.

Given a user request, the dispatcher first calls the *input interpreter*. In a user-initiated turn, the interpreter often creates a new conversation segment with a *user unit* that records the interpretation result. An interpretation result captures both the intention and attention of an input [6]. For example, the interpretation of U1 in Figure 1 is: seeking (intention) a set of houses (attention). The *conversation facilitator* is called next to suggest a set of conversation acts by weighing various factors, such as the properties of the retrieved data and communication obligations [21]. In the case of U1, the conversation facilitator decides to directly present the requested houses. This decision is then recorded in a *RIA unit*, which becomes part of the conversation history (Figure 5b).

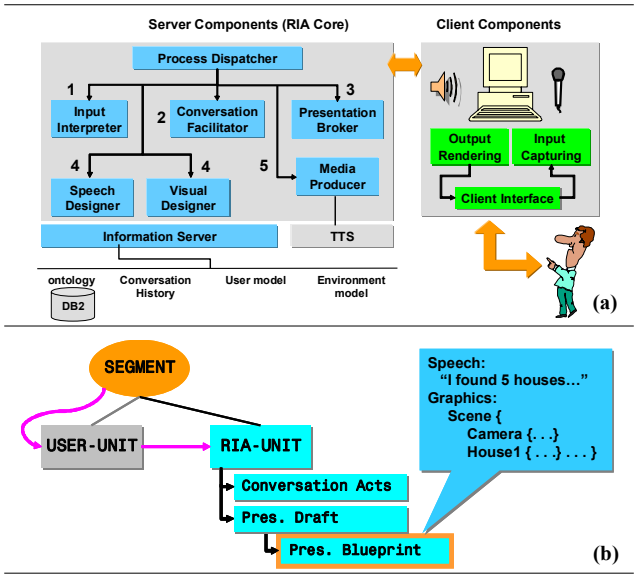

Figure 5. RIA Architecture and a snapshot of conversation history after step 4.

A *conversation act* usually indicates the type of a response without specifying the exact content (e.g., house attributes) or the form of the response. The presentation broker is then called to refine each act. The *presentation broker* handles content selection and media allocation, similar to the functions of the content layer defined in [6]. Consequently, it produces a *presentation draft* that defines the intended content [30] and the media usage [32]. Based on the draft, the *speech designer* and *visual designer* create media-specific responses. The final design is recorded in a data structure called *presentation blueprint* that defines all verbal and visual details. Figure 5(b) is a snapshot of the conversation history up to this point. The *media producer* is called last to send the presentation to be rendered on the client side. Currently, it sends over the audio produced by a Text-to-Speech (TTS) engine and a visual scene script (Figure 5).

All components interact with a common information server to access various data, including application data (e.g., houses in our real-estate application), conversation history, user model (e.g., user profile), and environment model (e.g., device capabilities).

As described above, RIA relies on two key sets of technologies: context-sensitive input interpretation and automated output generation. Since we have detailed these technologies elsewhere [6, 10, 23, 22, 30, 32, 31], here we explain our design rationale of honing these technologies for practical applications. Moreover, we emphasize the exploitation of the synergy between these technologies so that they can work cooperatively in RIA.

4.2 Context-Sensitive Input Interpretation

Based on our WOZ study, it is highly desirable to have a robust and accurate input interpretation engine that can understand diverse user expressions in context. Since building a general purpose interpretation engine is very difficult, RIA focuses only on understanding user information-seeking requests. Furthermore, we employ three complementary strategies to enable robust and accurate interpretation of diverse user data requests in context. First, we use a context-driven approach to optimize RIA interpretation by exploiting various contexts simultaneously, including data semantics, linguistic cues, and conversation history. Second, we provide system guidance in context to allow users and RIA to adapt to each other's expressions over time. Third, we leverage the strength of multiple modalities to achieve robust interpretation.

4.2.1 Context-driven interpretation

Currently, RIA focuses on user requests to databases. As we have observed (e.g., from our WOZ study), while these requests exhibit substantial syntactic variations, they share a common semantic structure. Based on this observation and the conversation theory [8], we use a set of semantic constructs to model a user request. Specifically, a user request includes two top-level constructs: intention and attention. *Intention* encodes the user information-seeking task (e.g., data access or comparison). *Attention* captures the data target of the intention, made up of lower-level constructs, such as data concepts/attributes to be retrieved, and a set of constraints that the retrieved data must satisfy. It also includes derived meta features that characterize the overall properties of a request. For example, feature followup signals whether a request is new or a follow on. Such features are used to tailor RIA responses to a specific user interaction flow (Section 4.3.3).

Since we focus on a finite set of information-seeking tasks, identifying the intention of a user request is relatively easy. Our main

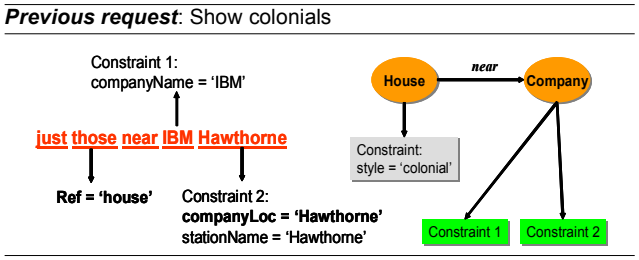

Figure 6. Partial semantic constructs of an input

effort is thus on interpreting the attention of a request. Similar to PRECISE [24], RIA focuses on identifying the semantic constructs of an input (e.g., data concepts and constraints) and the relationships between the constructs (e.g., relationships between two data concepts). However, unlike PRECISE, which handles only complete requests, RIA allows users to incrementally revise their requests in context instead of specifying a complete query every time. To derive the full meanings of incomplete (e.g., "*the cheapest*") or imprecise requests ("*what about Pleasantville*"), RIA exploits various contexts, including data semantics, linguistic cues, and conversation history [5, 10].

Figure 6 shows a partial interpretation made by RIA. To interpret this input, RIA first identifies various semantic constructs using a lexicon that is largely derived automatically from the databases. RIA then resolves references like "*those*" and semantic ambiguities such as in "*Hawthorne*". To do so, RIA uniformly models contextual cues as a set of constraints, including conversation history, data semantics, and syntactic information derived from a syntactic parser [17]. It then uses optimization-based approaches to derive the most probable interpretation of an input by maximizing the satisfaction of all constraints (e.g., probabilistic graph-matching in [5]). As a result, RIA is able to consider all constraints simultaneously, which in turn helps RIA to optimize its interpretation. Continuing our example above, RIA is able to resolve "*those*" to be the houses mentioned before, and identify that "*Hawthorne*" most likely be a company location instead of a train station.

A practical issue of this approach is to balance the satisfaction of various constraints, especially conflicting ones. For U3 in Figure 1(c), there are two conflicting constraints suggesting what "*those*" may refer to. A conversation history constraint hints that "*those*" likely refers to cities—foci of the previous turn. However, a semantic constraint may suggest that "*those*" likely refer to something else since there is no semantic relation between two sets of cities. When a conflict rises, RIA tries to satisfy the constraints with higher priorities. In the above example, RIA favors the semantic constraint and resolves "*those*" to be the houses mentioned before. In general, we assign priorities to constraints based on their reliability. For example, semantic constraints derived from a data ontology are assigned higher priority than that of dynamically inferred conversation history constraints.

As a result, RIA can handle a wide range of user expressions regardless their syntactic forms, ranging from keywords (e.g., "*colonials 3+ bedrooms*") to full English sentences, all in context. Such flexibility of RIA is much appreciated in a practical application, where RIA must accommodate various user linguistic styles, and tolerate imperfect user inputs (e.g., abbreviated and ungrammatical expressions). Moreover, our approach helps to minimize the effort for supporting new domains, since it does not require a large training corpus or a large set of syntactic rules. It is also easy to incorporate new constraints (e.g., pragmatic constraints) to drive the interpretation without changing the underlying algorithms.

4.2.2 Adaptive interpretation

Despite our effort described above to help achieve more accurate and robust interpretation, RIA's interpretation capability may still be insufficient for our targeted, real-world applications. Instead of directly improving RIA interpretation capability in a conventional way, we build a two-way adaptation engine that allows both users and RIA to dynamically adapt to each other's expressions during the course of interaction [23]. For example, RIA failed to understand U6 in Figure 7 initially, so it suggested five valid alternative queries using a set of criteria, including matched data semantics and surface expressions. Among the suggested queries, the user selected the second one from the list and edited it (U6') to suit her needs. Accepting U6' and observing the pair (U6, U6') over time, RIA can then use this association to handle a *similar* unsupported request like "*Find ranches in good school districts*". Consequently, the adaptation enhances the usability of RIA by turning a novice user into a power user, who can work effectively within RIA's capability. In addition, RIA improves its interpretation capability through self-adaptation, minimizing the overall effort of developing an effective interaction system. As our adaptation algorithms are detailed in [23], here we discuss two practical issues of integrating such mechanisms within a working RIA.

The first issue is *how often* RIA should invoke the adaptation engine. If RIA's error tolerance is too low, any input errors would trigger the adaptation process (e.g., typo "*then*" in *"show houses cheaper then $500k"*). As observed in our experiments, this may result in more user interaction turns and longer task completion time, since users must correct every error. In contrast, if the error tolerance is too high, RIA would skip unknown or problematic expressions without notifying the adaptation engine. As a result, RIA might misinterpret an input and would not be able to learn new expressions over time. To achieve a balance, we measure the error severity occurring in an input using multiple features, including the percentage of detected unknown words and the impact of such words. Through extensive experiments, we tune the error tolerance so that the adaptation engine is invoked only when unforgivable errors occur.

The second issue is *when* RIA should invoke the adaptation engine. RIA processes an input in multiple stages, including labeling input tokens using a lexicon and grouping different semantic constructs. It may identify input ambiguities or anomalies at any stage. For example, RIA detects the unknown word "*good*" in U6 (Figure 7) during the labeling stage, since "*good*" is not in the lexicon. Should RIA invoke the adaptation engine immediately or defer it to a later stage? In practice, calling the adaptation engine early could avoid unnecessary analysis. For example, even after a full analysis, RIA would not know better about the unknown expression "*river views*" in "*show houses with river views*". Without a full analysis, however RIA may make less accurate recommendations. Considering U6, without a full analysis RIA cannot associate "*good*" with "*school districts*". In turn, RIA would recommend a different set of queries. Currently, RIA invokes the adaptation engine after a full analysis, favoring the accuracy of query recommendations over computational efficiency.

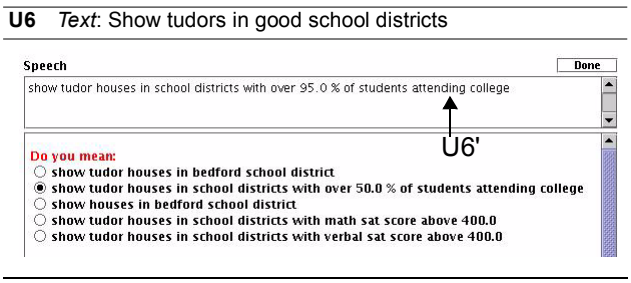

Figure 7. An example of input adaptation.

4.2.3 Leveraging GUIs and language inputs

Besides combining language inputs and deictic gestures as in other systems [12, 29, 19], we have explored the usage of GUIs to complement language inputs for two reasons. First, it is easier for users to use GUIs to express certain data requests (e.g., using a slider for dynamic data query [1]). Second, GUI inputs are explicit and thus help RIA to process the accompanying language input. By default, RIA interprets a user request in the context of previous requests. However, users may break from the previous conversation flow without explicitly signaling it in their language input. Consider a user input "*show houses in Armonk*" after U3 in Figure 1. It is unclear whether RIA should inherit the previous house constraints or simply start fresh. While RIA is able to detect some of these breaks, it also lets users use a GUI button to explicitly signal the start of a new flow. In fact, users can use different GUI buttons to control a conversation flow, including interrupting a RIA response (barging in), starting over (wiping out the entire conversation history), and going back (reverting to the previous turn).

4.3 Automated, Customized Output Generation

RIA is designed to support a highly dynamic user conversation, where it is difficult to predict how the conversation would unfold. It is thus impractical to plan in advance the content and forms of all possible RIA responses. To tailor RIA responses to a user interaction context, we develop a suite of automated response generation technologies. More importantly, we take practical issues into account when developing these technologies to achieve desired system coverage and extensibility. Specifically, we devise an optimization-based framework to select response content [30] and allocate suitable media [32]. We combine machine learning with other approaches to dynamically synthesize verbal and visual responses [22, 31]. In addition to tailoring responses to individual requests, RIA leverages user input patterns to customize responses to a specific user interaction flow. We also build fall-back mechanisms that allow RIA to act robustly, even in unanticipated situations (e.g., missing data or machine learning failure).

4.3.1 Optimization-based content and media selection

A *user interaction context* consists of a number of factors, including query expressions and conversation history. Any subtle variations of these factors, such as changes in data volume or query patterns, often require different response content or presentation media to be used. Since it may require a huge set of rules/plans to handle such diverse situations exhaustively, it is impractical to use conventional rule/plan-based approaches in RIA. Instead, we develop an optimization-based framework for content selection and media allocation [30, 32]. In this framework, we uniformly model all factors as presentation desirability/cost constraints (e.g., a presentation cost constraint derived from device properties). We then use optimization-based algorithms to maximize the satisfaction of all constraints. For example, we use a graph-matching algorithm to allocate media by maximizing the satisfaction of all allocation constraints [32]. As a result, RIA optimizes the content and media selection by dynamically balancing a comprehensive set of factors. Moreover, our approaches can be easily extended to cover new situations, since adding a new factor/constraint does not require modification of the underlying algorithms.

4.3.2 Example-based media-specific design

Similar to the reasons listed above, it is also impractical to use a rule/plan based approach [3, 7, 15, 13] to create media-specific outputs in RIA. Instead, we employ case-based learning to create both visual and verbal responses from a set of graphics and English sentence examples, respectively. Our learning engines not only can reuse suitable examples, but they can also compose new forms of outputs by dynamically combining different example fragments [22, 31]. As a result, RIA can cover a wide range of interaction situations using only a small number of examples. For example, it uses about 20 visual examples and 200 sentence examples for our real estate application that covers 25+ concepts, each with a number of attributes (e.g., a house has 40 attributes). The usage of a small example set helps to set up a system quickly. Moreover, we can easily extend RIA capability by adding new examples.

Nonetheless, a case-based learning engine alone is inadequate in meeting all RIA's needs. For example, it is inefficient to use case-based learning to abstract sentence aggregation rules, since it would require a large number of examples [22]. Similarly, case-based learning would not be used to learn precise visual arrangements from examples (e.g., exact positions and sizes). Such parameters must be recomputed for specific visual scenes. Therefore, we use case-based learning to learn overall presentation structures (e.g., visual or sentence structure) and use other approaches to fine tune presentation details. Specifically, our speech designer uses a rule-based approach to handle sentence aggregation [22]; and our visual designer uses various other means, including constraint-based and optimization-based approaches, to fine tune visual layout and manage visual context [28].

4.3.3 Context-sensitive response design using input features

A better understanding of a user input helps to create a more tailored response. To tailor its responses to a specific user interaction flow, RIA leverages its fine-grained interpretation results, especially the meta features derived about the request. Here we illustrate the use of two such features.

Feature followup is derived during RIA input interpretation to signal whether a given user request is new or a continuation of a previous request. In Figure 1, U2 is a follow-on of U1, since it inherits certain data constraints specified in U1. To maintain the desired level of semantic continuity between follow-up requests, the visual designer uses this feature to compute the amount of visual content overlap between two successive visual responses. In general, RIA maximizes the overlap between follow-on requests, while reducing the overlap when a new flow starts [28].

Another derived input feature navDirection, indicating the change of direction in user data navigation, also influences RIA response generation. When exploring a data space, a user often changes his data foci in several ways (Figure 8): filtering a data set (U1–U3), expanding a data set (U4), or switching to a different data set (U5). To tailor RIA responses to a user interaction flow, both our visual and verbal designers exploit this feature in their synthesis. First, navDirection helps the visual designer to decide the amount of visual context to be maintained between displays. For example, if RIA detects that a user is narrowing down a data set, it will reduce visual content overlap across displays to let users focus on the filtered data set [28]. Similarly, this feature helps the language designer to decide how much information it should repeat in successive verbal responses. To avoid repetitions, the language designer often generates progressively more terse expressions, such as ellipses, in response to a series of similar requests like continuous data filtering (R1–R3). It could also use this feature to generate more informative responses, confirming the current navigation direction (R3').

U1: 4 bedroom, 2 bathrm colonials
R1: I found 47 houses satisfying your criteria
U2: at least 2000 sq.ft.
R2: There are 26 houses
U3: built after 1990
R3: 1 house
R3': Based on your request, I have narrowed down to 1 house
U4: what about any style
R4: I found 3 houses
U5: Tell me about the schools

Figure 8. The verbal portion of a user-RIA conversation.

4.3.4 Robust response generation

Due to technology imperfections and unanticipated interaction dynamics, RIA may occasionally fail to produce a response. For example, our case-based learning engine may fail to find qualified examples to produce a legal verbal output. Unexpected missing or erroneous data, like missing house locations, could also throw RIA off its normal visual design course. To act robustly in such situations, RIA is equipped with a set of fall-back strategies.

First, RIA uses default presentation templates in case it fails to produce any output following its regular steps. Second, we always use a combination of mechanisms in case one fails. For example, we mix a constraint-based approach [9] with a procedural space management algorithm [4] to determine visual layout, since a constraint-based approach alone may not produce adequate solutions [9]. Third, we give users certain control to explicitly articulate their needs. For example, if a user happens to miss part of or all RIA verbal output, she can request RIA to repeat the verbal output. Similarly, the user can ask RIA to go back to the previous turn to review the visual output.

5. EVALUATION

We have tested RIA extensively on thousands of user queries in a real-estate application and installed it at the IBM Industry Solutions Lab for daily customer visits. To demonstrate the reusability and extensibility of RIA technologies, we have also built a travel application (e.g., looking for hotels and restaurants). The customization effort was relatively small (137 KB of declarative definitions), compared to the reusable portion (about 11 MB procedural code). For example, only 20 new sentence examples were added to extend RIA's verbal response coverage for the travel application.

In addition to evaluating RIA in focused areas (e.g., content selection [30] and media allocation [32]), we also study RIA as a whole to better understand RIA's capabilities and limitations. Specifically, we evaluate how RIA assists both power users and novice users in their browsing and target search tasks.

5.1 Experiments

We recruited two user groups: 2 *power users* who were trained to become knowledgeable about RIA and the real estate domain, and 18 *novice users* who never used RIA before and may or may not have house hunting experience. To facilitate comparison, we subscribed to the same multiple listing service as most major real-estate websites do. In this study, our test data set included 1800+ houses, 70+ cities, 400+ schools, and 100+ landmarks (e.g., parks).

Our experiments included two parts. First, we asked the two power users to act as realtors and to use RIA to complete a set of browsing tasks. Each power user interviewed 9 of 18 novice users individually, who pretended to be home buyers[1]. During their interaction, the power user used RIA to look for houses *dynamically* specified by a home buyer. For comparison purposes, we recorded the submitted search criteria. We also asked the two power users to satisfy the same criteria using online real-estate websites[2]. Second, we asked the 18 novice users to use RIA to complete a target search task satisfying 9 criteria, similar to the tasks tested in our WOZ study (Figure 2). At the end of the study, the users were asked 6 feedback questions to rate RIA usability.

5.2 Result Analysis

Overall, RIA performed adequately in both tasks, especially compared to existing online real-estate websites. In our first experiments, among 181 search criteria that users dynamically specified, RIA satisfied 79% of them (143/181), compared to 35% satisfaction rate achieved by any combination of online real-estate sites. Moreover, RIA achieved a satisfaction rate of 97% for 5 categories of most commonly specified criteria versus 61% achieved by the online websites (Figure 9). By our analysis, 80% of the 181 criteria fell into these five categories, while all 181 criteria spanning over 20 categories, ranging from house amenity (e.g., "*eat-in kitchen*") to location constraints (e.g., "*near train stations*"). Moreover, we analyzed the 38 criteria not supported by RIA. They fell into two categories: 24 requiring free-form text search not supported by RIA (e.g., "*cul-de-sac*"), and 14 due to unavailable data in our database (e.g., city library information).

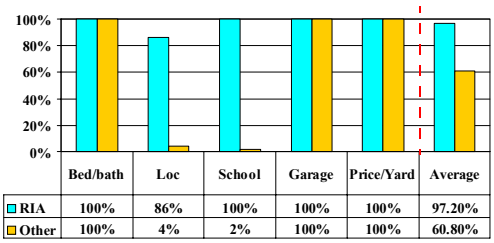

Figure 9. RIA satisfaction rate vs. any online site for 5 types of most commonly user-specified house hunting criteria.

In our second experiments, 16 out of 18 users completed the task using RIA. Of the two failed attempts, one found the wrong target due to human input error and the other did not find the target due to a system crash at the last user query. Due to the time and effort required, we did not ask the users to repeat their tasks using any online tools. Instead, we experimented with several online websites ourselves to complete the task. We did not find any existing site that can *directly* satisfy all 9 constraints without requiring manually stitching requested information at different steps. As a reality check, we compared these 9 constraints with the user-specified criteria collected in our first experiments. All 9 constraints appeared before, and 6 were mentioned by more than 50% of the users[3]. For the six subjective evaluation questions, our users provided positive feedback. On a scale of 1 (least) to 5 (best), our users produced the average scores of 3.86 and 4.3 when asked to rate RIA's input and output capabilities, respectively. All users but one answered that they were able to complete the tasks and obtain satisfactory results. Table 1 summarizes the users' most and least liked RIA features.

5.3 Result Summary

From our above analysis, we conclude that RIA is highly valuable to our users. Not only did all our users except two complete realistic tasks, but also all of them provided very positive feedback. More importantly, our analysis suggests that our technologies are adequate for supporting real-world applications, such as the real-estate application tested. We base this conclusion on the following

What do you like best about RIA
Integrated, incremental presentation of information (11 users)
Context-sensitive, natural language input (8)
Busy music (1)
What do you like least about RIA
Insufficient GUI interaction (e.g., editing accumulated criteria) (5)
Natural language understanding limitations (4)
System robustness (2)
Verbal and visual output coordination (2)
Verbal output (1)
Busy music (1)

Table 1. User most and least liked RIA features.

1 We did have several eager home buyers who wanted to be helped.

2 Whatever possible, they used combinations of multiple websites, including realtor.com, realestate.yahoo.com, and houlihanlawrence.com.

3 Here we consider the constraints by type regardless of their values (e.g., both 3 bedroom and 4 bedroom are bedroom constraints).

three observations. First, RIA supports diverse, truly "natural" query expressions and revisions in context. In our above study, RIA achieved 92% interpretation accuracy for about a total of 550 queries received. Second, RIA is able to automate the design and generation of tailored visual and verbal responses for highly dynamic interaction situations. For the 550 queries, RIA automatically produces desired responses for 95% of the queries[4]. Third, RIA greatly empowers "power users" who are familiar with RIA and the application domain. As demonstrated in our study, RIA helps power users to accomplish open-ended browsing tasks in real world conditions, which are extremely difficult to achieve using existing tools. Consequently, we expect that service providers, such as real-estate and travel agents, could be trained to become power users and use RIA to greatly help their customers.

Current RIA technologies also have their limitations. In our study, we observed highly diverse user interaction patterns even for deterministic target search. Despite the obvious advantages of supporting flexible language inputs, it is also imperative to optimize the use of other input modalities. Table 1 shows that 5 users prefer to directly edit the RIA-confirmed search criteria currently shown in plain text (see video). Supporting such interaction not only aids users in their query revisions, but also reduces the burden on NLU. Another limitation of RIA is its current focus on database queries. Our study shows that it is desirable for RIA to handle various data forms (e.g., supporting free text search for house amenities). Moreover, RIA currently presents the requested data as is and this may not scale up for a domain where a large quantity of data is often retrieved. Ideally, RIA should present better data summaries in such cases to guide users in their further exploration.

6. CONCLUSIONS

A *context-sensitive information system* can understand user information requests and present the requested information in context. We have hypothesized and confirmed that such an interaction paradigm can significantly improve users' ability to perform information seeking tasks. We have presented our work in three parts. First, we used a Wizard-of-OZ study to demonstrate the effectiveness of our hypothesized system and to identify additional requirements. Second, we described the development of RIA, a fully automated context-sensitive information system based on the findings from our study. Moreover, we highlighted two sets of core RIA technologies, context-sensitive multimodal input interpretation and multimedia output generation, which work together to enable robust and intelligent information seeking. Finally, we presented a formal study that evaluated the usability of RIA on realistic information-seeking tasks. Our study shows that RIA does have a great appeal for users. It also demonstrates the practicality of our approaches for supporting real-world applications. Hopefully, our success will inspire and help others to follow in our footsteps and build more RIA-like systems in the future.

References

[1] C. Ahlberg and B. Shneiderman. Visual information seeking: Tight coupling of dynamic query filters with starfield displays. In *ACM Proc. SIGCHI '94*, pages 313–317, 1994.

[2] J. Allen, G. Ferguson, and A. Stent. An architecture for more realistic conversational systems. In *IUI '01*, pages 14–17.

[3] E. Andre and T. Rist. Generating coherent presentations employing textual and visual material. *AI Review*, 9:147–165, 1995.

[4] B. Bell and S. Feiner. Dynamic space management for user interfaces. In *Proc. UIST '00*, pages 239–248.

4 Output is considered correct if it matches the interpretation results.

[5] J. Chai, P. Hong, and M. Zhou. A probabilistic approach to reference resolution in multimodal user interfaces. In *IUI'04*, pages 70–77.

[6] J. Chai, S. Pan, M. Zhou, and K. Houck. Context-based multimodal input understanding in conversation systems. In *ICMI '02*, pages 87–92.

[7] S. Feiner and K. McKeown. Automating the generation of coordinated multimedia. *IEEE Computer*, 24(10):33–41, October 1994.

[8] B. Grosz and C. Sidner. Attention, intention, and the structure of discourse. *Journal of ACL*, 2(3):175–204, 1986.

[9] H. Hosobe. A modular geometric constraint solver for user interface applications. In *Proc. UIST '01*, pages 91–100.

[10] K. Houck. Contextual revision in information-seeking conversation systems. In *Proc. ICSLP '04*, pages 201–204.

[11] M. Johnson, S. Bangalore, G. Vasireddy, A. Stent, P. Ehlen, M. Walker, S. Whittaker, and P. Maloor. MATCH: An architecture for multimodal dialogue systems. In *ACL '02*, pages 376–383.

[12] M. Johnson, P. Cohen, D. McGee, S. Oviatt, J. Pittman, and I. Smith. Unification-based multimodal integration. In *ACL'97*, pages 281–288.

[13] S. Kerpedjiev, G. Carenini, S. Roth, and J. Moore. Integrating planning and task-based design for multimedia presentation. In *IUI' 97*, pages 145–152.

[14] G. Marchionini. *Information Seeking in Electronic Environments*. Cambridge University Press, 1995.

[15] M. Maybury. Planning multimedia explanations using communicative acts. In M. Maybury, editor, *Intelligent Multimedia Interfaces*, chapter 2, pages 60–74. AAAI Press/The MIT Press, 1993.

[16] M. Maybury and W. Wahlster, editors. *Readings in Intelligent User Interfaces*. Morgan Kaufmann, 1998.

[17] M. McCord. Slot Grammar: A system for simpler construction of practical natural language grammars. In R. Studer, editor, *Natural Language and Logic: Intl. Scientific Symposium, Lecture Notes in Computer Science*, pages 118–145. Springer Verlag, 1990.

[18] R. Morris. Toward a user-centered information service. *J. of American Society for Information Science*, 45(1):20–30, 1994.

[19] J. Neal, C. Thielman, Z. Dobes, S. Haller, and S. Shapiro. Natural language with integrated deictic and graphic gestures. In M. Maybury and W. Wahlster, editors, *Readings in Intelligent User Interfaces*, pages 38–51. 1998.

[20] S. Oviatt. Multimodal interactive maps: Designing for human performance. *Human-Computer Interaction*, 12:93–129, 1997.

[21] S. Pan. A multi-layer conversation management approach for information-seeking applications. In *Proc. ICSLP '04*, pages 245–248.

[22] S. Pan and J. Shaw. SEGUE: A hybrid case-based natural language generator. In *Proc. INLG '04*, pages 130–140, 2004.

[23] S. Pan, S. Shen, M. Zhou, and K. Houck. Two-way adaptation for robust input interpretation in practical multimodal conversation systems. In *IUI '05*, pages 25–32.

[24] A. Popescu, O. Etzioni, and H. Kautz. Towards a theory of natural language interfaces to databases. In *IUI '03*, pages 149–157.

[25] X. Shen, B. Tan, and C. Zhai. Context-sensitive information retrieval with implicit feedback. In *SIGIR '05*, pages 43–50.

[26] J. Teevan, C. Alvarado, M. Ackerman, and D. Karger. The perfect search engine is not enough: A study of orienteering behavior in directed search. In *CHI' 04*, pages 415–422.

[27] J. Teevan, S. Dumais, and E. Horvitz. Personalizing search via automated analysis of interests and activities. In *SIGIR'05*, pages 449–456.

[28] Z. Wen, M. Zhou, and V. Aggarwal. An optimization-based approach to dynamic visual context management. In *InfoVis '05*. To appear.

[29] M. Zancanaro, O. Stock, and C. Strapparava. Multimodal interaction for information access: Exploiting cohesion. *Computational Intelligence*, 13(7):439–464, 1997.

[30] M. Zhou and V. Aggarwal. An optimization-based approach to dynamic data content selection in intelligent multimedia interfaces. In *Proc. UIST '04*, pages 227–236. ACM, 2004.

[31] M. Zhou and M. Chen. Automated generation of graphic sketches by examples. In *IJCAI '03*, pages 65–71.

[32] M. Zhou, Z. Wen, and V. Aggarwal. A graph matching approach to dynamic media allocation in intelligent multimedia interfaces. In *IUI '05*, pages 114–121.