RC23259 (W0406-149) June 23, 2004
Computer Science

IBM Research Report

An Optimization-Based Approach to Dynamic Data Content Selection in Intelligent Multimedia Interfaces

Michelle X. Zhou, Vikram Aggarwal
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

An Optimization-based Approach to Dynamic Data Content Selection in Intelligent Multimedia Interfaces

Michelle X. Zhou   **Vikram Aggarwal**
IBM T. J. Watson Research Center
19 Skyline Dr.
Hawthorne, NY 10532
{mzhou, vikram}@us.ibm.com

ABSTRACT

We are building a multimedia conversation system to facilitate user information-seeking in large and complex data spaces. To provide tailored responses to diverse user queries introduced during a conversation, we automate the generation of a system response. Here we focus on the problem of determining the data content of a response. Specifically, we develop an optimization-based approach to content selection. Compared to existing rule-based or plan-based approaches, our work offers three unique contributions. First, our approach provides a general framework that effectively addresses content selection for various interaction situations by balancing a comprehensive set of constraints (e.g., content quality and quantity constraints). Second, our method is easily extensible, since it uses feature-based metrics to systematically model selection constraints. Third, our method improves selection results by incorporating content organization and media allocation effects, which otherwise are treated separately. Preliminary studies show that our method can handle most of the user situations identified in a Wizard-of-Oz study, and achieves results similar to those produced by human designers.

Keywords: intelligent multimedia interfaces, automated generation of multimedia presentations, content selection.

1. INTRODUCTION

To aid users in exploring large and complex data spaces, we are building a framework, called *Responsive Information Architect* (RIA), which engages users in a dynamically generated multimodal, multimedia conversation. Currently, RIA is embodied in a real-estate application testbed that is designed to assist potential buyers in finding residential properties. RIA allows a user to express her information requests using multiple modalities, including text, speech, and gesture (e.g., Figure 1). Based on the understanding of a user request [8], RIA creates a response in two steps: formulating conversation acts and authoring a multimedia response to realize these acts.

Conversation acts are abstract directives, indicating the desired types of RIA responses. For example, RIA formulates a conversation act Describe <House> to respond to query U1 (Figure 1). This act directs RIA to present the requested houses but without specifying the exact content (e.g., house attributes) or the form of the presentation. To realize such an abstract act, RIA must first decide the data content of the response. In this case, based on a user profile RIA chooses to convey house size and cost information, such as the number of bedrooms and price (Figure 1a). Moreover, based on the query expression RIA includes the location of the Phelps Memorial hospital to provide the spatial context for the houses. After determining the content, RIA decides the form of the response using suitable media and presentation techniques. In Figure 1, RIA uses speech to convey the number of retrieved houses and graphics to depict spatial information, such as house locations and city boundaries. Finally, a multimedia response is synthesized and presented to the user.

U1	*Speech*: Show houses near Phelps Memorial Hospital
R1	*Speech*: I found 3 houses near Phelps Memorial Hospital *Graphics*: Display (a)
U2	*Speech*: Tell me more about it *Gesture*: Point to the house on the right
R2	*Speech*: Here are the attributes of this 6-bedroom home. *Graphics*: Display (b)

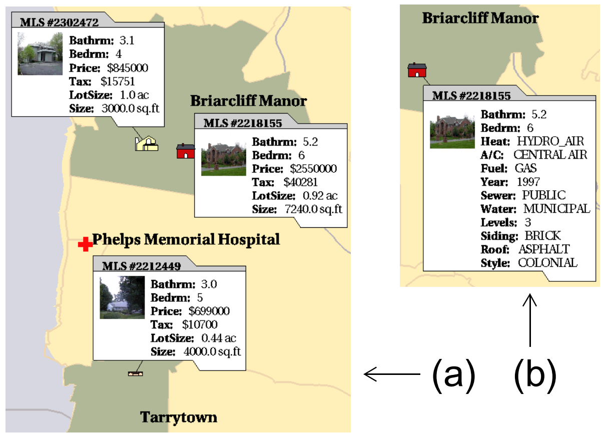


Figure 1. A recorded user-RIA conversation fragment: U1 and U2 are user queries, R1 and R2 are corresponding RIA-generated responses.

As demonstrated by this example, RIA must tailor a response to a number of factors at run time, including user interests and user queries. Since it is difficult to predict how a conversation would unfold, it is impractical to plan all possible responses, including their content and forms, in advance. Thus, we automate the generation of RIA responses. Here we focus on *data content selection*, a process that dynamically chooses data content in response to user queries. To handle diverse user queries and unanticipated information introduced in a conversation, we model content selection as an optimization problem, whose objective is to maximize the satisfaction of all relevant constraints (e.g., content quality and quantity constraints). To the best of our knowledge, we believe that our work is the first to address content selection under an optimization-based framework. As a result, our approach offers three unique contributions:

1) We present a general solution to content selection that dynamically chooses the desired content for a wide variety of interaction situations.
2) We model selection constraints using extensible, feature-based metrics. This model enables us to easily adapt our approach to new situations (e.g., a new user task).
3) We improve selection results by incorporating relevant content organization (e.g., data grouping) and media allocation effects (e.g., using suitable media).

We start with a brief discussion of related work and a RIA overview. We then describe our optimization-based approach to content selection, highlighting the aspects mentioned above. Finally, we present our evaluation results.

2. RELATED WORK

Our work on content selection is closely related to content planning in automated multimedia presentation systems [5, 12, 17, Section II in 18]. Since most of these systems support limited user interaction (e.g., [14] uses a pre-designed menu, and [12] handles one patient at a time), they often use a rule-based or schema-based approach to select content. In contrast, RIA is designed to support context-sensitive user queries to large and complex data sets, where developing an exhaustive set of selection rules or plans is impractical. Hence, RIA selects content by dynamically balancing a number of factors (e.g., being informative vs. being relevant to a query).

Content selection has also been addressed extensively in natural language generation [7, 25, 6, 26, 23]. These approaches focus on selecting content by specific factors, such as content importance [25], user knowledge [7], user preferences [6], or user tasks [26, 23]. In contrast, our approach weighs all these factors simultaneously to support user data exploration in different interaction contexts.

Another piece of related work is using utterance history to select data attributes for a spoken output [20]. Compared to this work, RIA uses a more comprehensive set of factors including the utterance history, to rank the desirability of data attributes. As a result, RIA can effectively handle data queries like U1 (Figure 1). Since this is the first query and it does not mention any house attributes, it is unclear how [20] would respond to U1.

Many information-seeking dialogue systems also address response generation. Similar to formulating conversation acts in RIA, they focus on dialogue strategies (e.g., making a clarification or a presentation) [9, 2]. They do not address the content selection issues presented here, as response content is already encoded in each strategy (e.g., in a schema [2]). Our work also differs from interactive visual information-seeking systems (e.g., [1, 13]), where either users must explicitly define the data content to be viewed/manipulated [13] or a template is used to present the retrieved data [1].

3. RIA SYSTEM OVERVIEW

Figure 2 provides an overview of RIA core components. RIA uses an array of recognizers (e.g., speech and gesture recognizers) and a multimodal *interpreter* to understand a user input. An interpretation result captures both the intention and attention of the input. In Figure 1, the interpretation of U1 is to seek (intention) a set of houses (attention). Exploiting a rich context (e.g., data semantics and conversation history), our interpreter can also handle abbreviated and imprecise inputs [8].

Given an interpretation result, the *conversation facilitator* suggests a set of conversation acts by weighing various factors, such as data properties (e.g., data volume) and communication obligations. Assume that U1 (Figure 1) results in a large data set. The facilitator may formulate two acts: Apologize (too much data to present) and Describe (e.g., showing the first *N* houses). Depending on the context (e.g., dealing with an experienced buyer), the facilitator may Ask the user for additional constraints, such as her preferred style, instead of showing any houses.

Since conversation acts often do *not* specify the exact content (e.g., house attributes) or the form or a response, they are passed to the *presentation broker* for refinement. Our presentation broker handles content selection/organization, media allocation, and media coordination, similar to the functions of the content layer defined in [5]. Consequently, it produces a response outline that precisely defines the intended content and the media usage (e.g., speech to express the number of retrieved houses and graphics to depict house locations). Based on this outline, the *visual designer* and *speech designer* work together to create a coordinated multimedia response [21, 28].

RIA relies on an *information server* to supply and manage various types of information, including domain data (e.g., houses in real-estate domain), conversation history (exchanges between RIA and a user), user model (e.g., user profile), and environment model (e.g., media capabilities).

4. EXAMPLES

We use a set of examples to explain how different factors influence the choice of data content in response to a user query. First, data volume (the size of the result set for a user query) impacts content selection. Normally, data volume is inversely proportional to the amount of information presented per instance due to resource limitations (e.g., screen real-estate). For example, Figure 3(a–b) reveal more

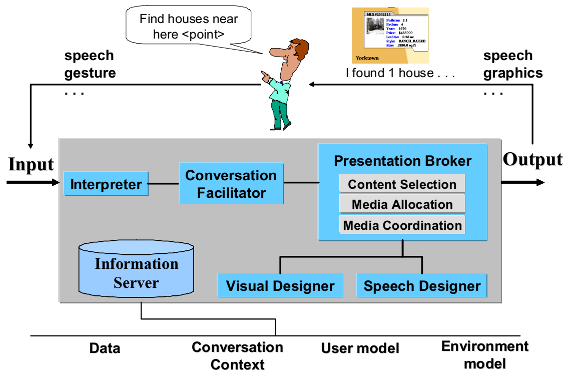
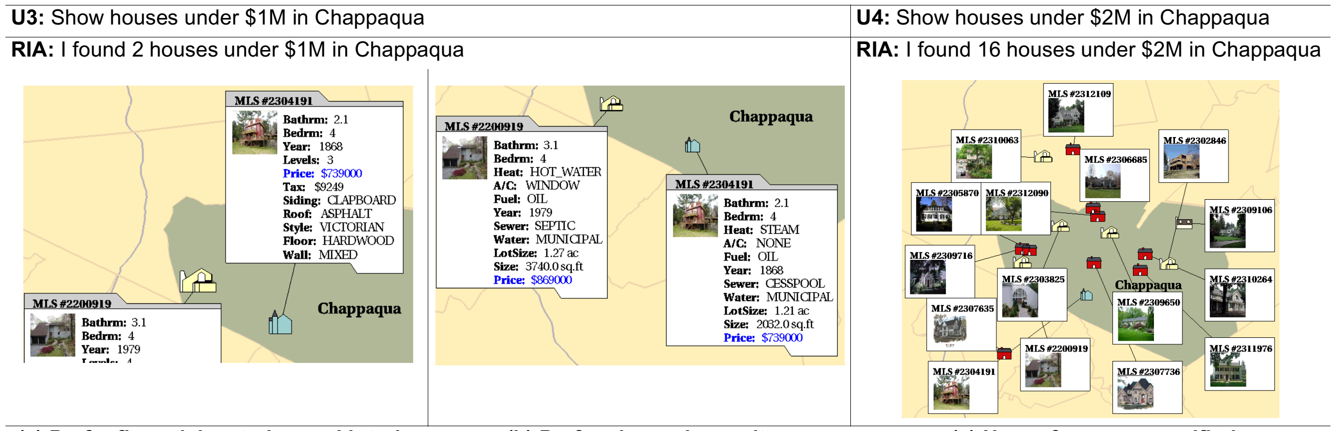


Figure 2. RIA core components.


U3: Show houses under $1M in Chappaqua

RIA: I found 2 houses under $1M in Chappaqua

U4: Show houses under $2M in Chappaqua

RIA: I found 16 houses under $2M in Chappaqua

MLS #2304191
Bathrm: 2.1
Bedrm: 4
Year: 1868
Levels: 3
Price: $739000
Tax: $9249
Siding: CLAPBOARD
Roof: ASPHALT
Style: VICTORIAN
Floor: HARDWOOD
Wall: MIXED

MLS #2200919
Bathrm: 3.1
Bedrm: 4
Year: 1979

MLS #2200919
Bathrm: 3.1
Bedrm: 4
Heat: HOT_WATER
A/C: WINDOW
Fuel: OIL
Year: 1979
Sewer: SEPTIC
Water: MUNICIPAL
LotSize: 1.27 ac
Size: 3740.0 sq.ft
Price: $869000

MLS #2304191
Bathrm: 2.1
Bedrm: 4
Heat: STEAM
A/C: NONE
Fuel: OIL
Year: 1868
Sewer: CESSPOOL
Water: MUNICIPAL
LotSize: 1.21 ac
Size: 2032.0 sq.ft
Price: $739000

(a) Prefer financial, exterior, and interior **(b) Prefer size and amenity** **(c) No preferences specified**

Figure 3. Example queries and responses (Cropped screen shots focus on the houses retrieved).

house attributes for 2 retrieved houses than Figure 3(c) does for 16 houses.

Data properties influence what to convey. In Figure 3, RIA includes house locations due to their importance in the real-estate domain, and selects house images for their ability of conveying rich information. Data properties dictate content selection when other factors are not present. In Figure 3(c), no user preferences are specified, and RIA selects content mainly by data properties, such as importance and informativeness (the amount of information being contained). In addition to individual data properties, data relations impact content selection. For example, it is preferable to present the number of bedrms and bathrms together (Figure 1). It is also desirable to convey house attributes (e.g., image) with a Multiple Listing Service (MLS) number to facilitate data identification (e.g., users may refer to the houses by their MLSs) [24]. However, it is undesirable to present MLS numbers alone, as they carry little information.

Response content should also be tailored to user interests [6, 26]. For example, for the same query RIA conveys different data to suit different user interests (Figure 3a–b). For one user who is interested in financial, exterior, and interior aspects, RIA chooses data, such as tax, siding, and wall (Figure 3a). For the other who cares for size and amenities, RIA selects data, like lot size and heat (Figure 3b).

User queries impact content selection, since they often imply a user's interests to which the responses should be tailored to. In Figure 1, query U1 may imply that the user be interested in the relationships between the houses and the hospital. Accordingly, RIA responds by incorporating the relevant hospital information (Figure 1a).

Conversation history also influences content selection. Query U2 follows up U1 (Figure 1). Based on the conversation history, in this case RIA introduces new content (e.g., year in Figure 1b), while keeping the important content to maintain the response continuity (e.g., showing house locations and bedrms in Figure 1a–b).

In short, when determining data content for a user query, RIA considers a number of factors, including data properties and user interests. Generally, any subtle variations in these factors, such as changes in data volume or query patterns, may require different content sets to be selected, which in turn prompt different responses.

5. DATA CONTENT SELECTION

To handle all the situations described above and all their possible variations, it is impractical to use a rule-based or plan-based approach, which would require an exhaustive set of selection rules or plans. Alternatively, we develop an optimization-based approach to dynamically decide content based on an interaction context, such as the specific user interests and given presentation resources. We explain our approach in three steps. First, we describe our context representation. Second, we define a set of feature-based metrics to model various context-sensitive selection constraints. Specifically, these metrics dynamically measure the presentation desirability and cost of data content. Third, we present a greedy algorithm that uses these metrics to select content such that its overall desirability is maximized and the total cost is within a given presentation budget.

5.1 Interaction Context Representation

Based on previous work on creating multimedia presentations [22, 24, 17] and information-seeking dialog systems [26, 20], we model an interaction context from five aspects: domain data (e.g., houses and cities), environment (e.g., media capabilities), query, user, and conversation history.

Data Model

Since RIA handles user queries to databases, we model the properties of structured data. For each application, we define a *data space*, which is made up of concept spaces. For example, a real-estate data space includes concepts, such as house, city, and school, while a travel application space may contain concepts, such as flight, hotel, and rental car. A *concept space* contains a set of *data dimensions*, each of which describes a specific aspect of the concept. For example, a house space contains data dimensions, such as price and style, and a city space has dimensions, including name and population. Each dimension is characterized by a set of features that specify the dimension's semantic properties (e.g., price is an attribute of house) and meta properties (e.g., price is quantitative data). We use an ontology to represent a data space, consisting of concept spaces, dimensions, and dimension features. Here we focus on features that directly impact content selection (Table 1). In particular, these features characterize a data dimension from its presentation desirability and cost perspective, and most of them are dynamically evaluated in context.

Feature	Definition
Content quality	
objectiveness	degree of objectivity of the contained information
Content quantity	
informativeness	the amount of information contained
availability*	how well the dimension is populated in a database
Content relevance	
importance	natural importance to the domain
I-relevance*	relation to a user's interests
K-relevance*	relation to a user's knowledge
Q-relevance*	relation to the current query
H-relevance*	relation to the interaction history
dependency	relations to any other dimensions
media-suitability*	relation to the fitness of a medium
media-capability*	relation to the capability of a medium
Cost	
space-cost*	number of pixels needed to display the data
time-cost*	number of words needed to speak the data

Table 1. Dimension features (*dynamically computed).

Environment Model

To tailor a response to a particular application environment, we model two media-related properties: media availability and presentation budget. Normally, RIA uses visual and spoken media, and adheres to space and time budgets.

Space budget *sBudget* counts the usable screen space in pixels: $sBudget = f \times (H\text{-}h) \times (V\text{-}v)$, where H and V are horizontal and vertical pixels, h and v are used pixels (e.g., by GUI trims), and f is a slack coefficient that adjusts the actual usable space. We set $f = 0.6$ to prevent visual clutter.

Time budget *tBudget* limits the maximal time (in seconds) during which a RIA spoken output can last. Based on the human attention studies [27] and our own experiments, we set *tBudget* to 20secs to avoid overloading a user's working memory.

User Query

We represent a user query in RIA using a 5-tuple:

Query $= <T, F, C, D, S>$[1].

Here T represents the user task; F indicates whether it is a new query or a follow-up; C and D denote the data concept and dimensions to be queried; and S is a set of constraints that the retrieved data must satisfy. In Table 2, Q1 requests houses that satisfy three constraints, and Q2 follows up Q1, asking for taxes of the houses that meet an additional constraint (Constraint4). We represent a constraint in a 4-tuple:

Constraint $= <Dc, relOp, V, St>$.

Here *Dc* is the constrained data dimension (e.g., style in Q1), *relOp* is the relation operator (e.g., equality operator ==), *V* is the constrained value (e.g., COLONIAL), and *St* indicates the status of the constraint: new (e.g., Constraint4 in Q2), or inherited (e.g., Q2 inherits 3 constraints from Q1).

Based on the user tasks (*T*), queries fall into three categories: data access, analysis, and lookup queries. *Data access queries* ask for a specific set of data instances (e.g., Table 2a). On the other hand, *data analysis queries* look for aggregate properties of desired data sets, such as count, min, and max. In Table 2, Q3 seeks the count of the desired school districts, and Q4 requests the price range of specific houses. In addition, *data lookup queries* enable users to examine known data entities (e.g., Table 2c). RIA now supports a wide variety of data access and lookup queries, but limited data analysis queries.

User Model

We model a user from two aspects: the user's knowledge and the user's interests. In particular, we model both aspects based on data factors. Here a *data factor*, containing a sub-set of data dimensions, describes a collective aspect of a concept. For example, the house financial factor includes two dimensions: price and tax. A dimension may be related to multiple factors, e.g., style belongs to both house exterior and interior factors. The relations between dimensions and data factors are defined in our ontology.

Based on our notion of data factors, a user's knowledge of a domain is a union of all factors that she knows about. For example, if a user indicates that she has owned a house before, RIA then assumes that she knows about house exteriors and interiors. Similarly, a user's interests are a union of factors that she cares for, e.g., caring for house financial and size factors. Currently, RIA uses a form-based questionnaire to acquire a user's knowledge and interests of a domain when the user logs in.

Modeling a user's knowledge and interests is a complex task [11]. More sophisticated models are always desired to gain a better understanding of a user. Nonetheless, our simple model helps to show how we can systematically incorporate a user model into a content selection process. Moreover, we use data factors instead of data dimensions in the user model to simplify the user profiling process. For example, RIA asks users to indicate their interests from a list of 6 house factors, instead of requesting them to choose among 40 house dimensions.

Conversation History

Based on the conversation theory [16], we model a conversation history to record the detailed exchanges between a

(a) Task = Data access queries
Q1: Show colonials under $300K in cities with over 5000 people
Concept = House; Dimension = null, Followup = false Constraint1: style == COLONIAL, Constraint2: price < $300K Constraint3: located-in ?C== [City, Constraint: population>=5K]
Q2: Just show taxes of those with brick siding
Concept = House; Dimension = {Tax}, Followup= true Constraint4: siding = BRICK; Constraints1–3: from Q1
(b) Task = Data analysis queries
Q3: How many school districts have students over 2000
Concept = SchoolDistrict; Dimension = {Count}
Q4: What is the price range for ranches in Armonk
Concept = House; Dimension = {MinPrice, MaxPrice}
(c) Task = Data lookup queries
Q5: What is the population here <pointing to a location>
Concept = City; Dimension = {Population}
Q6: Show Phelps Memorial Hospital
Concept = Hospital; Dimension = null

Table 2. Data queries and their representations.

1 Conjunctive queries, such as "show houses and cities", can always be decomposed into queries concerning a single main concept at a time.

user and RIA. Abstractly, each exchange consists of a user act (e.g., a query or a reply) and the corresponding RIA act (e.g., a direct reply or a follow-on question). Since we have described our conversation history model in [8], we do not elaborate it here.

5.2 Feature-based Desirability and Cost Metrics

As described in Section 3, the presentation broker takes a set of conversation acts as its inputs. Based on their content, we roughly divide these acts into two categories: acts that convey social messages and acts that present data. RIA uses a template-based approach to select social messages and an optimization-based approach to dynamically decide data content. Initially, the intended data content is defined by the current user query. However this information is often insufficient for RIA to create a response, since the query may not fully define the presentation content[2]. In Table 2, Q1 does not explicitly specify the dimensions D to be retrieved; in Figure 4, U5 defines D only partially, as additional dimensions, such as MLSs and house locations, are needed to produce an effective response. It is undesirable for RIA to convey *all* dimensions by default, since each data entity may contain a large number of dimensions. To choose proper data dimensions, RIA employs a set of selection constraints. Specifically, these constraints dynamically measure the presentation desirability and cost of a data dimension.

Presentation desirability metrics

We define a set of metrics to compute the presentation desirability of a data dimension by three Gricean maxims for effective information communication [15]: content quality, quantity, and relevance. We normalize all desirability scores to lie between [0, 1], with 1 being the most desirable.

Measuring content quality. The quality maxim states that we convey truthful information. For this purpose, we use feature objectiveness to measure how objectively a dimension conveys information. Currently, we approximate data objectiveness by a 3-level scale. If a dimension encodes a fact (e.g., location), its objectiveness is 1. In contrast, if a dimension conveys a subjective opinion (e.g., the house seller's remarks), its objectiveness is 0. The objectiveness is 0.5 for everything else falling in between (e.g., images taken by the seller). Our current content quality metric is solely based on information objectiveness:

Formula 1. $F_1(d) = objectiveness(d)$.

This states that the more objective the information is, the better the content quality is.

U5: Show prices of houses under $1M in Chappaqua

RIA: I got 2 houses under $1M in Chappaqua. Here are their prices

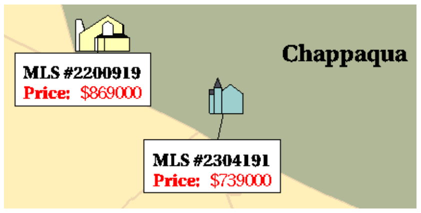


Figure 4. A varied query pattern and its response.

2 In a database application, normally the developers pre-select the interested dimensions, if a query does not explicitly specify any.

Measuring content quantity. The quantity maxim asserts that we make presentation as informative as needed. To assess how much information a dimension conveys, we compute feature informativeness. In particular, we define the informativeness of a data dimension within a concept space. Assume that the informativeness is 1.0 for a concept space with N dimensions. Within this space, the informativeness of a dimension is $1/N$ (e.g., price in the house space). However, the informativeness of a complex dimension, such as image, is M/N, where M is the average number of dimensions conveyed by the complex dimension, e.g., an image may convey house shape and color.

Measuring informativeness is meaningless if the value of a dimension does not exist. For example, images are not populated for all houses in our database. To accurately assess content quantity, we use feature availability to compute how well dimension d (e.g., price) is populated for a particular set of instances in a database. Specifically, we compute availability by each query Q:

$availability(d, Q) = K/N$, where N is the total number of instances requested by Q, and K is the number of instances out of N that have dimension d populated.

Using both informativeness and availability metrics, we calculate content quantity:

Formula 2. $F_2(d)=informativeness(d)\times availability(d, Q)$.

Our metric states that the more information that a dimension can convey, the higher its content quantity is.

Measuring content relevance. The relevance maxim states that we convey relevant information. Based on previous work on creating tailored data presentations [4, 7, 25, 6, 26, 20], we assess content relevance from 5 aspects: domain, user, query, history, and media. We select content important to a domain (domain relevance) and suitable for a user's knowledge and interests (user relevance). Moreover, we tailor the content to answer a query (query relevance) in context (history relevance), and ensure that the content be presentable (media relevance).

Domain relevance. It is desirable to convey important content [25]. Feature importance measures how important a data dimension is in a concept space. We learn the importance of data dimensions offline by analyzing existing applications. For example, we analyze real-estate websites (e.g., www.realtor.com and www.century21.com) to study the importance of various house dimensions (e.g., price and style). In particular, at each site we assess the importance of a dimension by its presentation prominence, including when it is presented (e.g., at the first click), where it is presented (e.g., at the top of a page), and how it is presented (e.g., in a bold face font). The final importance of a dimension is the average of its importance scores over many sites. We normalize the importance to lie between [0, 1], with 0 being unimportant, and 1 being the most important. Thus, we define domain relevance, $R(d, Domain)$ as:

Formula 3. $R(d, Domain) = importance(d)$.

User relevance. A desired response must be tailored to a user's knowledge [7] and interests [6, 26]. To promote dimensions that a user is familiar with or interested in, we measure user relevance, $R(d, U)$, how relevant dimension d is to a user U. Based on our user model, we compute $R(d, U)$ using two features: K-relevance and I-relevance, which assess how relevant dimension d is to the data factors that a user knows about and cares for, respectively. Specifically, $R(d, U)$ is a weighted sum of the two measurements:

Formula 4. $R(d, U) = w_1 \times K(d, U) + w_2 \times I(d, U)$, where $K(d, U)$ denotes K-relevance, and $I(d, U)$ denotes I-relevance. Since I-relevance is a more comprehensive metric, we set $w_1 = 0.3$ and $w_2 = 0.7$.

We define a simple K-relevance metric to address two cases. First, if dimension d does not belong to any data factors that a user knows about, then the user knows nothing about d (K-relevance = 0). Otherwise, the user's knowledge of d is the same as her best knowledge of the data factor that d belongs to. For example, style belongs to the house exterior factor. If the user is familiar with exteriors, so is she with style. Thus, we define K-relevance, $K(d, U)$ as:

$K(d, U) = Max[\beta K(U, f_i), \forall i]$, where if $d \in f_i$, $\beta = 1$; otherwise, $\beta = 0$; f_i is the ith data factor that the user knows; $K(U, f_i)$ is a binary value denoting the user's familiarity of f_i.

In contrast, we define a finer-grained I-relevance metric to cover more complex cases. First, if dimension d does not belong to any factors that a user cares for, then the user does not care for d (I-relevance = 0). Second, if d belongs to multiple factors that the user is interested in, d receives a higher I-relevance value. Assume that a user is interested in house exteriors and interiors. Dimension yearBuilt, related to both factors, would gain a higher I-relevance than that of roof, only related to exteriors. Moreover, if d belongs to a user-interested factor that contains fewer dimensions, it acquires a higher I-relevance. Suppose that a user cares for two factors: financial (containing just price and tax) and amenity (with 8 dimensions like sewer and water). In this case, I-relevance of price and tax would be higher than that of any dimensions in amenity. The rationale is to increase the chance for a dimension to be selected, if it belongs to a data factor containing fewer members. Accordingly, RIA makes a balanced decision whenever possible to cover every user-interested factor. Thus, we define I-relevance, $I(d, U)$ as:

$I(d, U) = Avg[\beta I(U, f_j)/N, \forall j]$, where f_j is the jth factor that the user is interested in; $I(U, f_j)$ is a discrete value denoting the user's interest in f_j; β is the same binary value as defined above; N is the total number of dimensions in f_j.

Query relevance. A user query may imply what a user cares for. In Figure 3, the user has specified a price constraint in U3, which may hint that she be interested in financial data, such as price and tax. To promote such content to be included (e.g., price in Figure 3a–b), RIA computes query relevance, how relevant dimension d is to a query. Based on our query representation, dimension d may be related to a query by three aspects: the user task (T), the dimensions being asked (D), or the query constraints (S). Here we make two assumptions to simplify our user relevance metric. First, we consider all T similar, since RIA mainly handles data access and lookup queries. Second, we assume that D is already part of the response content, since D is being explicitly asked for. Thus, Q-relevance, $R(d, Q)$, computes how relevant d is to query constraints S:

Formula 5. $R(d, Q) = Avg[R(d, s_i), \forall i]$, where $si \in S$.

Here $R(d, s_i)$ calculates how relevant d is to constraint s_i by our constraint representation. If d is the constrained dimension in s_i (e.g., siding in constraint4 in Q2 Table 2), then $R(d, s_i) = t$, $t \in (0, 1]$. We set t based on the constraint status: t=1, if s_i is a new constraint (e.g., constraint4 in Q2), otherwise, t is a time decay factor (e.g., constraint1 inherited by Q2). As a result, we favor dimensions that are relevant to newly specified constraints.

If d is not the constrained dimension in s_i, we examine how d may be related to the constrained dimension d_k in s_i. Assuming that everything else is equal, it may be more desirable to convey tax data than roof data in the response to U3 (Figure 3). In this case, tax is preferred because of the price constraint, and both tax (d) and price (d_k) belong to the financial factor. To capture such semantic relevance, $Sem(d, d_k)$, between d and d_k, we define $R(d, s_i)$ to be:

$R(d, s_i) = Sem(d, d_k) = Avg[P_j(d, d_k), \forall j]$, where $P_j(d, d_i)$ is the probability of relating d to d_k by the jth data factor f_j. Specifically, If $d \in f_j$ and $d_k \in f_j$, $P_j(d, d_k) = 1/M$, M is the number of dimensions in f_j. Otherwise, $P_j(d, d_k) = 0$.

History relevance. Discourse history influences what to convey [25, 20]. Here we model how an interaction history impacts content selection when a follow-up query is issued (e.g., U2 follows up U1 in Figure 1). In such cases, RIA promotes new information to address the follow-up query, similar to ranking attributes by utterance history [20]. It also tries to maintain a response continuity by slowing down the decay of important dimensions (e.g., bedrms in Figure 1a–b). In short, our history relevance metric, $R(d, H)$ is:

Formula 6. $R(d, H) = \beta[(K-i)^2/K^2]$.

Here β is a binary value, $\beta = 1$, if the current query is a follow-up query, otherwise $\beta = 0$; K is the total number of dimensions presented in the *previous* query; i is an integer between $[0, K]$. If d was not presented before, $i = 0$. Otherwise, we compute i based on d's desirability. For example, the rank of bedrms in Figure 1(a) is 1, as it obtains the highest desirability score in this case.

Media relevance. When selecting data dimensions, we prefer those suitable for presentation in given media. Specifically, we use media relevance, $R(d, M)$, to assess how effectively dimension d can be conveyed in given media M:

Formula 7. $R(d, M) = Max[E(d, m_j), \forall j]$, where medium $m_j \in M$, $E(d, m_j)$ computes the effectiveness of using m_j to convey d. Function $Max()$ ensures that dimension d be conveyed by the most effective media.

We further define media effectiveness using two features, media suitability $S(d, m_j)$, and capability $C(d, m_j)$:

$E(d, m_j) = S(d, m_j) \times C(d, m_j)$.

Feature media-suitability assesses the degree of fitness of using medium m_j to express d. For example, text is the best medium to convey numerical dimensions such as price and size, speech is only fair (due to temporal transience), and graphics (i.e., iconic encoding) is the least effective. We model media-suitability based on media properties and data semantic categories [4, 24] rather than by individual dimensions. Feature media-capability specifies the implemented capabilities of media-specific designers. For example, the underlying visual designer may be incapable of expressing spatial distances, although graphics may be the best medium for depicting such information. Modeling media capability allows RIA to easily support the evolution of media-specific designers. Using the above example, if the visual designer were improved, we only need to update the corresponding graphics capability values.

Using all the relevance metrics defined above, we define the overall relevance metric for dimension d:

Formula 8. $F_3(d) = \sum_i u_i \times R(d, x_i)$, where u_i is the weight, and $R(d, x_i)$ computes how relevant d is to one of the factors x_i by Formulas 3–7. Currently we assign heavier weights to the query and user relevance metrics.

Presentation cost metrics

There is always a cost associated with retrieving and presenting data. Here we consider only the presentation cost, assuming that the retrieval cost is the same for querying any dimension from a database. As RIA uses both spoken and visual media, we measure time and space costs.

Unit time cost. First we measure the *word cost*, the average number of words used for conveying *one instance* (unit) of data dimension d in speech. For example, the average number of spoken words for describing the style of one house is 3. We estimate the word cost of each dimension based on the corpus developed for the speech designer [21]. Since our time budget is measured in seconds, we convert the word cost into a time cost:

$timeCost(d) = s \times wordCost(d)/60$, where s is the TTS (Text-to-Speech) speed, at 160 words per minute.

Unit space cost. Space cost computes the pixels needed to convey one instance of dimension d in text or graphics. For example, the minimal space cost for displaying one house image is 100x100 pixels. We use expert-made presentations to estimate the space cost required to depict a dimension (e.g., counting the minimal number of pixels needed to make a text string or an icon recognizable on a desktop).

So far we have used simple metrics to assess the presentation desirability and cost of data content from a number of aspects (Table 1). Although we have found these metrics adequate for our user tasks, we can always use more sophisticated models. For example, we may measure user relevance using a probabilistic model that can dynamically track user preferences [11]. The crux of our feature-based model is its extensibility. If RIA were to support web-based document retrieval, we could easily incorporate new features, such as data reliability (accessing data quality) and retrieval cost.

5.3 Content Selection Algorithm

As described earlier (Section 5.2), content selection is to fully define the data content (stored in a structure called *attention*) of a conversation act. Our content selection algorithm handles one conversation act at a time (Figure 5). The output of our algorithm is the fully defined data content, including all relevant data dimensions (line 15). If no attention or concept is specified, our algorithm ends (lines 1–2). Otherwise, it decides content in three steps.

```
       ContentSelection (ConversationAct act)

       Interaction Context: Query Q, UserProfile U, Media M
       SpaceBudget S, TimeBudget T
1      if act.attention == null || act.attention.concept == null
2      then return endif
3      Concept concept ← act.attention.concept // a copy of Q.C
4      List requestedD ← act.attention.dimension // a copy of Q.D
       1. Select data dimensions for the main concept
5      List list ← SelectDimension (concept, requestedD)
       2. Select data dimensions for data access queries
6      if Q is data access query then
7        list ← append list SelectDimension(COLLECTION, null)
8      endif
       3. Select data dimensions for complex queries
9      if Q is complex then
10       List cList ← Q.getSecondaryConcepts()
11       for each concept c ∈ cList do
12         list ← append list SelectDimension (c, null)
13       endfor
14     endif
15     act.attention.dimensions ← list
```

Figure 5. Outline of content selection algorithm.

First, it selects data dimensions for the main concept being queried (line 5). For example, the main concept in Q1, Q5, Q6 (Table 1) is house, city, and hospital, respectively. The dimensions being explicitly requested (e.g., price in U5 in Figure 4) are also passed along (line 5). Second, it chooses data dimensions for summarization purpose if the current query is a data access query. For example, RIA provides the count of retrieved houses (Figure 3). To handle summary situations alike, we introduce a concept, COLLECTION, which has dimensions, such as count and range. Third, it selects dimensions for other concepts being queried, if the current query is a complex query (lines 9–14). A *complex query* is a query relating multiple concepts. In Figure 1, U1 links houses (main concept) to hospitals. To provide a coherent presentation, RIA attempts to convey information of all related concepts. For example, RIA provides the hospital name and location in addition to houses (Figure 1a). We address this situation last to ensure that there is sufficient budget for conveying the main concept first.

Dimension Selection

In all three steps, the same routine *SelectDimension*() is called with different parameters. The objective of this process is to find a subset of data dimensions such that their overall desirability is maximized and the total cost is within given space and time budgets. Since it is an optimization problem, similar to the 0-1 knapsack problem [10], we use a greedy algorithm[3] to approximate the process in two steps (Figure 6). First, our algorithm ranks all dimensions by their total rewards (lines 3–9). Routine *reward*() (line 7) computes $r(d)$, a weighted sum of d's desirability scores by its content quality (F_1), quantity (F_2), and relevance (F_3):

Formula 9. $r(d) = w_1 \times F_1(d) + w_2 \times F_2(d) + w_3 \times F_3(d)$.

Here w_1–w_3 are weights. We have tuned the weights through

3 A greedy algorithm works here, since most of our data dimensions have similar cost. Otherwise, we would use dynamic programming [10].

```
     List SelectDimension(Concept concept, List specifiedD)

         1. Rank data dimensions
1        List dimensions $\leftarrow$ concept.dimensions
2        List rankedList $\leftarrow$ emptySet
3        for each dimension $d \in$ dimensions do
4          $r \leftarrow$ reward ($d, Q, U, M$)
5          insert $d$ in rankedList by descending order of $r$
6        endfor
7        if specifiedD != null then
8          rankedList $\leftarrow$ append specifiedD top(rankedList) endif
         2. Pack data dimensions
9        List selectedList $\leftarrow$ emptySet
10       boolean success $\leftarrow$ false
11       for each dimension $d \in$ rankedList do
12         if $d \in$ selectedList || $r(d) < t$ then continue endif
13         $c \leftarrow$ cost (d)
14         success $\leftarrow$ packable ($d, c, S, T$)
15         if success then pack($d$, selectedList) endif
16       endfor
17       return selectedList
```

Figure 6. Outline of dimension selection algorithm.

a series of experiments to set: w_1= .15, w_2= .15, and w_3= .7.

We place the user-requested dimensions (e.g., price in Figure 4) on the top of the ranked list (line 8). To help users focus on what they ask for and to provide them with a coherent view, routine *top*() uses a clustering algorithm to return only a cluster of most desirable dimensions. For example, house locations and city names are chosen in addition to the requested price information (Figure 4).

Based on the ranked list, our algorithm packs as many dimensions as the budget allows. First, it checks whether dimension *d* has already been selected (line 12). It also checks whether the reward is below a certain threshold t (t = 0.35) to avoid selecting undesirable dimensions. If *d* is not packed, our algorithm calculates the cost of *d* (lines 13). Depending on which medium is the most effective for conveying *d* (Formula 7), we compute the corresponding space or time cost. The total cost is the number of retrieved instances multiplying the unit cost of *d*.

Using the total cost computed above, routine *packable*() tests whether there is sufficient budget to accommodate the current candidate dimension (lines 14). If the budget allows, routine *pack*() adds the dimension to the *selectedList* (lines 15). If one type of budget runs out, our algorithm would examine whether a different medium could present the dimension equally effectively. After a dimension is packed, the available budget is reduced accordingly. The packing stops when all dimensions have been considered.

Handling inter-related dimensions

So far our algorithm assumes that data dimensions are independent of each other. This is not always the case. For example, dimensions bedrms and bathrms always appear together based on our analysis of real-estate websites. To handle cases like this, our algorithm must consider the relationships among dimensions. In particular, we model *dimension dependency*, which states that if dimension A depends on dimension B, and A is selected to be conveyed, then B must be included. Currently, we define dimension dependencies based on different data relationships. For example, one type of dependency is to group co-related dimensions (e.g., bedrms and bathrms), the other is to gather complementary dimensions to form a coherent presentation (e.g., locating a house using city name and boundary) [19].

Based on our notion of dimension dependency, we introduce *group dimension*, which contains a set of dimensions by following a dependency chain. For example, if A depends on B, and B on C, then two group dimensions are formed: group [A, B, C] starting from A and group [B, C] starting from B. During content selection, a group dimension is used to replace the head of the chain. In above example, [A, B, C] and [B, C] replace dimensions A and B, respectively. As a result, a group dimension may appear in content ranking and packing (line 3 and line 11).

To consistently handle a group dimension *g* and an individual dimension *d* alike, we define the reward (line 4) and cost (line 13) of *g*. Replacing *d* with *g* in Formula 9, we compute the reward of *g*. But we need to define each feature value of *g* (e.g., objectiveness or informativeness). In particular, *g*'s value for feature f_i is a function G over f_i of all *g*'s members. We use different G for different features. For example, G is *Max*() for computing the importance of *g*, while G is *Avg*() for measuring the objectiveness of *g*. Likewise, we define the cost of *g* to be the total cost of all its members. To pack a group dimension, there must be enough budget to accommodate *all* members of the group. Using group dimensions, we ensure that all relevant dimensions be selected to produce a coherent view of the requested data.

Handling parasite dimensions

To create meaningful responses, we model a special dimension, called *parasite dimension*, which depends on *at least* one of the other dimensions. In other words, a parasite dimension cannot be presented alone, and it must be conveyed with at least one of the dimensions that it depends on. For example, an identifier like house MLS[4], is considered a parasite dimension. Since MLS conveys little information, it is undesirable for RIA to provide users with only MLSs of requested houses. On the other hand, without MLSs users cannot easily refer to the houses that they are interested in (e.g., "tell me more about MLS234076"). To address these situations, our algorithm treats parasite dimensions specially during the packing process (line 14 in Figure 6). For a parasite dimension, routine *packable*() checks whether at least one dimension that it depends on has been packed. If no such dimension has been packed yet, the parasite dimension is put aside and it will be checked again. Otherwise, the parasite dimension is selected.

Most previous systems handle content selection, organization (e.g., ordering or grouping content into presentable units), and media allocation separately [22, 4]. Since all three processes are interrelated, considering the interactions among them enables us to produce more desirable selection results. In particular, our method selects a cohesive set of content by addressing content ordering (ordering dimensions by their desirability) and grouping issues (e.g., grouping dimensions by their dependencies). In addition, our method ensures that selected content be presentable in given media and presentation budget, which is normally addressed

4 We do not model proper name identifiers, such as city name and school name, as parasite dimensions, since they are considered to provide a shorthand definite descriptions of data entities.

during media allocation.

6. IMPLEMENTATION AND EVALUATION

We have implemented RIA as a multi-threaded, distributed system using both Java and C++. All core components are implemented in Java running under Windows, while the graphics rendering component is developed using OpenGL/ Open Inventor/C++ running on Linux. RIA also uses an IBM DB2 server and an IBM ViaVoice/TTS engine.

We purchased the house listings from the Westchester Board of Realtors, and extracted all GIS data, including the cities, school districts, and landmarks, from the U.S. census database. Our database contains 1800+ houses, 70+ cities and towns, 300+ school districts and schools, and 100+ different landmarks (e.g., hospitals and golf courses). The corresponding data ontology contains more than 20 concepts, and each concept is associated with a number of dimensions, e.g., a house has 40 dimensions and a city has 25.

6.1 Evaluation

Prior to developing RIA, we conducted a Wizard-of-Oz (WOZ) study on house hunting. Based on this study, we have informally tested RIA as a whole involving many users and thousands of queries[5]. So far RIA has successfully handled most of the user situations identified in our WOZ study[6]. Moreover, we devised and conducted a set of experiments to evaluate our content selection approach. In particular, we compared how our approach performs against professional human designers.

Pilot Study

To collect realistic RIA usage scenarios (user query patterns and sequences), we first conducted a pilot study, involving 8 users. We asked each user to use RIA to find houses that meet 7 common criteria that were collected from our WOZ study (e.g., price and location criteria). Before our users started, they watched a 2-minute RIA tutorial without receiving additional training. Due to RIA's current interpretation capability, we limited our users to specify up to 4 criteria in a query. As a result, every user accomplished his/her task by taking an average of 5.6 turns. Eight users produced 8 distinct query sequences, each of which contains a series of queries[7]. After the study, we asked our users to evaluate their overall experience with RIA via a questionnaire. Although users expressed their initial frustrations of not knowing how to express their requests to RIA [8], they all favored how RIA responded to their queries. They commented that they were able to quickly obtain the pertinent information from the presentation without taking extra steps (e.g., asking for more details).

Experiment

We recruited two professional multimedia interface designers, one male and one female, both with house hunting experience. Due to the time and effort involved, we randomly chose 3 out of the 8 recorded sequences for this experiment. We asked each designer to sketch a multimedia response on paper for each query in 2 sequences. We also asked them to mark the data dimensions that were most likely to be presented and rank them based on their desirability. For comparison purposes, we assigned one common sequence to both designers (sequence 1 in Table 3). To avoid any content bias, we provided the designers with a list of data dimensions in an alphabetical order by each concept. Our designers were also informed about the user profiles associated with tested sequences (some of our users entered profile data in the pilot). We ran RIA on all three sequences and logged the selected data content.

	Sequence 1	Sequence 2	Sequence 3
RIA-Designer1	0.056	0.162	n/a
RIA-Designer2	0.161	n/a	0.135
Designers1-2	0.132	n/a	n/a

Table 3. Evaluation results.

Result Analysis

For each query, we computed the distance between two sorted data content lists, one produced by RIA and the other by a human designer. Our distance metric considered both content and ranking similarities. The distance is normalized to lie between [0, 1]: the distance is 0.0 if two sets of selected content are identical, including their ranks; the distance is 1.0 if there is no overlap at all between the two lists. The distance for a query sequence is the average of the distances for all queries in the sequence (rows 1–2 in Table 3). For contrasting purposes, we also compared the results produced by the two designers (row 3 in Table 3).

In general, RIA selected data content similar to that chosen by both designers (Table 3). For sequence 1, RIA and Designer 1 chose almost identical content ranked in a similar order for every query. We also examined the causes of the differences made between RIA and our designers. We summarized two main reasons. First, it was difficult for our designers to accurately estimate the available space budget. For example, a query in sequence 2 resulted in 16 houses. Although we informed Designer 1 about the data count and the available space budget, the designer still selected more dimensions than RIA did (RIA adhered to the budget constraint). This cause also contributed to the differences made between RIA and Designer 2 in sequence 3. Second, different presentation techniques caused differences in content selection. While both RIA and Designer 1 mainly used map-based presentations to convey information, Designer 2 sketched a list-based presentation. As a result, Designer 2 did not select spatial information (e.g., house location) as both RIA and Designer 1 did. Designer 2 also tended to select more content due to his scrollable list presentation. This fact confirms a claim made by Andre and her colleagues [3]: content selection and presentation design should interleave, as these two processes may influence each other.

From a list of factors considered by RIA (Table 1), we asked our designers to rate those that had influenced their decisions the most. Both designers rated the query-relevance and user-relevance the highest, consistent with the current model used by RIA (expressed by weights in Formula 8). It is worth mentioning that both designers spent a consid-

5 We have installed RIA at the IBM Industrial Solution Lab, as part of its live demonstrations for daily customer visits.

6 Most failed situations are due to RIA's interpretation capability.

7 Just for the house concept,7 criteria produce 2^7-1 possible query patterns, not counting these query patterns can be given in any permuted order.

erable amount of time (well over an hour) to finish their tasks (about 10 queries for 2 sequences). We also observed that it is difficult for them to rank and select content items from a long list of candidates. In this experiment, they were dealing with more than 100 dimensions over 5 concepts. These observations help confirm the value of our work in response automation, especially when dealing with large and complex data sets.

7. CONCLUSIONS & ONGOING WORK

To help users to explore large and complex data spaces, we have built RIA, an infrastructure that supports user information seeking via an intelligent multimedia conversation. Given a user query, RIA automatically generates a multimedia response that is tailored to a conversation context, including the specific user interests and the query expressions. In this paper, we focus on the problem of determining the data content of a response. Specifically, we have presented an optimization-based approach to content selection in two main steps. First, we introduce feature-based metrics to systematically model various context-sensitive selection constraints. In particular, these metrics dynamically evaluate the presentation desirability and the presentation cost of data content. Second, we describe a greedy algorithm that selects content such that its overall desirability is maximized and the total cost is within a given presentation budget. We have also designed and conducted a set of experiments to evaluate our approach. Our results show that RIA can handle most of the user situations identified in a Wizard-of-Oz study and selects content similar to that chosen by human designers.

We are working on several areas to improve content selection. To support complex data analysis tasks, we are exploring how to use data mining algorithms to choose content for creating tailored and meaningful responses [23]. For example, we would like to handle queries such as "Tell me about the real-estate market in this area". Due to the computational cost involved in examining a large data set, our current approach examines the retrieval results only at an aggregate level (e.g., the total count) without inspecting the content of each instance. For example, RIA knows little about missing data or erroneous data within specific instances. However, this information may help RIA to create a better response, e.g., skipping the missing data or identifying the erroneous data. We are examining how to determine these situations efficiently so that RIA can better handle different retrieved instances.

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. ACM Press, 2001.

[3] E. Andre and T. Rist. The design of illustrated documents as a planning task. In M. Maybury, editor, *Intelligent Multimedia Interfaces*, chapter 4, pages 94–116. 1993.

[4] Y. Arens, E. Hovy, and S. van Mulken. Structure and rules in automated multimedia presentation planning. In *Proc. IJCAI*, pages 1253–1259, 1993.

[5] M. Bordegoni, G. Faconti, S. Feiner, M. Maybury, T. Rist, R. Ruggieri, P. Trahanias, and M. Wilson. A standard reference model for intelligent multimedia presentation systems. *Computer Standards and Interfaces*, 18(6–7):477–496, December 1997.

[6] G. Carenini and J. Moore. An empirical study of the influence of user tailoring on evaluative argument effectiveness. In *IJCAI '01*, pages 1307–1314, 2001.

[7] A. Cawsey. Planning interactive explanations. *Intl. J. of Man-Machine Studies*, 38:169–199, 1993.

[8] J. Chai, S. Pan, M. Zhou, and K. Houck. Context-based multimodal input understanding in conversation systems. In *Proc. IEEE ICMI '02*, pages 87–92, 2002.

[9] J. Chu-Carroll and S. Carberry. Collaborative response generation in planning dialogues. *Computational Linguistics*, 24(3):355–400, 1998.

[10] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *An Introduction to Algorithms*. MIT Press, 2 edition, 2001.

[11] A. Csinger, K. Booth, and D. Poole. AI meets authoring: User models for intelligent multimedia. *AI Review*, 8(5-6):447–468, 1994.

[12] M. Dalal, S. Feiner, K. McKeown, S. Pan, M. Zhou, T. Hoellerer, J. Shaw, Y. Feng, and J. Fromer. Negotiation for automated generation of temporal multimedia presentations. In *Proc. ACM Multimedia '96*, pages 55–64, 1996.

[13] M. Derthick, J. Kolojejchick, and S. Roth. An interactive visual query environment for exploring data. In *Proc. UIST '97*, pages 189–198, 1997.

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

[15] P. Grice. *Logic and conversation*, pages 41–58. Academic Press, 1975.

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

[17] S. Kerpedjiev, G. Carenini, S. Roth, and J. Moore. Autobrief: a multimedia presentation system for assisting data analysis. *Computer Standards and Interfaces*, 18:583–593, 1997.

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

[19] K. McKeown, S. Feiner, J. Robin, D. Seligmann, and M. Tanenblatt. Generating cross-references for multimedia explanation. In *Proc. AAAI '92*, pages 12–17, 1992.

[20] D. Olsen and J. Peachey. Query-by-critique: spoken language access to large lists. In *Proc. UIST '02*, pages 131–140, 2002.

[21] S. Pan and W. Wen. Designing a speech corpus for instance-based spoken language generation. In *Proc. INLG '02*, pages 49–56, 2002.

[22] S. Roth and W. Hefley. Intelligent multimedia presentation systems: Research and principles. In M. Maybury, editor, *Intelligent Multimedia Interfaces*, chapter 1, pages 13–58. 1993.

[23] S. Sripada, E. Reiter, J. Hunter, and J. Yu. Generating English summaries of time series data using the Gricean maxims. In *Proc. ACM SIGKDD '03*, pages 187–196, 2003.

[24] A. Sutcliffe and P. Faraday. Designing presentation in multimedia interfaces. In *Proc. ACM SIGCHI*, pages 92–98, 1994.

[25] K. Tanaka-Ishii, K. Hasida, and I. Noda. Reactive content selection in the generation of real-time soccer commentary. In *COLING '98*, pages 1282–1288, 1998.

[26] M. Walker, S. Whittaker, A. Stent, P. Maloor, J. Moore, M. Johnston, and G. Vasireddy. Speech-Plans: Generating evaluative responses in spoken dialogue. In *Proc. INLG '02*, pages 73–80, 2002.

[27] C. Wickens. Processing resources in attention. In R. Pararsuraman and D. Davis, editors, *Varieties of Attention*. Academic Press, 1983.

[28] M. Zhou and M. Chen. Automated generation of graphic sketches by examples. In *IJCAI '03*, pages 65–71, 2003.