An End-to-End Conversational Second-Screen Application for TV Program Discovery

Peter Z. Yeh, Deepak Ramachandran, Benjamin Douglas, Adwait Ratnaparkhi, William Jarrold, Ronald Provine, Peter F. Patel-Schneider, Stephen Laverty, Nirvana Tikku, Sean Brown, Jeremy Mendel, Adam Emfield

■ *In this article, we report on a multiphase R&D effort to develop a conversational second-screen application for TV program discovery. Our goal is to share with the community the breadth of artificial intelligence (AI) and natural language (NL) technologies required to develop such an application along with learnings from target end users. We first give an overview of our application from the perspective of the end user. We then present the architecture of our application along with the main AI and NL components, which were developed over multiple phases. The first phase focuses on enabling core functionality such as effectively finding programs matching the user's intent. The second phase focuses on enabling dialogue with the user. Finally, we present two user studies, corresponding to these two phases. The results from both studies demonstrate the effectiveness of our application in the target domain.*

The recent explosion of content (such as movies, TV shows, sports, and others) available on television coupled with an increase use in mobile devices (that is, smartphones and tablets) has created significant interest in second-screen applications from both end users and content providers. Second screen applications are designed to run from mobile devices and to enhance the television viewing experience in numerous ways, one of which is helping end users effectively find and control content on television through spoken natural language (that is, conversational TV program discovery).

Conversational TV program discovery applications have recently become available in the marketplace from select cable/satellite providers. However, these applications are limited. They support a predefined set of utterance types (for example, switch to <channel>, find a <genre>movie, and find a movie with <actor >). Hence, end users must conform to

 ISSN 0738-4602

Utterance Type	Example
Search: Multislot	Action movies with Tom Cruise playing tonight.
Search: High Precision	Find a French movie with a British actor.
Search: Logical Expression	Watch action movies without Tom Cruise or Bruce Willis.
WH-Question	Who directed the Dark Knight? Where was Terminator filmed?
Command	Switch to HBO.

Table 1. Types of Utterances Supported.

these types, and cannot combine them in an ad hoc manner (for example, search by genre, actor, and TV station).

More advanced research prototypes (Liu et al. 2012) do not have these limitations. However, these prototypes focus on a piece of the overall problem (for example, entity recognition), and do not support the full range of features required of an end-to-end system. For example, these prototypes do not support question answering (for example, who is the french actress in the movie the dark knight). They also don't support rich dialogue across multiple turns of interaction with the user or handle expressive utterances involving conjunction, disjunction, and negation (for example, find a movie without Tom Cruise and Nicole Kidman), nor do they handle the complexities of searching and controlling live television.

In this article, we report on a multiphase research and development effort at Nuance Communications to develop an end-to-end conversational second-screen application for television program discovery that addresses these limitations. Our solution integrates the following artificial intelligence (AI) and natural language (NL) technologies in a comprehensive manner: (1) Statistical and linguistic-based natural language understanding technologies (Ratnaparkhi 1996; Maxwell and Kaplan 1993) to construct a rich semantic representation of the end user's utterance. (2) Dialogue technologies (Bohus and Rudnicky 2003; Larsson 1998) to enable multiturn conversations through conversational state tracking and dynamic prompt generation. (3) A large-scale commonsense knowledge-base[1] that serves as the target output of linguistic processing and supports SQL query generation. (4) Techniques from natural language interface to databases (NLIDB) (Popescu, Etzioni, and Kautz 2003) to transform the output of linguistic processing into a SQL query to execute against a commercial electronic program guide (EPG) database, which is updated on a daily basis. (5) NL generation technologies (Gatt and Reiter 2009) to summarize and confirm the outcome of acting on the end user's utterance.

Our goal is to share with the community the breadth of AI and NL technologies (mentioned previously) that are required to develop such an end-to-end system, the considerations involved in integrating these technologies, and the learnings from target end users. We start by giving an overview of the main features of our system. We next describe our system's architecture along with the main AI and NL components of the architecture, developed over multiple phases. We then present two user studies. The first study evaluates the core functionality of our system, that is, its ability to find programs matching the user's intent. We also present an in-depth analysis of the failure cases that surfaced from this study. The second study evaluates the effect of supporting dialogue, that is, allowing the system to carry on a multiturn conversation with the user as she or he searches for content on television. We conclude with efforts to further enhance our application with the eventual goal of making it available to a large user population.

Application Overview

When a user starts the application for the first time, it will prompt the user for his/her zipcode and cable/satellite provider. The application uses this information to limit all results to the user's provider and viewing area. The user is then taken to a start screen with a speech icon that she or he can tap on to begin speaking to the application. Table 1 shows the types of utterances supported by the application.

If the spoken utterance is a search request (for example, watch an action movie tonight or find a movie with Tom Hanks), then the application will display all relevant results ordered by start time (see figure 1). The application will also display a confirmation of these results in the prompt box at the bottom of the screen, along with dynamic prompts such as suggestions for refining the results. The user can scroll through these results, and tap on any one to view additional details such as the program synopsis, cast, ratings, and others.

The user can also tap on the speech icon to continue the conversation, in which case, the application will combine history from previous utterances with the current utterance issued by the user. For example, if the user started with action movies fol-

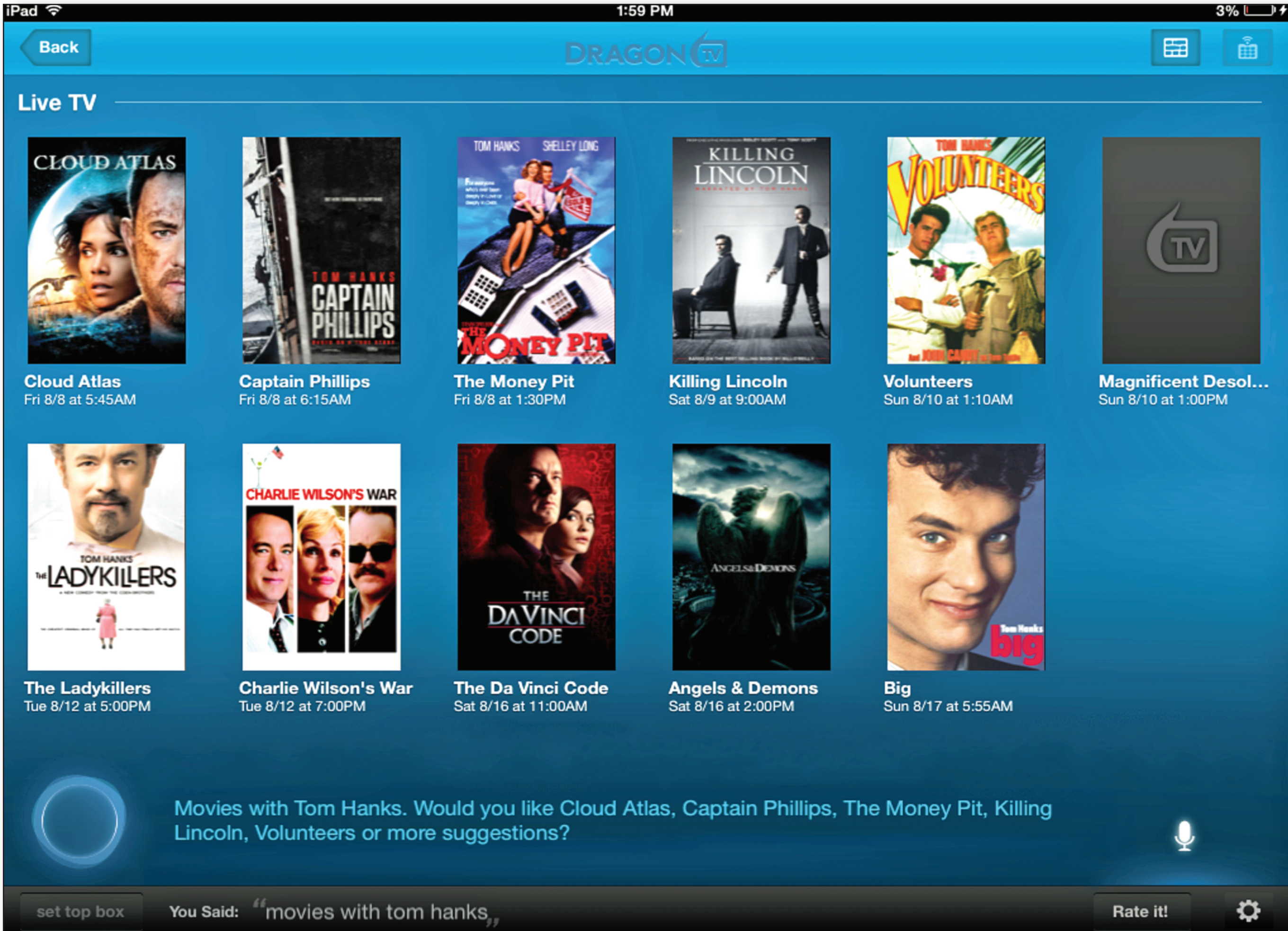

Figure 1. Results Screen.

A text confirmation, along with dynamic prompts to refine the results, is shown at the bottom of the screen.

lowed by something with Tom Hanks, then the application will combine the genre request from the first utterance with the actor request from the second one. Table 2 shows an example dialogue with the application.

If the utterance is a question (for example, where was Tom Cruise born?), the application will display the answer (Syracuse, NY) in the prompt box, along with a prompt directing the user back to the current dialogue. The application will also display any programs relevant to the question such as, for example, any Tom Cruise movies or TV shows that are playing.

If the utterance is a command (for example, change channel, increase volume, and so on), the application will execute the command. For channel change commands, the application will also display the programs that are currently showing on the new channel. The application will prompt the user accordingly for utterances that it does not understand.

Architecture Overview

Our application implements a client-server architecture (see figure 2). The client is responsible for calling Nuance's automatic speech recognition (ASR) service to convert the speech input to text,[2] displaying the results, and controlling the TV.

The server is responsible for the natural language interpretation, retrieval of results, dialogue management, and response generation. We focus on the server in this article, which is implemented as a hub-and-spoke architecture. Each spoke performs a specific task (see table 3 for an overview), and the hub invokes them in the proper order. Hence, the resulting system is highly modular, allowing future spokes to be added with minimal impact to the rest of the system. For example, the blue (darker gray) spokes were developed (and evaluated) first to provide the core functionality of our application, that is, finding programs matching the user's intent. The green

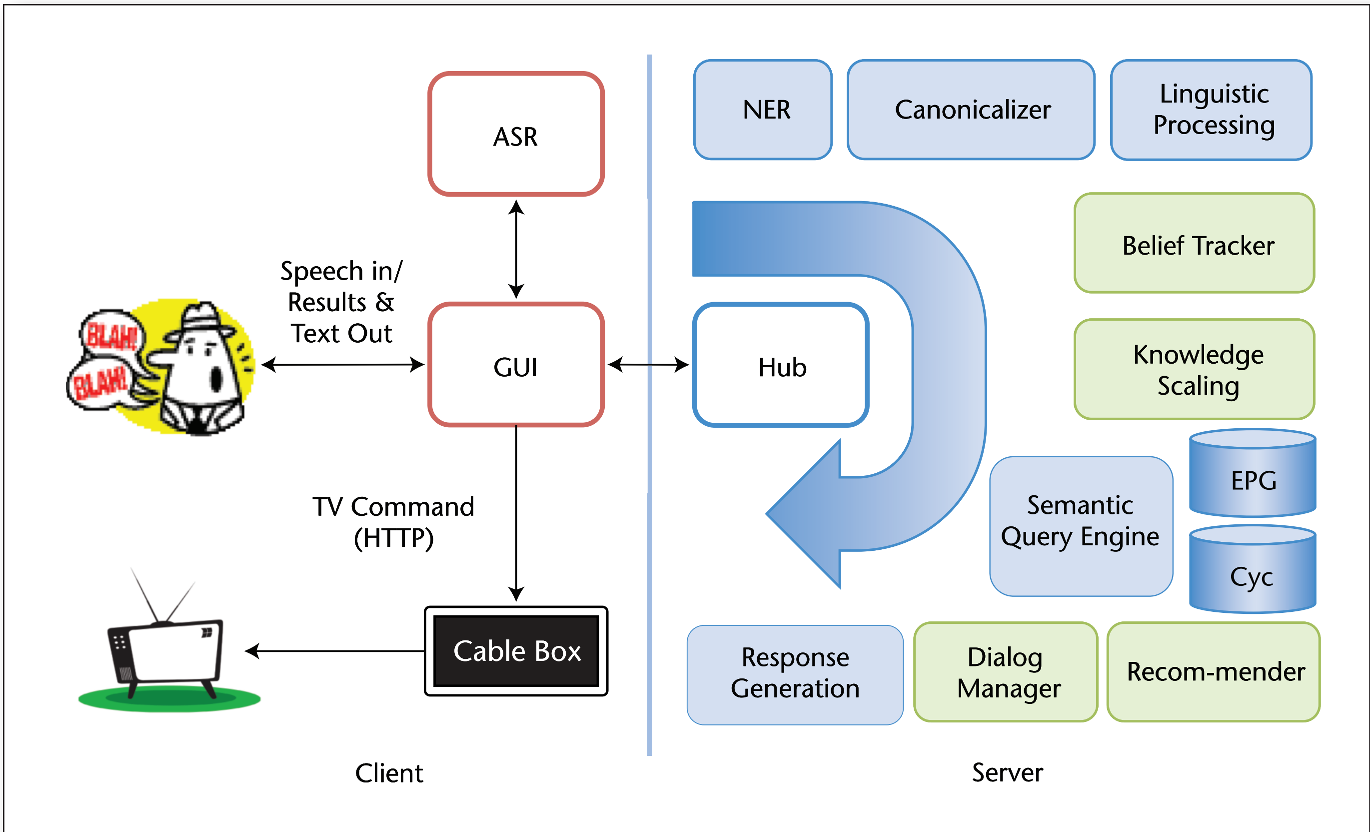

Figure 2. Architecture Overview.

The Hub invokes the spokes in a clock-wise manner starting with NER. The blue spokes were developed (and evaluated) first to provide the core functionality of our application, that is, finding programs matching the user's intent. The green (lighter gray) spokes were added later to enable dialogue and query expansion.

(lighter gray) spokes were added later, primarily to enable dialogue, with minimal impact to the existing spokes.

Ontology and Data Source

Our hub-and-spoke architecture requires a common representation across all the spokes. Moreover, this representation should support challenges that may occur during NL interpretation and SQL query formulation. We believe these requirements can be served by a large multipurpose ontology, and chose ResearchCyc (Cycorp 2013) for this purpose. For example, the named entity recognizer (NER) may have difficulty distinguishing between TV and movie titles. Cyc's rich subsumption hierarchy can provide one concept that subsumes both and can be the target for NER. In particular, the Cyc term VideoConceptualWork includes the desired categories of movie and tv show, and excludes undesirable but related categories such as books and music. Similarly, linguistic processing can produce rich relational structures containing semantic relations grounded in Cyc between the entities detected by NER. Cyc's rich domain and range constraints on these relations can be used during SQL query formulation to further constrain the query.

Our application also requires a continuously up-to-date database of programs playing on TV. We use a third-party commercial electronic program guide (EPG) as our target database. This EPG is a relational database and contains schedule information for all cable and satellite providers in the United States and Canada for the upcoming two-week period. It also contains additional metadata for each program such as the cast, filming location, birth dates of cast members, and others. Moreover, the EPG vendor provides daily updates, which our system downloads and applies on a nightly basis.

Named Entity Recognition

The Named Entity Recognizer (NER) takes the ASR output from the client and detects proper nouns like movie titles and people names. It also detects other phrases that are not proper nouns but have signifi-

User	romantic movies tomorrow
System	Romantic movies tomorrow. Would you like something with Sandra Bullock, Nicole Kidman, Kristen Stewart, or more suggestions?
User	more suggestions
System	How about Robert Pattinson, Kate Winslet, Katherine Heigl, or more suggestions?
User	Winslet
System	Romantic movies with Kate Winslet tomorrow. Would you like *Titanic* or *Little Children?*
User	the first one
System	You chose *Titanic* on Bravo at 10:30 AM on Saturday
User	movies about the Gulf War
System	Movies set during the Gulf War. Would you like *Courage Under Fire, Jarhead,* or *The Finest Hour?*
User	explain the second one
System	This film is an adaptation of the book *Jarhead: A Marine's Chronicle of the Gulf War and Other Battles,* whose subject is the Gulf War. Would you like *Courage under Fire, Jarhead,* or *The Finest Hour?*

Table 2. Example Dialogue with Our Application.

iPad UI results omitted for brevity.

Spoke	Description
NER	Detects entities of interest from utterance
Canonicalizer	Maps entities to DB elements
Linguistic Processing	Produces relational structure from utterance
Belief Tracker	Tracks user's intent across multiple utterances
Knowledge Scaling	Expands entities that do not map to fields in EPG.
Semantic Query Engine	Formulates SQL query to execute against EPG
Dialog Manager	Manages conversation with user
Response Generation	Generates NL responses

Table 3. Overview of Task Performed by Each Spoke.

Input	a tv show with jerry seinfeld playing this weekend
Output	a [TVShow-CW] tv show [/] with [Person] jerry seinfeld [/] playing [CalendarDay] this weekend [/]

Table 4. Example of NER Input and Output.

cance in the TV domain, for example, genres and time phrases. Table 4 shows an example of NER input and output where the tag for each detected entity is grounded in our target ontology.

Our NER is a BIO-style tagger where each word is tagged with *bX, iX,* or *o,* indicating the start of entity *X,* the continuation of entity *X,* or that the word is outside any entity, respectively. The NER is a machine-learned approach and uses the maximum entropy framework to predict BIO tags from annotated data, similar to that described by Borthwick et al. (1998). The model features and search algorithm are borrowed from the part-of-speech tagging approach of Ratnaparkhi (1996), but the original contextual features have been changed to include all consecutive word bi-grams in a window of ±2 words

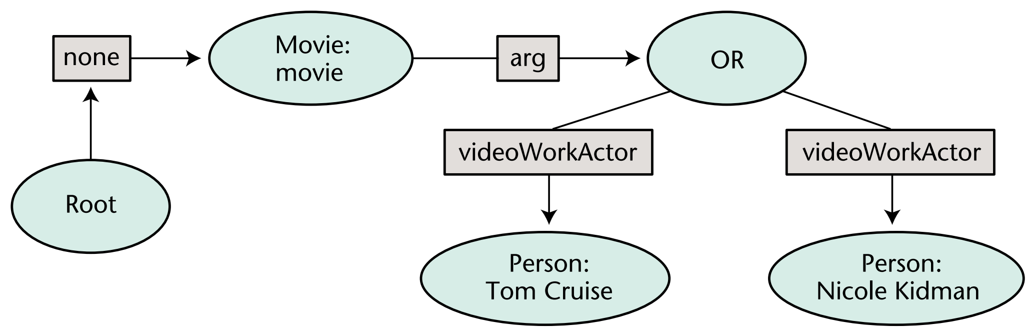

Figure 3. An Example Relation Structure.

This example relation structure is for the utterance "movies with tom cruise or nicole kidman." The root (or main) concept corresponds to the main verb of an utterance or head noun, if the utterance is a noun phrase.

from the current word, and the previous tag, and previous 2 tags conjoined with the current word.

Our NER also uses list match features to flag phrases in the utterance that match those in an externally provided dictionary. We construct this dictionary by extracting approximately 160,000 entries (that is, movie and TV show titles, actor names, and role names) along with their type (that is, movie, actor, and so on) from our third party commercial EPG. Each word in a phrase is assigned a feature if the phrase has an exact match in the dictionary. The features are of the form *bY, iY, eY,* and denote the beginning, middle, and end of a phrase of type *Y.* A word can receive multiple list match features if it participates in multiple matches.

We apply these feature patterns to the training data to create the actual feature set used by the model training algorithm. We use a combination of real and synthetic utterances for training (that is, 19,000 versus 166,000 utterances). The synthetic utterances are necessary because the real ones do not cover all the anticipated linguistic phenomena, and are generated using a combination of manually authored natural language patterns and dictionary derived from our third party EPG.

Canonicalizer

The Canonicalizer takes relevant entities detected by NER and maps them to the corresponding database element based on the surface form in the utterance. This mapping is necessary because of the mismatch between how a user may refer to an entity of interest (for example, movie, actor, and others) and how the entity is encoded in our target EPG. For example, a user may refer to the second terminator movie as terminator two, but the EPG may encode it as *Terminator 2: Judgment Day* (the official title).

We implement our Canonicalizer using the open source search engine Solr because it provides a wide array of fuzzy match options (which are absent from most relational database systems), allowing us to fine-tune the match strategy. Hence, for each relevant entity (for example, TV show, movie, actor, and others) the Canonicalizer performs a fuzzy match lookup of the entity's surface form (that is, the phrase used by the end user to refer to the entity) in the Solr index over the EPG table and attribute corresponding to the entity's type. Each match result is a 3-tuple of the form $< T, A, I >$ where T is the table corresponding to the entity's type, A is the attribute in T containing the unique identifier for the entity, and I is the unique identifier. If there are multiple matches (for example, Avatar referring to both the movie and animated TV show), then the top N, based on popularity, are returned.

These results are associated with their respective entity for use by downstream spokes to further constrain the SQL query during query formulation. Moreover, downstream spokes need only include the identifier (and not the surface form) in the resulting SQL query, which speeds up query execution.

Linguistic Processing

The Linguistic Processing spoke produces rich relational structures, which are necessary to properly handle complex utterances involving disjunction, negation, and complex semantic relations (for example, movies with Tom Hanks versus movies by Tom Hanks or British movie with a French actor). These relational structures are directed acyclic graphs where

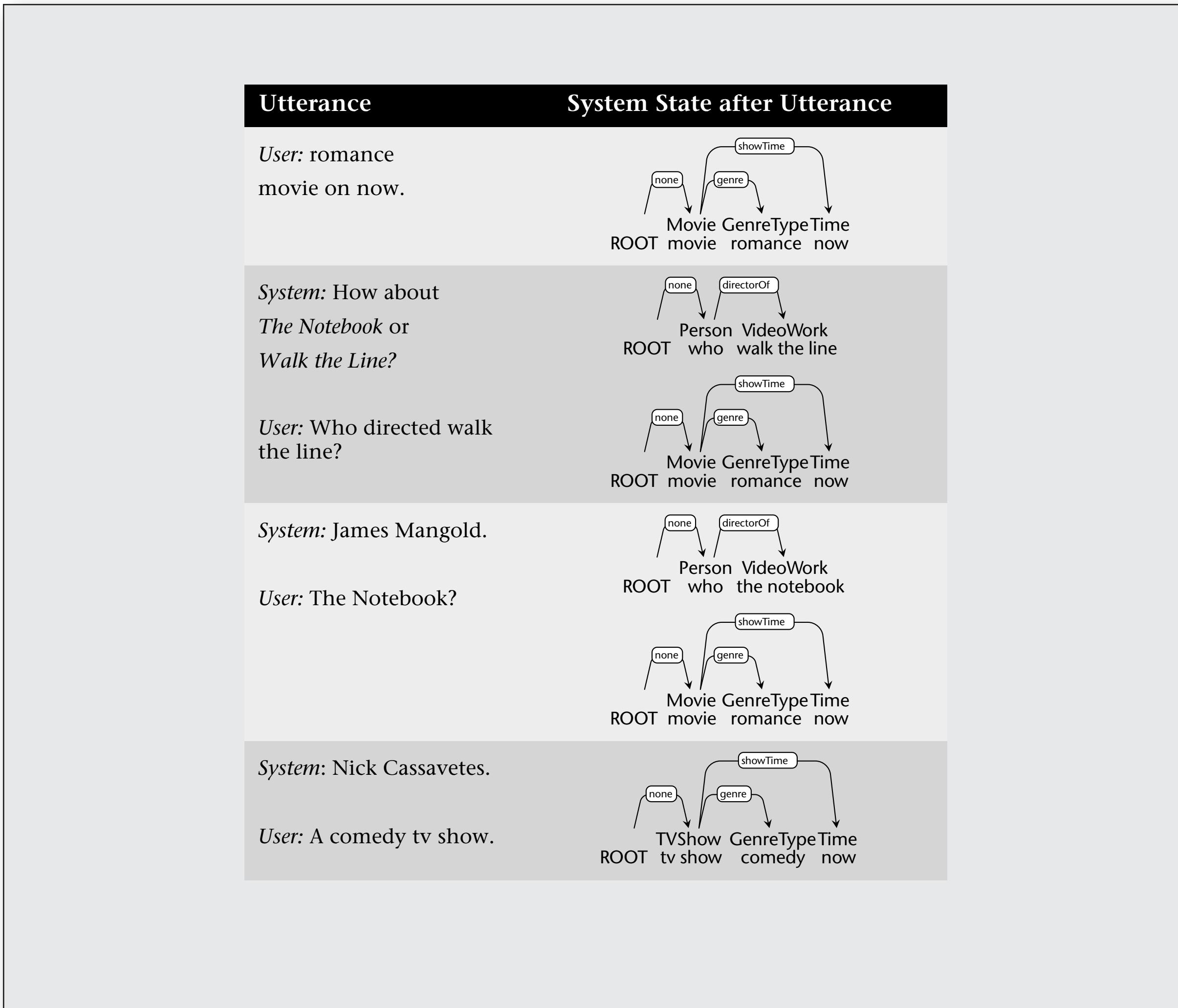

Table 5. State Updates for Each Turn of an Illustrative Dialogue.

Each relational structure in the state represents a separate intent. The order of relational structures in the stack reflects the subdialogue structure of the conversation.

the nodes are concepts and logical operators grounded in our target ontology, and the edges capture semantic relations between the nodes (see figure 3).

This spoke uses the Xerox language environment (XLE) (Maxwell and Kaplan 1993), which incorporates a lexical functional grammar (LFG) parser and an integrated rule system. The LFG parser takes the input utterance and produces a packed representation (Maxwell and Kaplan 1993) that compactly encodes all viable alternative parses of the utterance, for example, encoding alternative prepositional phrase attachments. Moreover, entities detected by NER are used to control the parsing. For example, in "watch tom cruise" if NER tagged "tom cruise" as a person type, then the parser will observe this tag, and not generate alternative parses such as Tom being the subject of a cruise event.

The rule system (Crouch and King 2006) rewrites the parse output into alternative relational structures using three sets of rewrite rules. First, the rule system rewrites the parse structure by adding WordNet (Miller 1995) word senses for each concept term (including NER entities) in the parse.

The rule system then rewrites the resulting structure into alternative abstract knowledge representation (AKR) formulae (Bobrow et al. 2005), which encode the space of possible thematic roles between the concept terms based on the alternative parses from the LFG parser. The formulae also use logical contexts to capture various linguistic notions such as utterance type (for example, question, command, and others), disjunction, negation, and others. We note that this abstract knowledge representation serves as an intermediate representation that allows

$$P_1 : \mathrm{ROOT/none/(Program/)}$$
$$P_2 : \mathrm{ROOT/none/([Movie|TvShow|Game]/)}$$
$$E : \{\}$$
$$T : \{\mathrm{Unify}\ (P_1.@1,\ P_2.@1)\}$$

Figure 4. Example Update Rule for Main Concept Unification.

If the main concept of the previous relational structure is a program and the main concept for the current utterance is a specialization of program then unify these concepts. Relations of the previous concept become relations of the current one.

different ontologies to be supported, hence increasing the modularity of our system.

Finally, the rule system rewrites the formulae into alternative relational structures grounded in our target ontology. WordNet senses for each concept term are mapped to appropriate terms in the ontology. Thematic roles are mapped to predicates (that is, semantic relations), and type-checking rules are applied to ensure terms are compatible with the arguments of the predicates, removing alternatives that are ill-typed. For example, the formula for *play terminator two* has multiple WordNet word senses for *play,* including one for playing a musical instrument and one for playing a movie. The former can be removed because *terminator two* is detected as a movie by the NER, and choosing it triggers a type violation.

The resulting alternative relational structures are scored using a set of simple heuristics that prefer the most common (that is, frequently occurring) interpretation for the TV domain. For example, in *watch a movie with tom cruise on tv* it is unlikely that *tom cruise* will be sitting on the TV, so this alternative is scored lowly (and removed). Should multiple relational structures (and hence unresolved ambiguity) remain, then one is randomly selected as the final result.

Belief Tracker

The Belief Tracker merges the relational structure for the user's current utterance (produced by linguistic processing) with the relational structures from previous utterances to produce a coherent representation of the user's intent (Williams et al. 2013; Yeh, Porter, and Barker 2005). Consider the illustrative dialogue in table 5. The user starts with the intention of finding a romance movie to watch but is then led by the system response into asking a question about one of the search results. The user then modifies the argument of the question to ask about a different movie. Finally, the user returns to the original search intent and modifies the genre. Hence, a model of dialogue state needs a representation that is both dynamic (capable of representing changing intents) and layered (capable of representing multiple intents in a priority order).

To address these representation requirements, the Belief Tracker uses a stack of relational structures (table 5). Each relational structure in the stack represents a single intent (that is, search, query, or command), and its position in the stack defines the order in which the intent will be processed by the system. When the intent on the top of the stack has been resolved, it is popped off and the next intent is processed. This approach is inspired by the notion of question under discussion in the conversational analysis literature (Larsson 1998).

The tracking algorithm is specified by a set of state update rules that apply to pairs of relational structures from the previous state R_2 and the current input R_1. An update rule is a tuple (P_1, P_2, E, T) where P_1 and P_2 are path regular expressions (that is, regular expressions that match against directed paths in a relational structure) applied to R_1 and R_2, respectively, E is a set of constraints on paths matching P_1 and P_2 (for example, equality of two node labels), and T is a sequence of transformation rules to apply to both structures. An example rule is shown in figure 4.

The update rules are applied in sequence to the relational structure on the top of the stack. If no match is found, the relational structure is popped (signifying a change in intent) and the rules are applied to the next structure. This process continues until a matching rule is successfully applied to a relational structure on the stack or the stack becomes empty, in which case the new structure is added on top of the original stack as a new intent. These update rules are currently hand-crafted, but we plan to eventually learn them from data.

Knowledge Scaling

The Knowledge Scaling spoke utilizes large-scale knowledge graphs (in our implementation, Freebase [Bollacker et al. 2008]) to perform inference that expands entities that do not map to fields in a structured database into entities that do. These entities may include occupations (for example, wizards), historical events (for example, the Vietnam War), and mythical creatures (for example, vampires) that have significance in the TV domain, but typically are not encoded in any structured fields of a EPG database. The resulting inference also enables the generation of logically motivated explanations for the results. We give an overview of this spoke in this article, and refer the readers to Yeh and Ratnaparkhi (2014) for additional details.

This spoke examines the merged relational structure produced by the Belief Tracker spoke for entities (for example, Occupation, Event, and others.) and relations (for example, description) that are triggers for query expansion (and hence inference), for example, movies about an occupation of interest such as lawyers. If these triggers are present, this spoke collects all applicable inference paths (learned in a semisupervised manner), and uses them to traverse the knowledge graph. Figure 5 shows an example of an inference path for finding movies about an occupation of interest.

All results (that is, nodes in the knowledge graph reached by an inference path), along with instantiations of the inference paths that lead to the results, are returned for use by downstream spokes. For example, the SQE spoke will incorporate these results during SQL query formulation. Similarly, the Response Generation spoke can generate a logically motivated explanation of why each result is being shown by applying predefined templates to the corresponding instantiated path, for example, *The character Elle Woods in* Legally Blonde *is a lawyer.*

The Knowledge Scaling spoke learns the inference paths offline in a semisupervised manner. It takes as input a knowledge graph (that is, Freebase), and a small set of training examples. Each training example is a pair with an instance of an inference topic (for example, *lawyer,* which is an instance of occupation) and an expansion result (for example, *Legally Blonde*). This spoke then learns inference paths through the following steps:

Hypotheses Generation. An initial set of instance-level paths — we'll call hypotheses — to the expansion results, whose instances and edges have the strongest association.

Hypotheses Activation. Spreading activation (Collins and Loftus 1975) is performed to gather support for the initial hypotheses and to discover additional, variant hypotheses, that is, additional, variant instance-level paths. Spreading activation is appropriate for this step because it provides an effective method to search a network for nodes (in our case hypotheses) that are similar to the source.

Inference Path Generation and Selection. The instance-level paths are generalize into inference paths by replacing instances with their types and collapsing resulting generalizations that are identical. The resulting generalizations are used online.

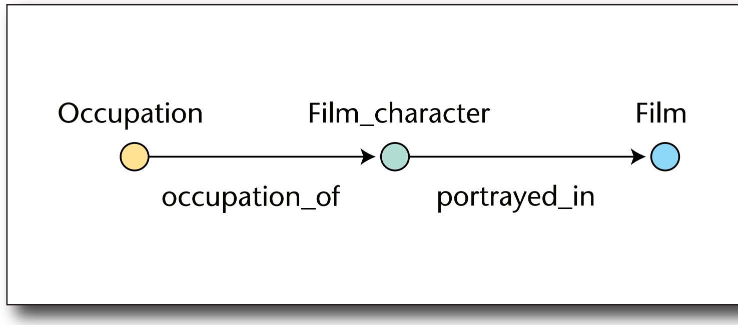

Figure 5. An Example Inference Path.

The edges encode which relations to traverse for a particular inference (in this case movies about an occupation). The nodes encode constraints on the types of entities/instances in the knowledge graph that can instantiate the path subject to the relations specified.

Semantic Query Engine

The Semantic Query Engine (SQE) takes the merged relational structure from the Belief Tracker spoke, and maps it to a SQL query. There are two approaches to this problem: (1) learn the mappings from an utterance to a target query (Zelle and Mooney 1996; Kate, Wong, and Mooney 2005); or (2) compose a query from manually defined mappings between linguistic and database elements (Popescu, Etzioni, and Kautz 2003). We adopt the latter approach because it does not require training examples, which are difficult to acquire at scale for this task.

SQE first tries to specialize each entity's type based on semantic relations between them in the relational structure. This step compensates for fine-grained types that may be difficult for NER to detect. For example, given the utterance *movies with tom cruise,* NER tags *tom cruise* as a person type, and linguistic processing relates *tom cruise* to *movies* through a videoWorkActor relation. Hence, SQE can retrieve the domain (and range) constraints of videoWorkActor from the underlying ontology. If this type constraint (that is, Actor) is a subclass of the original type (that is, Person), then SQE can specialize it.

Second, SQE constructs a query tree (rooted at an and node) by traversing the relational structure in a depth-first manner, starting at the main concept. Each logical operator (that is, and, not, or) and entity traversed is converted into an operator and entity node, respectively. These nodes are attached to the most recent operator node traversed or the root and node if no operator has been traversed yet (see figure 6). For compactness, an and or or node with one child is removed, and its child is attached to its parent node. SQE uses this tree in the next step to generate nested queries and to connect them.

SQE then maps each entity type into a SQL fragment: a 3-tuple of the form $< T, A, C >$ where T is the database table to include in the from clause of the query, A are relevant attributes from T to include in the select clause, and C is a set of constraints to include in the where clause. Each constraint is a 3-tuple of the form (A', V, Op) where A' is the constraint attribute from T, V is the constraint value on A', and Op is the constraint operator (for example, equality, membership, and others). We manually define these mappings based on our target EPG database. Canonicalizer results (see above) associated with the entity are also added to C. For example, the tuple for tom cruise (an Actor type) and associated canonical is

< credit, name, {(type, 'Actor', =), (id, 30914, =)} >

Based on these mappings, SQE finds the shortest join path between the tables in each fragment pair

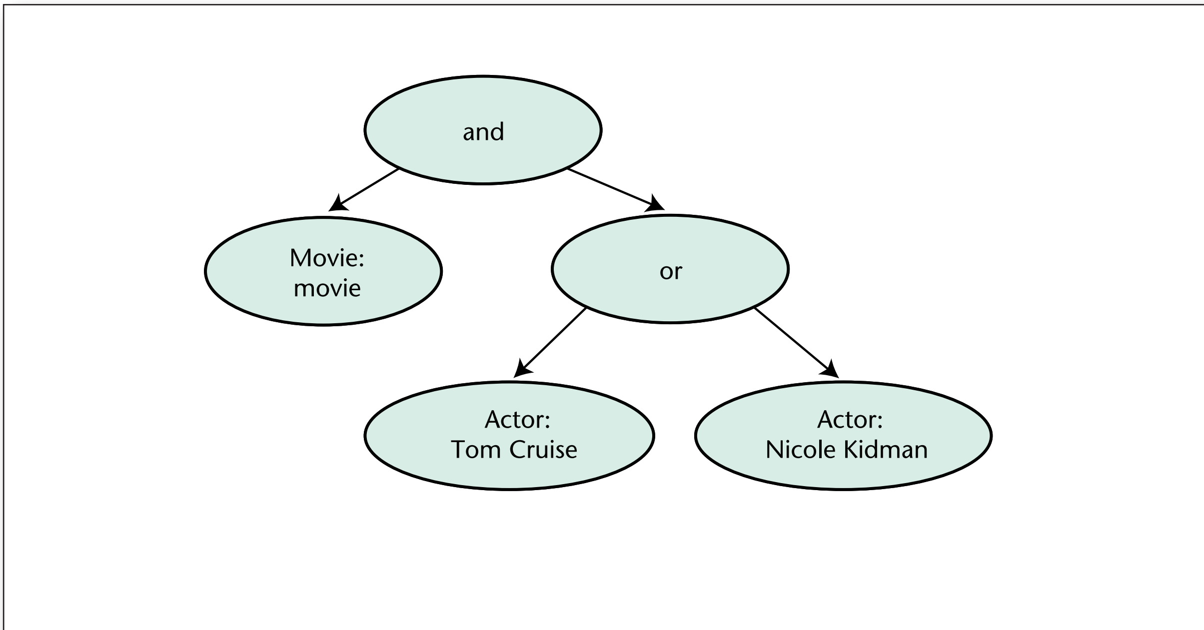

Figure 6. Query Tree Constructed from Relational Structure Show in Figure 3.

through a breadth-first search over possible joins in the database. SQE also observes the structure of the query tree, and greedily merges fragments with overlapping database elements (that is, tables and attributes).

Finally, SQE checks the type of the utterance produced by linguistic processing. If the type is a WH-question, then SQE includes the table and attribute associated with the question type in the from and select clause of the query respectively, and extracts the return value as the answer. This strategy is sufficient because many WH-questions can be answered by applying the appropriate facet over the set of results satisfying the question constraints. The resulting SQL query is executed against the EPG.

Dialog Manager

We use a Nuance proprietary system similar to the Ravenclaw architecture (Bohus and Rudnicky 2003) as our Dialog Manager (DM). The DM has two main strategies for pursuing a conversation: concept refinement and list refinement. Concept refinement solicits from the user additional entities and relations to constrain his or her search. An entropy-based statistical measure and popularity metrics for programs are used to decide which concept to refine, along with relevant suggestions for its value. List refinement is a strategy applied once the number of results is small, to either select or remove elements directly from the list until the user has a single selection. In all cases the system maintains a mixed-initiative paradigm, so the user can provide any input at any point in the dialogue. For example, if the system selected a concept refinement strategy to prune the results, the user could either select one of the system's suggestions or override the refinement with additional commands, questions, or entirely new concepts and values. Subdialogue strategies were implemented for question answering, device control, and explanations (that is, describing the system's reasoning for the results displayed).

The Dialog Manager outputs its communicative intent to the Response Generation spoke using a taxonomy of 26 parameterized speech acts such as REQUEST.ASK for concept refinement and INFORM.REPORT for notifications of system actions.

Natural Language Response Generation

The Natural Language Generation (NLG) spoke generates responses across four categories:

Confirmation Prompts. A restatement of the constraints requested by the user. With noisy ASR and NER, confirmations let the user know whether his/her request was understood correctly. In cases where no results are found, the system will also indicate this.

Dynamic Prompts. Dynamically generated prompts for concept refinement, explanations, and others. The generation of these prompts is driven by parameterized speech acts produced by the Dialog Manager.

Answers. Presentation of possible answers found for WH-questions posed by the user. Additional processing, such as converting the time represented in the EPG to local time, is performed based on the question type.

Exception Responses. Responses to inform the user of exception conditions, for example, NER did not detect any entities, no answers were found for a question, and others.

This component uses templates, the SimpleNLG package of Gatt and Reiter (2009), and transformation heuristics to generate concise prompts. SimpleNLG allows us to more easily enforce common grammatical constraints such as number and noun-verb and article-noun agreement. We predefine a set of SimpleNLG syntax tree templates, and our system selects the appropriate one based on the speech act and its parameters (that is, slot-value tuples) produced by the Dialog Manager. The selected template is instantiated appropriately, and relevant transformations (for example, suppressing portions of the template) are applied based on the context (for example, number of results, result type, and so on).

For example, if the NLG component is asked to generate a confirmation prompt for the speech act CONFIRM.IMPLICIT with slot-value tuples genre = "romantic comedy" and type = "movie or tv show," it will suppress the type slot (if the result includes both movies and TV shows) to generate the concise response "romantic comedies," whereas a pure template-based approach will generate the more verbose response "romantic comedy movies or tv shows." This strategy allows our system to better handle variation, brevity, and fluency of natural English.

Evaluation 1 — Core System

We present the first of two user studies with Nuance's Usability and Interaction Design group. The goal of this first study is to assess the core functionality of our application, that is, finding programs matching the user's intent. The core system evaluated in this study includes only the blue (darker gray) spokes in figure 2.

Experiment Design

We used the following experiment design to answer three key questions: (1) How satisfied is the user with the application? (2) How effective is the application at finding programs matching the user's intent? (3) What is the response time of the application?

We sourced 16 subjects from a third-party staffing agency for this study. These subjects represent target users of our application: users between the ages of 18 and 55, equal mix of male and female users and technical and nontechnical users. For each subject, a moderator first gave the subject a high-level overview of the application (installed on an iPad mini) and the experiment environment — that is, a simulated living room with a TV that can be controlled by our application (see figure 7).

The moderator then gave the subject instructions for a practice trial. The subject was informed of a stack of magazines in the living room, and asked to relax as if he or she is at home. While the subject is relaxing, she or he was told to flip through these magazines for inspiration on what to watch on TV. The subject was then told to tell the application what she or he wanted to watch. Based on the results returned, the subject was asked to rate her or his overall satisfaction on a 7-point Likert scale and to assess the effectiveness of the application by scoring the trial as a success or failure. A trial is successful if (1) at least one of the results on the first page matched the subject's intent or (2) the application correctly gave no results when no programs matching the subject's intent are showing on TV. The application also logged the time spent to process the request.

After the practice trial, the moderator instructed the subject to perform 10 additional trials following the same instructions as above. During these trials, the moderator observed the subject in an adjacent room through a one-way mirror, and only interacted with the subject if she or he had any questions (or experienced any technical issues).

This design is unintrusive (putting the subject at ease), and limits the introduction of biases. Subjects were not exposed to example utterances, which may bias their requests. They came up with requests entirely on their own.

Results

A total of 160 trials were completed (10 per subject). Two raters reviewed each trial to determine those that are out of scope — for example, requests in adjacent domains such as music, or unsupported requests such as showing trailers of upcoming movies. A total of 39 trials — where both raters agreed as out of scope – were removed. Five additional trials were removed because the moderator had to intervene due to technical issues, and 13 trials where the subject incorrectly recorded his/her ratings were removed as well. The remaining 103 trials were used to compute the results.

Figure 8 shows the user satisfaction ratings. The average rating is 4.81 on a 7 point scale with a standard deviation of 1.69. This result is encouraging given the in-the-wild nature of the trials — that is, subjects were allowed to pose any request that came to mind to the application. Moreover, this result is statistically significant compared with a random baseline that assumes a uniform expected frequency over the ratings ($p < 0.01$ for the chi-square goodness-of-fit test, $df = 6$).

Table 6 shows the number of successful versus failed trials. Again, these results are encouraging given the in-the-wild nature (and hence difficulty) of the trials. We also found a strong positive correlation between the percentage of successful trials and the average satisfaction rating per subject ($p < 0.005$ for

Figure 7. Picture of the Simulated Living Room Taken Through a One-Way Mirror from the Adjacent Moderator Room.

# Successful	# Failed	Accuracy (%)
64	39	62.14 percent

Table 6. Number of Successful Versus Failed Trials, and Overall Task Accuracy.

(Successful trials over total trials).

the Spearman rank correlation, $\rho = 0.8338$, $n = 16$). This positive correlation supports the validity of the satisfaction ratings.

Finally, the average response time of our application across all trials is 828.41 milliseconds (sd = 1097.77 ms). None of the subjects expressed concerns over the response time during the evaluation, but this is an area that can be improved upon.

Failure Analysis

We performed an analysis of the failed trials to better understand the cause. For each failed trial, we identified the spoke that caused the failure, and categorized the nature of the failure. Table 7 shows the top-five

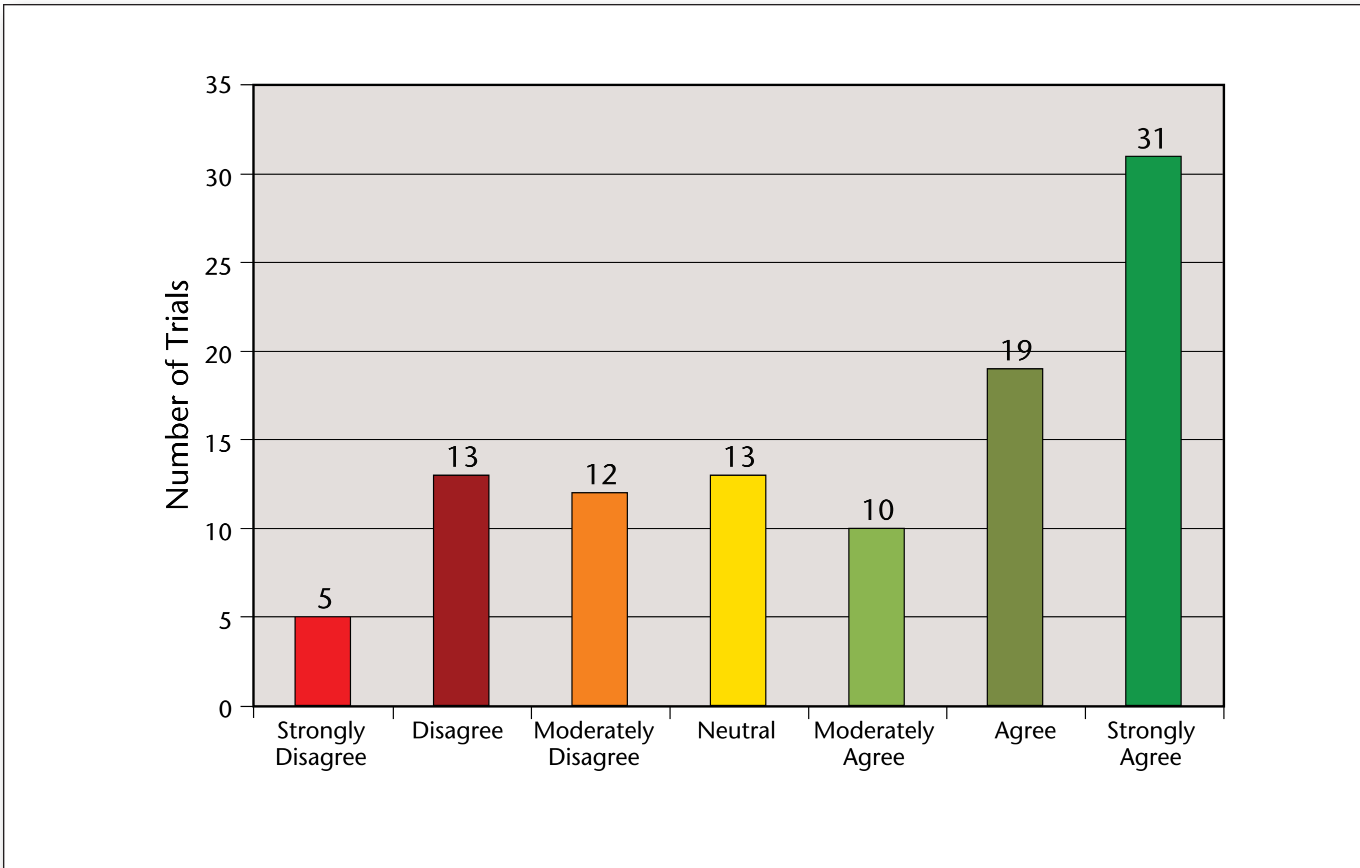

Figure 8. User Rating Distribution over a 7 Point Scale.

Users were rating in response to the question: "The system was able to provide the information I was looking for."

Failure Type	Description	# Trials
Incorrect NER	NER spoke incorrectly detected an entity (that is, wrong phrase boundary or type).	17
Missed NER	NER spoke missed an entity critical to processing an utterance.	11
Incorrect ASR	ASR component produced incorrect text output for speech input.	6
Failed DB Mapping	SQE spoke failed to map an entity required to formulate a correct SQL query.	4
Incorrect relational structure	Linguistic Processing spoke produced a relational structure that prevented the generation of a correct SQL query	3

Table 7. Top Five Failure Types, and Number of Affected Trials.

A trial can have multiple failures.

failure categories. From this analysis, we identified the NER spoke as the top source of failure. Incorrect or missed NER accounted for 28 of 39 failed trials. The primary reason for these failures is that the combination of real and synthetically generated training examples did not fully cover the breadth of user requests and linguistic phenomena that occurred in practice, resulting in an undertrained NER model.

Further analysis confirmed that 24 of these trials would have been successful had the NER performed correctly, increasing the number of successful trials from 64 (62.14 percent) to 88 (85.44 percent). Hence,

System	# Successful	# Failed	Task Success Rate (%)
Dialog	84	14	85.72%
One-Shot	62	36	63.27%

Table 8. Number of Successful Versus Failed Scenarios, and Overall Task Success Rate.

We define the task success rate as the number of successful scenarios (that is, tasks) over the total number of scenarios.

Dialog	One-Shot
84.29 (15.70)	85.71 (15.52)

Table 9. Composite SUS Score Across All Subjects for the Dialog and One-Shot Applications.

Standard deviation is shown in parentheses. The difference between the two versions is not statistically significant.

we are actively investigating ways to improve the performance of this spoke.

Another interesting failure is failed DB mapping, which occurs when a subject refers to a database element at a different level of granularity. For example, a subject may want to watch a show in the home decorating genre, but the EPG only encodes the more generic genre of home and garden. This granularity mismatch may be resolved using logical inferences (for example, subsumption), which we are investigating.

Evaluation 2: Effect of Dialogue

We present a follow-on user study with Nuance's Usability and Interaction Design group to assess the impact of adding dialogue to our application, that is, adding the Belief Tracker and Dialog Manager spokes in figure 2.

Experiment Design

We used the following experiment design to answer three key questions: (1) What is the success rate of an application with dialogue versus one without dialogue in helping users find programs that they want to watch? (2) What is the overall usability of an application with dialogue versus one without? (3) Do users prefer applications with a stateful dialogue model or a one-shot model?

We evaluated two versions of our application:[3] a stateful version (Dialog) where the dialogue components (that is, the Belief Tracker and Dialog Manager spokes, described earlier) were activated, and a repeated one-shot version (One-Shot) where the dialogue components were deactivated. Both versions were run on iPad minis with identical configurations.

We employed the same third-party staffing agency to recruit 14 new subjects from the general public with the same demographics as the first study. These subjects (that is, target users) were between the ages of 18 and 55, with 8 male and 6 female.

For each subject, the moderator counter balanced the selection of which application to start with, and walked the subject through a practice trial to familiarize him/her with the selected application. After the practice trial, the moderator presented the subject with 7 scenarios (that is, tasks) in randomized order. An example scenario is as follows:

> You have young nieces and nephews coming over. Find a program you would like to watch with them.

For each scenario, the moderator asked the subject to imagine him or herself in the scenario, and then speak to the application to find a program that he or she would like to watch. The subject was allowed to continue speaking with the application until he or she either found a suitable program (in which case the scenario was recorded as a success) or gave up (in which case a failure was recorded). Moreover, the moderator did not impose any restrictions on what the subject could say to the application. Hence, the application was exposed to real-world conditions.

After completing all the scenarios, the moderator asked the subject to respond to the system usability scale (SUS) (Brooke 1996), an industry-standard 10-item Likert scale for measuring overall system usability. The moderator then repeated the above protocol with the other version. For each scenario, the moderator asked the subject to start with the same utterance that she or he started with for the first application. The subject could say anything afterwards. This design enabled a more direct measure of the effect of dialogue. After the subject finished both sets of scenarios, the moderator concluded by asking the subject which version of the application (that is, Dialog or One-Shot) he or she preferred, along with reasons for the decision.

Results

Table 8 shows the task success rate for Dialog and One-Shot, that is, the frequency with which a subject found a program that he or she would like to watch for the given scenario. The difference between the two versions was statistically significant ($p < 0.001$ for the chi-square test, $df = 1$). Dialog had a higher success rate because the stateful model allowed subjects to pose more complex requests, that are difficult to formulate as a single utterance, into simpler ones that led to better results. It also had higher success because the concept and list refinement strategies of the DM helped subjects explore more of the space of available programs, which led to better results. The average number of turns for Dialog and One-Shot were 5.38 and 4.81, respectively.

Table 9 shows the composite SUS score across all subjects for each application. The composite score for each system is around the mid-eighties, and correspond to a SUS grade of B or adjective rating of Good verging on Excellent (Bangor, Kortum, and Miller 2009). These results are encouraging, given the open-ended nature of the experiment, that is, subjects can pose any utterance to the application. Moreover, these results suggest that both versions of our application are highly usable for finding relevant content on live television.

Table 10 shows which application version subjects prefer. We observe that 11 of the 14 subjects prefer Dialog, and this difference is statistically significant ($p < 0.001$ for the chi-square goodness-of-fit test, $df = 2$). We note that the preference for Dialog is not due to poor usability of One-Shot. The two versions have comparable usability. Rather, the most common reasons given by subjects include (1) the Dialog version gave useful feedbacks in the form of dynamic prompts indicating dialogue state, refinement suggestions, and others; (2) the Dialog version gave suggestions that helped subjects discover content they had not thought of; and (3) the Dialog version was more natural to interact with. These results further support the positive impact of dialogue in the TV program discovery domain.

	Dialog	One-Shot	No Preference
# of Subjects	11	0	3

Table 10. User Preference.

Conclusion and Follow-On Efforts

In this article, we presented a conversational second-screen application for TV program discovery. Our application has several unique features, such as the use of a modular architecture plus a common ontology across the different components of the architecture, which enable a flexible, extensible system; the use of deep natural language understanding, which supports a wide range of expressive utterances; and support for dialogue (that is, multiturn interactions with the user), which further improves the user's experience in TV program discovery. Our application operates at scale, that is, enabling TV program discovery over "live" data that covers the entire United States and Canada.

We also presented two user studies. The results were encouraging given the in-the-wild nature of both evaluations, and demonstrate the effectiveness of our application in the domain of TV program discovery.

Additional enhancements and investigation are needed before our system can be made available to a large user population. First, we are actively addressing the top failure types from our failure analysis. For example, we could achieve a significant performance lift by improving the robustness of the NER. We are also developing additional enhancements to our application — such as user preference modeling — to further improve the end user experience. Finally, we are actively investigating possible ways in which our system could be embedded within existing mobile virtual assistant capabilities and solutions to support rich user requests (and interactions) in the TV domain.

Acknowledgment

The authors would like to thank Ronald M. Kaplan for his encouragement, guidance, and support on this project. The authors would also like to thank the reviewers for their helpful feedback and suggestions for improving the article.

Notes

1. Cycorp 2013. Research Cyc. (www.cyc.com)

2. A detailed description of Nuance's automatic speech recognition (ASR) is outside the scope of this article. Information on a publicly available, commercial version of the ASR service that we used is available at nuancemobiledeveloper.com.

3. For this study, a machine-learned approach based on the maximum entropy framework was used for linguistic processing. Details omitted due to space limit. The relational structures produced by this approach are equivalent to those produced by XLE.

References

Bangor, A.; Kortum, P.; and Miller, J. 2009. Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale. *Journal of Usability Studies* 4(3): 114–123.

Bobrow, D.; Condoravdi, C.; Crouch, R.; Kaplan, R.; Karttunen, L.; King, T.; de Paiva, V.; and Zaenen, A. 2005. A Basic Logic for Textual Inference. In *Inference for Textual Question Answering: Papers from the 2005 AAAI Workshop*. Technical Report WS-05-05. Palo Alto, CA: AAAI Press.

Bohus, D., and Rudnicky, A. 2003. RavenClaw: Dialog Management Using Hierarchical Task Decomposition and an Expectation Agenda. Paper presented at the 8th European Conference on Speech COmmunication adn Technology. 1–4 September, Geneva, Switzerland.

Bollacker, K.; Evans, C.; Paritosh, P.; Sturge, T.; and Taylor, J. 2008. Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*. New York: Association for Computing Machinery.

Borthwick, A.; Sterling, J.; Agichtein, E.; and Grishman, R.

1998. Exploiting Diverse Knowledge Sources via Maximum Entropy in Named Entity Recognition. In *Proceedings of the Sixth Workshop on Very Large Corpora*. San Francisco: Morgan Kaufmann Publishers.

Brooke, J. 1996. SUS: A Quick and Dirty Usability Scale. In *Usability Evaluation in Industry*, ed. J. Brooke. Milton Park, UK: Taylor and Francis.

Collins, A., and Loftus, E. 1975. A Spreading-Activation Theory of Semantic Processing. *Psychological Review* 82(6): 407–428.

Crouch, R., and King, T. 2006. Semantics via F-Structure Rewriting. Paper presented at the Lexical Functional Grammar Conference, 10–13 July, Konstanz, Germany.

Gatt, A., and Reiter, E. 2009. SimpleNLG: A Realisation Engine for Practical Applications. Paper presented at the 12th European Workshop on Natural Language Generation 30–31 March, Athens, Greece.

Kate, R.; Wong, Y.; and Mooney, R. 2005. Learning to Transform Natural to Formal Languages. In *Proceedings of the 20th National Conference on Artificial Intelligence*. Menlo Park, CA: AAAI Press.

Larsson, S. 1998. Questions Under Discussion and Dialogue Moves. Paper presented at the 13th Twente Workshop on Language Technology / 2nd Workshop on Formal Semantics and Pragmatics of Dialogue. 13–15 May, Enschede, The Netherlands.

Liu, J.; Cyphers, S.; Pasupat, P.; McGraw, I.; and Glass, J. 2012. A Conversational Movie Search System Based on Conditional Random Fields. Paper presented at the 13th Annual Conference of the International Speech Communication Association, Portland Oregon, 9–13 September.

Maxwell, J., and Kaplan, R. 1993. The Interface Between Phrasal and Functional Constraints. *Computational Lingusitics* 19(4): 571–589.

Miller, G. 1995. WordNet: A Lexical Database for English. *Communications of the ACM* 38(11): 39–41.

Popescu, A.; Etzioni, O.; and Kautz, H. 2003. Towards a Theory of Natural Language Interfaces to Databases. In Proceedings of the 2003 International Conference on Intelligent User Interfaces. New York: Association for Computing Machinery.

Ratnaparkhi, A. 1996. A Maximum Entropy Model for Part-of-Speech Tagging. Paper presented at the Conference on Empirical Methods in Natural Language Processing, Philadelpha, Pennsylvania, 17–18 May.

Williams, J.; Raux, A.; Ramachandran, D.; and Black, A. 2013. The Dialog State Tracking Challenge. In *Proceedings of the SIGDIAL 2012 Conference, The 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Stroudsburg, PA: Association for Computational Linguistics.

Yeh, P., and Ratnaparkhi, A. 2014. Mining Large-Scale Knowledge Graphs to Discover Inference Paths for Query Expansion in NLIDB. In *Natural Language Access to Big Data: Papers from the AAAI Fall Symposium*. Technical Report FS-14-06. Palo Alto, CA: AAAI Press.

Yeh, P.; Porter, B.; and Barker, K. 2005. Matching Utterances to Rich Knowledge Structures to Acquire a Model of the Speaker's Goal. In *Proceedings of the 3rd International Conference on Knowledge Capture (K-CAP 2005)*. New York: Association for Computing Machinery.

Zelle, J., and Mooney, R. 1996. Learning to Parse Database Queries Using Inductive Logic Programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference*. Menlo Park, CA: AAAI Press / The MIT Press.

Peter Z. Yeh is a senior principal research scientist at Nuance Communications. His research interests lie at the intersection of semantic technologies, data and web mining, and natural language understanding. Prior to joining Nuance, Yeh was a research lead at Accenture Technology Labs where he was responsible for investigating and applying AI technologies to various enterprise problems ranging from data management to advanced analytics. Yeh is currently working on enhancing interpretation intelligence within intelligent virtual assistants and automatically constructing large-scale knowledge repositories necessary to support such interpretations. Yeh received his Ph.D. in computer science from the University of Texas at Austin.

Deepak Ramachandran is a research scientist at Nuance Communications. His research interests are in reinforcement learning and dialogue management. He obtained his Ph.D. from the University of Illinois at Urbana-Champaign and has previously been a scientist at the Honda Research Institute. He has served on program committees for ICML and IJCAI, was a co-organizer of the Dialog State Tracking Challenge, and has published more than 25 technical publications.

Benjamin Douglas has a background in applied natural language processing (NLP) and search technologies. While at Nuance Communications, Douglas was an NLP software engineer, where he developed the infrastructure necessary to enable advanced conversational systems and requisite technologies such as canonicalization. Douglas currently is a software engineer at Google.

Adwait Ratnaparkhi is a conversational systems architect at Nuance Communications, where he manages a team responsible for end-to-end spoken dialogue prototypes. Prior to joining Nuance, Ratnaparkhi held a variety of research and development positions in the areas of natural language processing, online advertising, machine learning, and conversational systems. He received a B.S.E. from Princeton University in 1992 and a Ph.D. from the University of Pennsylvania in 1998, both in computer science. His past employers include the research laboratories of IBM, Microsoft, and Yahoo. He also has worked for several startups, including 33Across, where he served as the chief scientist.

William Jarrold is a senior scientist at Nuance Communications. His research in intelligent conversational assistants draws upon expertise in ontology, knowledge representation and reasoning, natural language understanding, and statistical NLP. Throughout his career he has developed computational models to augment and understand human cognition. In prior work at the University of California, Davis and the SRI Artificial Intelligence Lab he has applied statistical NLP to the differential diagnosis of neuropsychiatric conditions. At SRI and the University of Texas he developed ontologies for intelligent tutoring (HALO) and cognitive assistants (CALO). Early in his career he worked at MCC and Cycorp developing ontologies to support commonsense reasoning in Cyc — a large general-purpose knowledge-based system. His Ph.D. is from the University of Texas at Austin and his B.S. is from the Massachusetts Institute of Technology.

Ronald Provine is a natural language understanding research engineer at Nuance Communications, where he constructs next-generation dialogue management tools and systems. He has extensive experience in transitioning research results into fielded applications. Prior to joining Nuance, Provine was an advanced computing technologist at Boeing, where he helped deliver Boeing's Simplified English Checker, an expert system for tooling and fixturing planning, an automatic generator of numerical control (NC) programs from CAD models, and a hypertext documentation system (intelligent graphics) for airplane wiring diagrams. He also led a multiuniversity research program on dialogue-based human-robot teaming. His research interests revolve around the integration of natural language processing and reasoning to increase the intelligence of computing solutions. He holds a B.A. in physical sciences from the University of California, Berkeley.

Peter F. Patel-Schneider is currently a senior principal research scientist at Nuance Communications. Patel-Schneider's research interests center on large-scale representation of knowledge and information. He has made long-term contributions to description and ontology logics, particularly the W3C OWL Web Ontology Language. He developed large portions of OWL and its predecessor DAML+OIL, as well as SWRL, the Semantic Web Rule Language, and RDF, the W3C language for representing data in the semantic web. He is currently working on extracting semantic information from data sources and building large-scale knowledge repositories

Stephen Laverty is the technical lead on the Innovation Team at Nuance. Laverty has seen speech technology come of age over his 10 years in the industry. Using his background in signal processing and his industry experience, he pushes the boundaries of speech technology on an ever-expanding set of platforms including mobile phones, smartwatches, televisions, and the devices that make up the internet of things. Laverty received his M.S. in electrical engineering from the Worcester Polytechnic Institute while studying hands-free speech and microphone arrays.

Nirvana Tikku is currently the technical innovation manager on the Innovation Team at Nuance. Over the past few years, Tikku has been at the forefront of building next-generation solutions utilizing speech and natural language technologies for intelligent virtual assistants, wearable devices, and the Internet of Things. Most recently, he has been heavily involved in architecting cutting-edge speech and language tools for developers. Tikku received his B.Sc. in electrical engineering from Tufts University and is a passionate hacker, always looking to find ways to utilize technology for solving everyday problems.

Sean Brown has been, over the last decade, at the forefront of speech technology innovation. At the inception of Nuance's mobile division, Brown was involved in building the foundation for the company's large vocabulary cloud speech recognition systems. He has held product management positions for Nuance's search and dictation services, voice to text, predictive text keyboards, natural language understanding applications such as Dragon Go! and Dragon Mobile Assistant, Dragon Dictation, and Dragon Search. In his current role as director of product innovation, Brown oversees strategic and visionary product and technology programs for Nuance's mobile and consumer division. He holds a B.S. in computer science from Tufts University.

Jeremy Mendel was a senior user experience researcher at Nuance, where he was responsible for testing the usability and user experience of various products and prototypes within Nuance's mobile division. He is currently a user-experience researcher at Google.

Adam Emfield is a senior user experience researcher at Nuance Communications. He is responsible for the usability and user experience testing in the mobile division at Nuance. Emfield's work focuses mainly on testing early-stage prototypes and ideas, typically with speech-enabled interfaces, to improve the user experience of various products, and to provide guidelines to improve future design.