Chapter 9
A Natural Conversation Framework for Conversational UX Design

Robert J. Moore

Abstract With the rise in popularity of chatbot and virtual-agent platforms, from Apple, Amazon, Google, Microsoft, Facebook, IBM and more, a new design discipline is emerging: Conversational UX Design. While it is easy to create natural language interfaces with these platforms, creating an effective and engaging user experience is still a major challenge. Natural language processing (NLP) techniques have given us powerful tools for analyzing bits of language, but they do not tell us how to string those bits together to make a natural conversation. Natural conversation has a sequential organization that is independent of the organization of language itself. At IBM Research-Almaden, we are addressing this user experience (UX) design challenge by applying formal, qualitative models from the field of Conversation Analysis to the design of conversational agents. Our Natural Conversation Framework (NCF) is a design framework for *conversational* user experience. It provides a library of generic conversational UX patterns that are inspired by natural human conversation patterns and that are agnostic to platform and input method (text or voice). This chapter will cover the four components of our Natural Conversation Framework: (1) an interaction model, (2) common activity modules, (3) a navigation method and (4) a set of sequence metrics. In addition, it will briefly outline a general process for designing conversational UX: from mock-up to working prototype.

9.1 Introduction

Today's platforms for creating chatbots and voice interfaces offer powerful tools for recognizing strings of natural language, like English or Spanish or Mandarin, but they leave it to designers to create their own interaction styles. Some styles work like web search where the system does not remember the sequential context across queries, nor recognize user actions other than queries, such as "help" or "thank you." Other natural-language interaction styles work like graphical or mobile user inter-

R. J. Moore (✉)
IBM Research-Almaden, San Jose, CA, USA
e-mail: rjmoore@us.ibm.com

R. J. Moore et al. (eds.), *Studies in Conversational UX Design*, Human–Computer Interaction Series, https://doi.org/10.1007/978-3-319-95579-7_9

faces in which users choose from sets of buttons to submit text-based commands. Conversation is a distinctive form of natural-language use that involves particular methods for taking turns and ordering them into sequences, the persistence of context across turns and characteristic actions for managing the interaction itself. The user experience (UX) designer must model these mechanics of conversation primarily through dialog management and context persistence. Neither natural language processing tools nor conventions for visual user interfaces, such as web or mobile, help designers decide how to string bits on natural language together into naturalistic conversational sequences.

Like natural language, *natural conversation* is a complex system to which whole scientific disciplines are devoted. The mechanics of how humans take turns and sequentially organize conversations are formally studied in the social sciences, especially in the field of Conversation Analysis (CA). To leverage this literature of observational studies, we are applying the concepts and findings from CA to the design of conversational agents. While this kind of approach of applying CA to the design of human-computer interaction has been undertaken before (Luff et al. 1990), both the natural language processing technologies and the field of CA have evolved significantly since then. The proliferation of the NLP technology itself has created a demand for a discipline of conversational UX design, as it has moved out of the research lab and into the wild.

In applying Conversation Analysis to user experience (UX) design, we have developed a Natural Conversation Framework (NCF) for the design of conversational user interaction and experience that is grounded in observational science. By "conversational" we mean a natural-language interface that both recognizes common, conversational actions and persists the sequential context of previous turns, across future turns, so the agent can respond appropriately. The NCF provides a library of generic conversational UX patterns that are independent of any particular technology platform. The patterns are inspired by natural human conversation patterns documented in the Conversation Analysis literature, for example, those of turn-taking or sequence organization (Sacks et al. 1974; Schegloff 2007). The NCF so far has been implemented on both the IBM Watson Assistant and Dialog services. But in principle it can be built on other platforms as well. These implementations provide a starting point for designers and builders so they do not have to reinvent the basic mechanics of conversational structure. The framework consists of four parts: (1) an underlying interaction model, (2) a library of reusable conversational UX patterns, (3) a general method for navigating conversational interfaces and (4) a novel set of performance metrics based on the interaction model. This chapter will describe each component of the framework in turn, as well as briefly outline a design process.

9.1.1 Interaction Model

The smallest interactive unit of human conversation, in which more than one person participates, is the *sequence*. Sequences are general patterns, which like tools or devices, can be used and reused in all kinds of different situations and settings, for all kinds of different purposes. Conversation analysts have identified two types of sequences: adjacency pair sequences and storytelling sequences (Schegloff and Sacks 1973; Schegloff 1982, 2007). "Adjacency pair" is a formal term for a class of recognizable social action pairs, such as, *greeting-greeting*, *farewell-farewell*, *inquiry-answer*, *offer-accept/reject*, *request-grant/deny*, *invite-accept/decline* and more. When someone does the first part of the pair, it creates an expectation, and an obligation, for someone else to do the second part. While the initiation of a sequence constrains the next speaker's turn, it does not determine it. Sequences are inherently collaborative and are the primary vehicles through which we build up conversations, turn-by-turn, and achieve a wide range of social activities.

When someone initiates an adjacency-pair sequence, the recipient does not always complete it in the next turn, although they may be working toward producing the second part of the pair. In other words, adjacency pair sequences are *expandable* (Schegloff 2007). Expansions are sequences that operate on other sequences. While "base" adjacency pairs, stand on their own, independently, their expansions do not; they are dependent on the base sequence (see Example 1; U refers to the user. A refers to the automated agent).

(1) Minimal Expansion

```
01 U:  can you recommend a Mexican restaurant within
02     walking distance?
03 A:  Mañana's is located at Fourth and Winchester.
04 U:  thanks!
05 A:  You're welcome! Anything else today?
```

An utterance like "thanks!" (line 4) does not stand on its own. It is inherently responsive to something else, something prior, in this case the agent's answer (line 3). And it does a particular job in this conversation: it closes the prior sequence. Closing a base adjacency pair is also often done with an "okay," "great" or other "sequence-closing third" (Schegloff 2007).

Sequence expansions enable speakers to manage the interaction itself. In addition to *closing* a sequence, they may be used for *screening, eliciting, repeating* or *paraphrasing.* The following excerpt demonstrates each of these expansion types in a single sequence. It is a working example from "Alma," the persona of our implementation of the Natural Conversation Framework on the Watson Assistant service (formerly Watson Conversation; Example 2).

(2) 5 Sequence Expansion Types

01	U:	**do you know about restaurants?**	((screen))
02	A:	I know a few places.	
03	U:	**a few what?**	((repeat))
04	A:	places.	
05	U:	**can you recommend one close by?**	((base))
06	A:	What kind of food do you want?	((elicit))
07	U:	**well, what are my choices?**	((elicit))
08	A:	Around here Mexican, Japanese or Indian.	
09	U:	**Mexican**	
10	A:	Okay. Mañana's is on First and Chester.	((base))
11	U:	**what do you mean?**	((paraphrase))
12	A:	Mañana's is a Mexican restaurant just a	
13		couple of blocks west on Chester Street.	
14	U:	**oh ok, thanks**	((close))
15	A:	You're welcome! Anything else?	

We can see an example of *screening* (line 1) in which the user does a preliminary inquiry into the capabilities of the agent. Such preliminaries, or pre-expansions, check conditions upon which the first part of the base sequence (line 5) depends (Schegloff 2007). If the agent were to respond, "I can look up current and upcoming movies" instead, the user would likely not ask for a restaurant recommendation next (line 5). The canonical preliminary is "are you busy tonight?," which is heard as checking the conditions for a forthcoming invitation or request (Schegloff 2007).

In between the two parts of the base sequence, we see two expansions that do *eliciting* (lines 6–9). First, the agent proposes that it needs an additional detail, a cuisine preference (line 6), as a condition for granting the user's request. Second, as a condition for answering the elicitation of a cuisine preference, the user proposes that he needs to know the cuisine choices (line 7). Most current chatbot and voice platforms specifically support the first kind of elicitation, the agent-initiated one, and call them simply "slots." An "intent," or user action, can have "slots," meaning bits of information required to fulfill the user intent, for example, cuisine preference or distance. If the user does not provide them in the request itself, the agent prompts for them. But this is not the only kind of "slot," or sequence expansion, in natural conversation; it is only an agent-initiated elicitation. Current platforms tend not to provide guidance regarding how to create other types of slots, like user-initiated elicitations, preliminaries or repairs.

The remaining sequence expansions (lines 3 and 11) are examples of what conversation analysts call "repair" (Schegloff et al. 1977). Repairs consist of a redoing of all or part of a previous turn, either by the speaker or a recipient, where that turn poses a trouble in speaking, hearing or understanding, which prevents the conversation from moving forward. In the first case, the user requests a *repeat* of part of the agent's prior response (line 3), namely, the part that came after "a few." This is a partial repeat request technique that elicits a repeat of just the part of the prior turn that the recipient did not hear. In the second case, the user requests a *paraphrase* of all of the agent's prior response (line 11). Repairs of hearing or understanding troubles can come after *any* utterance in a conversation.

The interaction model of the Natural Conversation Framework thus consists of *expandable sequences,* like an accordion. Compact sequences are common, but each sequence can be expanded by either party as needed. In the excerpt above, the whole thing is one sequence (all 15 lines), an expanded sequence. The expansions are parts of the base sequence (lines 5 and 10). The expansion types, *screening, repeating* and *closing*, enable basic coordination in and out of sequences, while the expansion types, *eliciting* and *paraphrasing*, enable the parties to compensate for emergent asymmetries in their knowledge. Taken together, these sequences that operate on other sequences enable *conversation management*. Because they are designed to manage the interaction itself, these actions are unique to conversation and not found in other forms of natural language use, such as web search, business forms or poems. In order to handle the two-way elicitations and repairs inserted into base sequences, the dialog manager must remember what the user's original inquiry or request was. In other words, *persistent sequential context*, through the use of context variables, is a requirement for this interaction model.

Sequence expansions enable conversational systems to adapt to the particular user on a local, turn-by-turn basis. For example, depending on how detailed the user's initial request is, the agent can elicit additional details as needed, rather than accepting only a complete request.

(3) Agent Elicitation

01 U: **can you recommend a restaurant**
02   **within walking distance?**
03 A: What kind of food do you want?
04 U: **Indian**
05 A: Okay. Taj Mahal is located on
06   Third and Union.
07 U: **thanks**
08 A: You're welcome! Anything
09   else I can do for you?

Here the user does not mention the type of cuisine she prefers in her initial request (lines 1–2) so the agent elicits that detail instead (line 3). The user then provides the detail in a separate turn (line 4). This makes the agent flexible and more like a human speaker than a database.

In addition, some users need more help than others. This may be due to differences in their knowledge or just to idiosyncratic confusions local to the conversation. Sequence expansions enable users to get localized help (Example 4).

(4) User Elaboration Request

01 U: **can you recommend a good sushi place?**
02 A: Walking distance or short drive?
03 U: **what do you mean?**
04 A: Do you want someplace close or
05   farther away?

```
06 U:  oh 'walking
07 A:  Shogun Sushi is located on First
08     and Taylor.
09 U:  thank you very much
10 A:  You're welcome! Is there anything
11     else I can do for you?
```

In this case, the user requests a paraphrase (line 3) of the agent's request for a distance preference (line 2). Perhaps it is a question he did not expect or perhaps "walking distance" is not a phrase with which he is familiar. The agent then paraphrases its prior question (lines 4–5), which enables the user to understand and answer it (line 6). Rather than designing every response of the agent in the simplest, elaborated form, which would be long and cumbersome, especially for voice interfaces, sequence expansions enable the agent's initial responses to be concise. This makes the conversation faster and more efficient. Then if a few users encounter trouble responding, understanding or hearing these more streamlined responses, they can expand the sequence as needed. This is how natural human conversation is organized: with a preference for minimization (Chap. 1; Sacks and Schegloff 1979). That is, speakers should try the shortest utterance that they think the recipient can understand first, see if it succeeds and then expand only if necessary.

Natural Conversation Understanding

Support for sequence expansion is critical in conversational UX design because one of the distinctive goals of conversation is *mutual understanding*. Accurate information alone is not enough. If the user or the agent cannot understand what the other said, the conversation has failed. Analyzing the user's utterance with Natural Language Understanding tools (NLC and entity extraction) is critical, but it is only the first step. Mutual understanding can only be determined when the recipient responds in "third position" (Schegloff 1992b). For example, if a user makes an inquiry (first position), the agent answers it (second position) and the user closes the sequence with "thanks!" (third position), then this is an implicit indication of mutual understanding. But if the user says, "what do you mean?" in third position, then the user does not understand the agent's answer. If the user says, "no, I mean X" in third position, then the agent did not understand the user's inquiry. And if the user says, "never mind" in third position, then mutual understanding has failed and the user is giving up. Sequence expansions provide natural indicators of the participants' state of understanding on a turn-by-turn basis. Therefore, mutual understanding cannot be achieved in one turn; it requires dialog and support for sequence expansion. We use the term "natural conversation understanding," then to refer to sequence-expansion and repair features that enable user and agent to achieve mutual understanding.

9.1.2 Common Activities

The goal of conversational interfaces is not only mutual understanding but also *conversational competence* (see also Chap. 3). Can the automated agent respond appropriately to common actions in conversation? Can the agent *do* conversation? The Natural Conversation Framework provides a starting library of UX patterns, which constitute various aspects of conversational competence. Outlining the patterns in this library requires first defining a vocabulary for naming the parts of a conversation.

Conversation analysts call the smallest unit in human conversation the "turn constructional unit (TCU)" (Sacks et al. 1974). Such units may consist of words, phrases, clauses or full sentences, but they constitute units after which the current speaker's turn is hearably complete, and therefore, speaker transition is potentially relevant. Turns in a conversation then consist of at least one TCU, but often more than one. Speakers take *turns* producing utterances and thereby building recognizable *sequences* of talk, such as pairs of actions, or adjacency pairs. For example, when a request is made, whatever the next speaker says will be interpreted for how it might be granting the request or not (Schegloff and Sacks 1973).

While sequences account for much of the sequential organization in conversations, they are also organized at a higher level into *activities*. Activities are series of related sequences that accomplish some larger goal. They include things like conversation opening, instruction giving, teaching, troubleshooting, joke telling, order placing, storytelling and more. At the highest level, conversations consist of multiple activities. The shortest, complete conversations tend to consist of an opening, a closing and at least one additional activity in between. Take the following (invented) example of a complete conversation that consists of only three activities: an opening, 'requesting a ride' and a closing (Table 9.1).

In this example, the opening consists of two sequences, a greeting sequence and welfare check sequence. The 'requesting a ride' activity consists of two sequences, a request sequence and an inquiry sequence, with expansions. And the closing consists of one sequence: a closing sequence with a pre-expansion. This short conversation is comprised of three activities–opening, 'requesting a ride' and closing–and five sequences–greeting, welfare check, request, inquiry and closing (first pair parts in bold and sequence expansions in italics). Understanding the differences among turns or utterances, sequences, activities and conversations is critical for designing and building them.

The Natural Conversation Framework (NCF) provides a library of modules for common activities. They provide sets of conversational UX patterns for a variety of basic social activities and can be configured and adapted to a wide variety of use cases. The activity modules are directly inspired by studies in Conversation Analysis (CA). The set is not exhaustive; more patterns can be mined from the CA literature. The NCF currently includes the following 15 activity modules containing over 70 sub-patterns (Table 9.2). Three modules are described in more detail below.

The NCF consists of common, reusable UX patterns for delivering the main content of the application, as well as patterns for managing the conversation itself. The

Table 9.1 Anatomy of a conversation

Turns	Sequences	Activities
A: Hello	**Greeting**	Opening
B: Hi!	Greeting	
A: How's it going?	**Welfare check**	
B: Good	Welfare report	
A: Hey, are ya going tonight?	***Pre-request***	Requesting a ride
B: Yes	*Answer*	
A: Can I get a ride?	**Request**	
B: Sure!	Grant	
A: What time?	**Inquiry**	
B: Seven o'clock	Answer	
A: Okay.	*Acknowledgment*	
A: Thank you so much!	***Pre-closing***	Closing
B: No problem!	*Pre-closing*	
A: Bye	**Farewell**	
B: Buh bye	Farewell	

Table 9.2 Natural conversation framework modules

Content	Conversation management
1. Inquiry	1. Conversation opening
2. Open-ended request	2. Offer of help
3. Story/Instructions	3. Capabilities
4. Troubleshooting	4. Repair
5. Quiz	5. Privacy
	6. Sequence closing
	7. Sequence abort
	8. Insult and compliment
	9. Conversation closing
	10. Conversation abort

content modules include patterns for users to make inquiries (e.g., U: "am I covered for flu shots?"), for users to make open-ended requests (e.g., U: "I'm planning a vacation with my family. Where should I go?"), for agents to give sets of instructions or tell multi-part stories (e.g., A: "First, sit comfortably and breathe slowly."), for agents to troubleshoot problems (e.g., U: "I've been feeling very anxious lately) and for agents to quiz users (e.g., A: "What is the force that results from two solid surfaces sliding against each other?"). Each of these activities is generic and can be used in a wide variety of scenarios. For example, a health insurance agent might use the Inquiry module to answer users' questions about their health insurance. Or a customer service agent might use the Open-ended Request module to elicit users' photocopier problems and diagnose the causes and then use the Instructions/Storytelling module to guide the user through procedures for fixing them. Or a tutoring agent might use the Instructions/Storytelling module to present material about the subject of physics and then use the Quiz module to test the users' comprehension. The interaction patterns are generic and independent of the content.

While the content modules needed for particular use cases may vary depending on the functions of the agent, the conversation management modules are appropriate for almost any use case that is conversational. Any conversation must be opened (Conversation Opening) and most conversations should be closed (Conversation Closing). If the agent is to be "intelligent," it should be able to talk about what it can do (Capabilities). Agents in service encounters should always offer to help the user, but other types of agents may do this as well (Offer of Help). After any response by the agent, the user may require a repeat, paraphrase, example or definition of a term in that response (Repair). Sequences should be closed if successful (Sequence Closing) or aborted if troubles cannot be repaired (Sequence Abort). And users may insult the agent (Insult and Compliment) or request to speak to a human (Conversation Abort) in many use cases. Below are examples of one conversation management module and two content modules.

Opening Module

The Opening module provides a pattern for the agent to open the conversation (Example 5). It contains several components that naturally occur in human conversations (Schegloff and Sacks 1973). All components after the greeting are optional (indicated by parentheses) and can be configured by setting certain context variables.

(5) Opening (Agent) Pattern

```
01 A:  Greeting,
02     (Direct Address)
03     (Self-identification)
04     (Organization-identification)
05     (First Topic)
06 U:  Response to First Topic
07 A:  (Second Topic)
```

After a *greeting* by the agent, other opening components can be configured, such as direct address using the user's name, self-identification using the agent's name or identification of the agent's organization using its name. In addition slots for a first topic and second topic can be configured with a range of components typical of conversational openings, such as name request, welfare check, offer of help, capability giving, problem request and more. The following are just a few of the many possible configurations (Example 6).

(6) Opening (Agent) Configurations

```
01 A:  Hello.
02 U:  hi
---
```

```
03 A:  Hi. How are you?
04 U:  fine
---
05 A:  Hi there. What's your name?
06 U:  Bob
---
07 A:  Hello, Bob. I'm Alma.
08 U:  Hi, Alma
---
09 A:  Hello. I'm Alma. What's your name?
10 U:  I'm Bob
---
11 A:  Hi there. I'm Alma. What's your name?
12 U:  Dave
13 A:  Hello, Dave! How are you?
---
14 A:  Hello. I'm Alma at IBM Research.
15 U:  This is Elizabeth at Google
---
16 A:  Hello. This is IBM Research. How can I help you?
17 U:  who invented the hard disk?
---
18 A:  Hello. I'm Alma at IBM Research. What's your name?
19 U:  Dave
20 A:  Hi, Dave! How can I help you?
---
21 A:  Hello. I'm Alma at IBM Research. I can answer
22     questions about computer trivia or quiz you.
23 U:  ok, quiz me
```

If the designer sets no configuration variables, the agent simply does a greeting (line 01). But if *agent name* is set, the agent gives its name (e.g., lines 07, 09, 11, 14, 18 and 21) and/or if *user name* is set, the agent acknowledges the user's name (line 07). Similarly, if *agent organization* is set, the agent identifies its organization (lines 14, 16, 18 and 21). In addition to greetings and self-identifications, the designer can set a *first topic* and a *second topic*. Common first topics are welfare checks (line 03), name requests (lines 05, 09, 11 and 18), offers of help (line 16) or statements of capabilities (lines 21–22). These topics can also be set as second topics (lines 13 and 20) for multi-sequence conversation openings.

Quiz Module

While most chatbots and voice agents can handle simple, two-turn sequences, such as question-answer or command-action, few demonstrate the ability to handle the quiz pattern, which requires more than two turns. The Quiz module provides a conversational UX pattern in which the agent asks the *user* questions and evaluates the user's answers. The user can give the correct answer, give an incorrect one or give up, as well as initiate the standard repeat and paraphrase repairs. This quiz pattern can be used for all kinds of question types and content: pedagogical testing, trivia games, riddles, joke telling and more (Example 7).

(7) Quiz Pattern

01 U: Quiz Request
02 A: Question
03 U: Correct Answer
04 A: Confirmation
05 U: Sequence Closer
---
06 A: Quiz Offer
07 U: Acceptance
08 A: Question
09 U: Repair Initiator
10 A: Repair
11 U: Incorrect Answer
12 A: Disconfirmation
13 U: Sequence Abort
14 A: Correct Answer
15 U: Sequence Closer

The Quiz pattern can be initiated by the user (line 01) or the agent (line 06). In either case, the agent then asks a question (lines 02 and 08). The user can then do one several actions: give the correct answer (line 03), give an incorrect answer (line 11), initiate repair on the question, for example, a repeat or paraphrase request (line 09) or the give up (line 13). When the user gives the correct answer (line 03), the agent confirms it (line 04), and when the user gives an incorrect answer, the agent disconfirms it (line 12). The following (Example 8) uses the quiz pattern for computer trivia questions.

(8) Computer Trivia (Alma)

01 U: **ask me a trivia question**
02 A: Who invented the hard disk drive?
03 U: **the what?**
04 A: hard disk drive?
05 U: **Microsoft?**
06 A: That's not it.
07 U: **I give up**
08 A: IBM
09 U: **ok**
10 A: Anything else for you?
11 U: **ask me another one**
12 A: What computer beat human champions at
13 Jeopardy in 2011?
14 U: **Watson!**
15 A: That's right!
16 U: **yay**
17 A: Anything else today?

In the Quiz module, the *agent* initiates an inquiry sequence (line 2), and the user tries to complete it with the correct answer. The user can get the answer correct (line 14), incorrect (line 5) or can abort the sequence (line 7). In addition to computer trivia, the Quiz module is used by Alma to tell question-answer jokes, as we can see in this working script (Example 9).

(9) Joke Telling (Alma)

```
01 U:  tell me a joke
02 A:  What can you hold without ever touching it?
03 U:  stocks?
04 A:  That's not it.
05 U:  a conversation?
06 A:  That's right!
07 U:  tell me another one
08 A:  Who earns a living by driving his customers away?
09 U:  Microsoft?
10 A:  Nope.
11 U:  you tell me
12 A:  A taxi driver.
13 U:  haha
14 A:  Anything else today?
```

In guessing the punchline of a question-answer joke, like answering a trivia question, the user can get it right (line 5), get it wrong (lines 3 and 9) or give up (line 11). This is how a question-answer-joke sequence works. In contrast, most of today's chatbots and virtual agents either give humorous one-liners or attempt to collapse the whole joke sequence into a single response, for example, "Why can't you trust atoms? They make up everything!," which gives away the punchline.

While the social activities are different, trivia game vs. joke telling, the underlying conversational UX patterns are identical. This is one example of how the same conversation module can be reused with different content and for different purposes. It also demonstrates the difference between content and a UX pattern. The aim of the Natural Conversation Framework is to provide a systematic set of UX patterns, not libraries of industry-specific content, such as intents, entities, questions or answers.

Story/Instructions Module

Although the action pair, or "adjacency pair," is the most prevalent type of conversational sequence, it is not the only one (Schegloff 2007). Another common sequence type is the *story*. A story is a longer unit of talk that requires multiple turns to complete. When telling a story, the speaker does not deliver the whole thing at once. The teller breaks the story into parts and waits for feedback from the recipient before continuing. The pattern for agent storytelling can also be used for giving instructions to the user (Example 10), and it can be extended to any number of parts or steps.

(10) Story/Instructions Pattern

```
01 U:  Story/Instruction Request
02 A:  Part/Step 1
03 U:  Continuer/Pause
```

04 A: Part/Step 2
05 U: Continuer/Pause
06 A: Part/Step 3
07 U: Repair Initiator
08 A: Repair
09 U: Continuer/Pause
10 A: Part/Step 4
11 U: Sequence Closer
---
12 A: Story/Instruction Offer
13 U: Acceptance
14 A: Part/Step 1
15 U: Story Complaint
16 A: Stop Offer
17 U: Acceptance

Storytelling or instruction-giving can be initiated by either the user (line 01) or the agent (line 12). The agent then gives the first part of the story or the first instruction in the set (lines 02 and 14) and waits for feedback from the user. The user can initiate repair (line 07) on that part of the telling, for example, a repeat or definition request, if necessary. Or the user can bid the agent to continue, using what conversation analysts call "continuers" (Schegloff 1982): "ok," "all right," "uh huh," "mhmm," etc. (lines 03, 05 and 09). In addition, the next part of the story or next instruction can also be triggered by the end of a short pause, obviating the need to produce the continuer while still giving the user a slot in which to initiate a repair on the prior part or to stop the telling (lines 15–17). The following is a simple example of using the Story/Instructions pattern for giving a set of three instructions (Example 11).

(11) Instructions (Alma)

01 U: **how do I meditate?**
02 A: First, sit comfortably and breathe slowly.
03 U: **ok**
04 A: Next, count each in breath and out breath
05 until you get to ten. Then repeat.
06 U: **what do you mean?**
07 A: As you breathe in, out, in, count one,
08 two, three all the way to ten.
09 U: **ok**
10 A: Finally, continue counting your breaths
11 for ten minutes.
12 U: **thanks**
13 A: You're welcome! Anything else?

After the initial part of the telling, in this case, the first instruction (line 2), the agent stops and waits for the user to produce a continuer (line 3). By waiting for a continuer after each part of the telling, the agent gives the user the opportunity to repair his hearing or understanding of the prior part before continuing on to the next. For example, in response to the second instruction, the user requests a paraphrase

Table 9.3 Basic navigation actions

1. What can you do?	Capability check
2. What did you say?	Repeat
3. What do you mean?	Paraphrase
4. Okay/thanks	Close sequence
5. Never mind	Abort sequence
6. Goodbye	Close conversation

(line 06) instead of bidding the agent to continue. After receiving the paraphrase (lines 07–08), the user produces a continuer (line 09), and the agent completes the set of instructions (lines 10–11). We see then that breaking an extended telling into parts and waiting for a signal to continue enables the recipient to repair hearing or understanding troubles *as the telling unfolds* instead of waiting until the end, at which point it would be more difficult to refer back to the source of trouble, as well as to comprehend the subsequent parts.

The Opening, Quiz and Story modules are just three of the 15 Common Activity modules with over 70 sub-patterns currently in the Natural Conversation Framework; however, our goal is to expand the library of UX patterns as we identify additional ones. Each module captures some aspect of conversational competence.

9.1.3 Conversation Navigation

With any computer interface, users must learn how to navigate the space. In command-line interfaces, users learn to navigate directories through cryptic commands. In graphical interfaces, users learn to drag files on a desktop to folders. In web interfaces, users learn to jump from page to page with URLs and hypertext. And in mobile interfaces, users learn to touch the screen, rotate it and "pinch" the images. But how should users navigate a conversational interface? What are the basic actions that they can always rely on at any point to navigate the conversation space or to get unstuck? Natural human conversation contains devices for its own management, as we see with sequence expansions. We propose a subset of these as basic actions for conversational interface navigation (Table 9.3).

Capability Check

Discovering the capabilities of a conversational agent can be challenging because, unlike a graphical interface, there are often no visual elements to click and explore. So users should always be able to talk to the agent about what it can do. "What can you do?" is perhaps the most general request for a description of the system's scope and functionality. This is somewhat analogous to the global help functionality of a graphical interface. The capability check should give the user enough guidance to use the app or to ask more specific questions about its capabilities, for example, "tell me more about destination recommendations” or “can you help me book a hotel?” or “do you know about restaurants?”

Repeat

In voice interfaces, unlike text interfaces, utterances are transient. Once the agent’s voice response is done it is gone. Therefore, users must be able to elicit repeats of all or part of the agent’s utterances. “What did you say?” is a natural conversational way to request a full repeat of the prior utterance. In voice interfaces, requesting repeats is somewhat analogous to ‘going back’ in visual interfaces. Although repeats are not as crucial in text-based interfaces, with their persistent chat histories, virtual agents appear dumb if they cannot understand a repeat request. The ability to repeat its prior utterance is such a basic feature of conversational competence. The NCF supports other repeat repairs, including partial repeat requests and hearing checks, but the full repeat request is the most general.

Paraphrase

While capability checks provide a kind of global help to the user, paraphrase requests provide local help on a turn-level basis. This is more analogous to tooltips in a graphical user interface, accessible by hovering the pointer over a button or icon, to get help on a particular feature. Similarly, “What do you mean?” elicits an elaboration or upshot of the prior utterance. In general, the agent’s responses should be concise to increase speed and efficiency, but the paraphrase of that response should be written in simpler language, should avoid jargon and include explicit instruction where necessary. As a result, the elaboration will be longer and more cumbersome than the initial response, but will be easier to understand. Conversely, if the agent’s initial response on occasion must be long and complex, then the paraphrase should be shorter and to the point, making the upshot of the prior utterance more clear. Paraphrase repairs enable users can control the level of detail they receive (see also Chap. 4) without slowing down the conversation for all users. The NCF supports other paraphrase repairs, including definition requests, example requests and understanding checks, but the full paraphrase request is the most general.

Close Sequence

Users should be able to close the current sequence when they receive an adequate response and move on to the next sequence. This is somewhat analogous to closing a document in a graphical user interface or a popup window in a web browser. “Okay” or “thanks” are natural conversational ways for the user to signal the completion of the current sequence and invite the agent to move onto any next topics. This creates a slot in the conversation in which the agent can initiate next topics, for example, offering to look up flights after recommending vacation destinations, or checking to see if the user has any last topics, for example, with “Anything else I can do for you?” Using “OK” to close a conversational sequence in third position is different from the typical use of “OK” in second position with a graphical dialog box to acknowledge a system prompt.

Abort Sequence

When users fail to elicit an adequate response from the agent, they should be able to abort the current sequence and move on to the next. “Never mind” in a conversation functions somewhat like escape in other computer interfaces. It enables the user to give up and move on. This is useful when the user has attempted to repair prior utterances, but the agent still fails to understand. Or if the agent is eliciting a series of details for a complex request (e.g., Open-Ended Request module), and upon hearing the required information, the user changes his or her mind, the user should be able to cancel the current activity by aborting the sequence. In addition, sequence aborts (e.g., “never mind,” “nvm,” “forget it”), provide implicit feedback that there is some kind of interactional problem, and they should be tracked.

Close Conversation

As in a human conversation, users should be encouraged to close their interaction with the agent. This is similar to closing an application or logging out of a system. “Goodbye” is the natural way to move to end a conversation. While the agent could respond with an immediate return farewell (“Goodbye!”), instead it should treat the user’s attempt to close the conversation as a “pre-closing” (Schegloff and Sacks 1973). The pre-closing gives the agent the opportunity to bring up a last topic before returning the “goodbye” and ending the session. This last topic, which can be configured in the Conversation Closing modules of the NCF, might be an organization appreciation (e.g., “Thank you for contacting IBM!”), a success check sequence (e.g., “Did you find what you were looking for?”), a name request (e.g., “Can I get your name before you go?”), a promotion (e.g., “One last thing, would you like to hear about our current promotion for new customers?) or any other last topic that the designer may want to insert into the conversation before losing the user. In addition, the NCF also recognizes user utterances that are designed as pre-closings rather than farewells, such as “got to go,” “are we done?” or “anything else?”

After any agent utterance in a conversation, users should be able to do any of the above six actions. At first users may need to be taught that, unlike a search engine or most of today’s chatbots, your conversational agent will recognize these six basic actions and respond appropriately and *usefully*. Because they are based on human conversation, they should already be familiar to the user and natural to perform, as well as easy to remember.

For an example of how these navigation actions might work, see the following exchanges with Alma (Example 12).

(12) Conversation Navigation A

```
01 U:  what can you do?
02 A:  I haven't been trained yet. I can only make small
03     talk, answer or ask some computer trivia questions
04     and tell some jokes.
05 U:  ok who invented the hard disk?
06 A:  The hard disk was invented by IBM in 1956.
07 U:  say again
```

08 A: The hard disk was invented by IBM in 1956.
09 U: **what do you mean?**
10 A: The hard disk was invented in 1956 by engineers
11 at IBM's laboratory in San Jose, California.
12 U: **thanks**
13 A: You're welcome!
14 A: Anything else for you?
15 U: **goodbye**
16 A: Did you find what you were looking for?

In Example 12, the user relies on five of the basic actions to navigate the conversation. At the beginning, he checks the agent's capabilities (lines 1–4) and then does an action within the scope of that response, asking a "computer trivia question" (line 5). In response to the answer, the user then requests a repeat (line 7), followed by an paraphrase request (line 9). The user then closes the trivia-question sequence with an appreciation (line 12) and moves to close the conversation (line 15). Instead of completing the closing sequence, the agent treats it as a pre-closing and brings up a last topic, a success check (line 16). Now contrast the next Example (13).

(13) Conversation Navigation B

01 U: **show me flights**
02 A: What do you mean?
03 U: **show me flights from SFO to JFK**
04 A: I'm sorry, I still don't understand.
05 U: **never mind**
06 A: Whatever you say, friend.
07 A: Want to know what I can do?
08 U: **yes please**
09 A: Okay.
10 A: I haven't been trained yet. I can only make small
11 talk, answer or ask some computer trivia questions
12 and tell some jokes.

Here the user fails to check the agent's capabilities at the beginning of the conversation and instead initiates a flight request (line 1). This time the agent responds with a default paraphrase request (line 2) to which the user offers an elaboration (line 3). This still fails to enable the agent to understand (line 4) so the user aborts the attempted flight request sequence with "never mind" (line 5). In response, the agent acknowledges the abort and offers to describe its capabilities (line 7), which the user accepts (line 8). The agent then attempts to align with the user's expectations by describing its capabilities (lines 10–12).

Our first implementation of the Natural Conversation Framework was for IBM's What's in Theaters web application in 2015. What's in Theaters was built on the Watson Dialog service (the precursor to Watson Assistant) as a simple demonstration of how to integrate the service with other components. But it also demonstrates an early version of our conversation navigation method (Example 14).

(14) What's in Theaters (2015)

U: what can you do?
A: I can look up current and upcoming movies by Genre or
MPAA Rating and show you trailers for them. But I'm
afraid I cannot search by number of stars or by movie
titles or actor and director names at this time.
U: **ok show me current movies**
A: Are you in the mood for a specific genre?
U: **say again**
A: I said... Are you in the mood for a specific genre?
U: **what are my choices?**
A: Action, adventure, animated, comedy, crime, documentary,
drama, family, fantasy, foreign, historical, horror,
music, mystery, romance, science fiction, TV movie,
thriller, war movies and western.
U: **science fiction**
A: Do you prefer a certain movie rating?
U: **what do you mean by movie rating?**
A: The rating is a recommendation by the Motion Picture
Association of America about the suitability of a
movie's content for particular age groups. For example,
G is for general audiences, while R is restricted to
people 17 and older. Do you prefer a certain movie rating?
U: **I don't care**
A: Good choice, Bob! I found 24 results for Current Science
Fiction movies.
((first movie results))
U: **no I mean PG-13**
A: Good choices, Bob! I found 1 results for Current
PG-13-rated Science Fiction movies.
((Justice League))
U: **thanks**
A: Is there anything else I can help you with?
U: **where is AMC Saratoga?**
A: I'm sorry, I don't understand. Try What's playing?
U: **what's the address for AMC Saratoga?**
A: I'm afraid I don't understand. I can't recognize movie
titles at this time.
U: **never mind**
A: Okay. Whatever you say, Bob! Is there anything else
I can help you with?
U: **no**
A: Okay. Did you find what you were looking for, Bob?
U: **yes**
A: Great! Goodbye.
U: **bye**
A: *Say anything to continue.*

In What's in Theaters, we can see demonstrations of all six conversation navigation actions: capability check (line 1), repeat request (line 8), partial paraphrase request (line 17), sequence closing (line 31), sequence aborting (line 38) and conversation closing (line 45). It also supports detail elicitations (line 10), no-answer responses (line 23) and self-corrections (line 27). Although the functional scope of What's in Theaters was intentionally limited, as a proof-of-concept, it nonetheless demonstrates expandable sequences and conversation navigation.

9.1.4 Sequence Metrics

Since our Natural Conversation Framework is based on *expandable sequences*, the base sequences and their expansions can each be measured separately. This enables corresponding metrics for the effectiveness and efficiency of the conversation itself, independently of customer satisfaction. Effectiveness and efficiency are also two-way metrics: they apply to both the automated agent and to the user. The NCF offers the following three metrics.

Number of Sequences is a measure of how many action pairs, or base adjacency pair sequences, were initiated by the user or by the agent. Instead of reporting simply total number of turns, we segment the conversation logs into expandable sequences, that is base sequences with their expansions, if any. This gives a more informative measure of how many social actions the user or the agent attempted than does total number of turns. The number of user-initiated sequences is a good measure of user engagement.

Sequence Completion Rate is the percentage of initiated base sequences that were completed by the recipient: agent or user. In other words, this is a measure of the rate of success across sequences in the conversations, which is closely related to the level of mutual understanding. If agent or user do not understand what the other means or is doing, they cannot complete the sequence initiated by the other. This is critical because *mutual understanding* is one of the distinctive goals of conversation.

Interactional Efficiency is a measure of how much extra work the user or the agent had to do in the conversation. The expandable-sequence interaction model gives us a theoretical minimum number of turns a given conversation could have. That is, if every completed sequence were compact, the theoretical minimum would be double the number of completed base sequences. We can then compare that minimum to the *actual* number of turns in the conversation and get a measure of the rate of sequence expansion.

Taken together, these sequence metrics tell us different things about conversations with our agent, Alma. Using extreme sets of artificial data (Examples 15–17), we can demonstrate how the metrics vary with different kinds of conversations.

(15) Example of Set A

```
01 U:  what computer beat Jeopardy?
02 A:  An IBM computer named Watson.
03 U:  what computer beat a human chess champion?
04 A:  It was named Deep Blue.
```

(16) Example of Set B

```
01 U:  what computer beat a human chess champion?
02 A:  It was named Deep Blue.
03 U:  Deep what?
04 A:  Blue.
05 U:  what do you mean?
06 A:  The IBM computer was named Deep Blue.
07 U:  oh ok
08 A:  Is there anything else I can do for you?
```

(17) Example of Set C

```
01 U:  what's in theaters?
02 A:  What do you mean?
03 U:  what movies are playing in theaters now?
04 A:  I'm sorry, I still don't understand.
```

These examples represent the kinds of sequences in each experimental data set, A, B and C. The data sets are "extreme" in that they only contain one kind of sequence, that is, compact sequences (A), expanded, completed sequences (B) and uncompleted sequences (C). These extreme data sets better enable us to see under which conditions these metrics vary than real data do (Table 9.4).

In data set A, we included only completed sequences initiated by the user and no expansions by the user. The Sequence Completion rate for agent and user combined was 88% and the Interactional Efficiency was 92%. In data set B, we included only completed sequences initiated by the user, but also numerous expansions by the user, such as repeat requests, "what did you say?", paraphrase requests, "what do you mean?", sequence closers, "ok" and "thanks," and more. In the case of these frequent expansions, the combined Sequence Completion rate was still high, 83%, but Interactional Efficiency dropped significantly to 49%. Finally, in data set C, we included only conversations in which none of the substantive sequences initiated by the user were completed. In other words, there was almost a complete lack of understanding by the agent of what the user was saying. The Sequence Completion rate plummeted to 14% and Interactional Efficiency to 9%.

Table 9.4 Sequence metrics for extreme data

Set	Number of sequences	Sequence completion	Interactional efficiency
A	17.4	0.88	0.92
B	10.4	0.83	0.49
C	7.2	0.14	0.09

In short, if both the Sequence Completion and Interactional Efficiency rates are high, the conversations themselves are effective. If they are both very low, the conversations have failed. But if Sequence Completion is high and Interactional Efficiency is moderate, the conversations are successful, but the user or agent is doing additional work to achieve that success. This invites the conversational UX designer to explore the nature of those sequence expansions. If they are eliciting details, the topic of conversation may be inherently complex. For example, buying airline tickets involves a lot of details and decisions. Moderate Interactional Efficiency may be normal for this activity. However, if the expansions are primarily due to understanding repairs, the conversation designer should re-evaluate the terminology that the agent uses and the knowledge that it assumes and determine if the conversation can be redesigned so that it is more comprehensible from the start. With inherently complex topics or activities, expansions may be unavoidable, but at least with repair features, the user and agent can still succeed in the face of understanding troubles. This is the value of a robust conversational repair system.

The Sequence Metrics also enable us to help disentangle user dissatisfaction with the agent itself from dissatisfaction with its message, for example, company policies. If a customer reports dissatisfaction after an interaction with a company's virtual agent, and the Sequence Completion and Interactional Efficiency rates are high for that conversation, then we know that the customer did not experience trouble understanding the agent and vice versa. Rather the dissatisfaction must have come from the message delivered by the agent and not the quality of the conversation itself. In other words, if the user complains and the agent recognizes and responds appropriately to that complaint, then the problem is not in the agent's ability to understand but in the substance of the complaint itself.

How it works

In order to measure the occurrence of base sequences and their expansions in conversation logs, we label both the user's and the agent's actions inside the dialog nodes themselves. We set context variables on each node that contains an intent or response, to indicate if the user inputs and agent outputs associated with that sequence are parts of base sequences or expansions. So for example, a simple greeting exchange would be labelled with the following context data (Example 18).

(18) Labelled Conversation Log

```
01 U:  hi
02 A:  Hello.
---
01  user_input = "hi"
    user_action = "greeting"
    confidence = 1.0
    user_APR = B1PP
02  repeat = "Hello."
    agent_action = "greeting"
    agent_APR = B2PP
```

What the user says, in this case "hi," is captured by a system variable, *input_text*, and set to the variable *user_input*. The user's action is captured by recording the intent name, "greeting," using the system variable, *intents[0].intent*, and setting it to *user_action*. In addition, the confidence level for that intent is captured. The sequential function of the user's utterance is captured by a set of structural codes, *user_APR*, that we have constructed based on the adjacency pair and repair models in Conversation Analysis (Schegloff 2007), for example, "B1PP" stands for the first pair part of a base adjacency pair sequence. On the other hand, what the agent says is captured through the custom variable, *repeat*, which is also used for repeat repairs, and the agent's action is hardcoded when the response is written and captured by *agent_action*. And like the user's utterance, the agent's is assigned a sequential function with the agent APR code, "B2PP," or base second pair part. Once the dialog nodes are labeled as such, the conversation logs label themselves as users interact with the agent!

One limitation of this approach is that when a user's input is unrecognized, that is, does not match any dialog conditions, we do not know what kind of action the user did nor its sequential function. To attempt to compensate for this missing data, we provide a modifier that represents the average percentage of unrecognized inputs that are initiators of base sequences. For example, if 80% of past unrecognized user utterances are base first pair parts (B1PPs), we set this modifier to 0.8. The modifier is then based on prior data in order to provide a correction to the metrics. Unrecognized user utterances, or random samples of them, should be inspected on a regular basis both to set the modifier but also to discover any systematic problems hidden in these unclassified inputs. Furthermore, conversational UX designers and product managers can learn to interpret the sequence metrics to determine if their conversational agent is doing better or worse than it did yesterday.

9.2 Conclusion

Our Natural Conversation Framework for Conversational UX Design contains four components: (1) an interaction model based on expandable sequences, (2) reusable modules for common conversational activities, (3) a conversation navigation method and (4) a set of metrics that measure effectiveness and efficiency in terms of sequence organization (Schegloff 2007). Each component relies on models and patterns documented in the Conversation Analysis literature. So far this design framework has been implemented on two natural language platforms, IBM Watson Dialog (now shuttered) and IBM Watson Assistant (formerly Conversation), but the patterns are abstract enough to apply to any natural language platform.

While much attention has been given to the natural language classification and entity extraction, an equally important component for conversational interfaces is "sequential context" (Schegloff 1992a). The primary context for interpreting utterances produced in conversation are the prior utterances in that conversation. For example, "how about Las Vegas?" is ambiguous in isolation, but following the utterance, "Here are things to do in Los Angeles," it becomes sensible. Speakers rely heavily on previous talk to provide a context for their next utterance, which enables them to take shortcuts. For the conversational UX designer, deciding what aspects of the sequential context to persist and how to represent it through so-called "context variables," is still a challenging design problem. Creating an interaction that works like a natural conversation requires capturing the current topic of the talk, the user's prior question or request, which entities the user has mentioned so far, whether the previous user utterance was recognized or not and more. Therefore *context design* is a critical area within conversational UX design that must be advanced if virtual agents are to handle many common conversation patterns.

In addition to outlining our design framework for conversational UX, we have also attempted to demonstrate a practice for representing the user experience for natural language and conversational systems: by using simple transcripts. Transcripts can represent sequences of utterances, and who is speaking each one, for either text- or voice-based conversations. Transcripts are easy to read, easy to create and easy to share. Designers and stakeholders can quickly iterate on a mocked-up transcript before building any parts of the conversation space. And because transcripts lack any representation of a graphical user interface, they enable the developers and stakeholders to focus on the design of the conversation without getting distracted by visual elements.

Conversation analysts trade in excerpts of detailed transcriptions of naturally occurring human conversation in order to share and demonstrate their analytic discoveries. Conversational UX designers should likewise trade in transcripts in order to demonstrate the form and beauty of their designs. The practice of sharing transcripts will enable conversation designers to share tips and tricks, learn from each other and collaborate: "Here's a design I made!" "Check out this working script!" "How did you do that?" Every discipline needs standard ways of representing its phenomenon in order to progress. Transcripts are the currency of conversation analysts and should be for conversational UX designers too!

References

Luff P, Gilbert N, Frohlich D (eds) (1990) Computers and conversation. Academic Press, London

Sacks H, Schegloff EA (1979) Two preferences in the organization of reference to persons in conversation and their interaction. In: Psathas G (ed) Everyday language: studies in ethnomethodology. Irvington, New York, pp 15–21

Sacks H, Schegloff EA, Jefferson G (1974) A simplest systematics for the organization of turn-taking for conversation. Language 50:696–735

Schegloff EA (1982) Discourse as an interactional achievement: some uses of 'uh huh' and other things that come between sentences. In: Tannen D (ed) Analysing discourse: test and talk. Georgetown University Press, Washington, DC, pp 71–93

Schegloff EA (1992a) In another context. In: Duranti A, Goodwin C (eds) Rethinking context: language as an interactive phenomenon. Cambridge University Press, Cambridge

Schegloff EA (1992b) Repair after next turn: the last structurally provided defense of intersubjectivity in conversation. Am J Sociol 98:1295–1345

Schegloff EA (2007) Sequence organization in interaction: a primer in conversation analysis, vol 1. Cambridge University Press, Cambridge
Schegloff EA, Sacks H (1973) Opening up closings. Semiotica 7:289–327
Schegloff EA, Jefferson G, Sacks H (1977) The preference for self-correction in the organization of repair in conversation. Language 53:361–382