

# Interview and Delivery: Dialogue Strategies for Conversational Recommender Systems

Pontus Wärnestål, Lars Degerstedt and Arne Jönsson

Department of Computer Science  
Linköping University, Sweden  
{ponjo,larde,arnjo}@ida.liu.se

## Abstract

In our work with conversational recommender systems we have derived two dialogue strategies called interview and delivery. We explore the symmetry between preferential interview and traditional clarification questions, and arrive at basic interview and delivery strategies suitable for conversational recommender system implementations. The strategies are based on a corpus analysis of recommendation dialogues in the movie domain. We illustrate the strategies in a conversational music recommender system called CORESONG.

## 1 Introduction

Recommender systems aim at assisting users in searching, sorting, and selecting items from large information repositories. They differ from traditional information search engines in that they provide *personalized* service, and the key issue is to correctly construct, update and utilize individual records of user preferences. We employ a conversational approach to preference modeling and recommender system interaction in order to achieve a collaborative dialogue, which is likely to improve system prediction accuracy and user interaction quality (Carenini et al., 2003). Users can explain preferences in a qualitative way as opposed to only rely on the standard nominal 1-5 rating of items (Burke et al., 1997); and the dialogue structure can be used as a tool for detecting variations in preference strength depending on the user's dialogue act and the conversational circumstance in which it occurs (Carberry et al., 1999).

The successful dialogue systems to date are typically implemented in well-known domains and exhibit standardized dialogue strategies (e.g. flight ticket bookings, hotel reservations, etc.) rather than having a completely free and human-like language interaction (Pieraccini and Huerta, 2005). The goal of this paper is to advance our knowledge on recommendation dialogues. We have studied recommendation dialogues in media domains, such

as music and movies, and present two *dialogue strategies* found in such dialogues.

For the purpose of describing the strategies, we view them as genres (Erickson, 2000), each with a specific purpose, form, and participatory structure. We illustrate the different strategy instances with a conversational music recommender system called CORESONG.

Note that these two strategies cover all possible forms of the studied interaction, including utterances within and outside of the domain as well as e.g. questions where the user asks for help on how to use the system. How to differentiate between these and correctly provide relevant information is a technical issue presented in section 3.

## 2 Recommendation Dialogue Strategies

First we turn to the case to be studied: recommendation dialogue in media domains. For this purpose we have conducted a study where a human-human dialogue corpus study was collected (Johansson, 2003). Twenty-four dialogues were recorded in a living room environment with two participants, one acting as a recommender and one as a customer looking for movie recommendations. The participants did not know each other's movie preferences previously, since one goal was to examine how humans go about assessing preferences from scratch. As information resource the recommender had access to a laptop with a large movie information database with the attributes title, actor, director, genre, plot, and year for over 6.3 million individual film/TV credits. In total, 2,684 utterances (mean 112 utterances per dialogue) were transcribed and analyzed using the dialogue distilling method (Larsson et al., 2000) for systematic rewriting into human-machine dialogue. In the distilled corpus, the recommender is referred to as the "system" and the customer as the "user". The distilled corpus was then examined and by clustering re-occurring patterns in the dialogue two principal *dialogue strategies* emerged: **interview**, and **delivery**. Our hypothesis is that a recommendation dialogue model suitable for implementing

conversational recommender systems can be seen as a combination of interview and delivery strategy instances of varying complexity.

## 2.1 The Interview Strategy

The purpose of the interview is to collect information about opinions on domain entity types (e.g. genres, directors or actors in the movie domain) and specific items (i.e. specific movies). We refer to this information as *preferences*, which form the base for deliveries (e.g. product recommendations, or database search results). There are two roles in the interview: an *interviewer* that asks an *interviewee* to constrain or, in some cases, relax attribute values.

The interviewee is assumed to respond cooperatively, and is also free to volunteer preferences not explicitly asked for. The question-selection strategy (i.e. the order in which entity type and value preferences are requested by the interviewer) follows an order based on the interviewer’s idea of importance in the domain and usually ends in “open” preference requests (e.g. utterance S4a in Figure 1). Our corpus analysis suggests a question-selection strategy that moves from generics to specifics (i.e. asking about genre preferences before asking about actor preferences). Note however, that the default order can be revised since interviewees may volunteer preferences in a different order and inform the interviewer of specific importance of certain attributes. Recommender systems that implement this kind of interviewer strategy are said to have a *dynamic* question-selection strategy (cf. (Bridge, 2002)), since the questions are chosen at run-time depending on what preferences have been given by the interviewee. As the dialogue progresses it may

- S1a Hello. Let’s set up your music preferences.  
 S1b Is there a particular genre you like?  
 U1 I like *techno*  
 S2a These are the artists and albums that belong to the genre *Techno*: [list]  
 S2b Any of these you like in particular?  
 U2 I like the artist *Flexter*  
 S3a These albums are made by the artist *Flexter*: [list]  
 S3b Any of these you like in particular?  
 U4 The first one is good  
 S4a Ok. Do you have any other preferences?  
 U4 What *metal* bands are there?  
 S4a These artists belong to the genre *Metal*: [list]

Figure 1: Preference question selection strategy in the constrain interview. Example from the music domain. Entity values are in *italics*. S = system, U = user.

become impossible to provide more deliveries based on the current preference model. The deliverer then takes on the interview strategy, but tries to relax the constraints. When asking for relaxations, the interviewer uses the inverse order in which attributes were requested in the constrain strategy. Figure 2 shows a sample relaxation interview used in the movie domain. When the preference

- S1a There are no more movies matching the current criteria.  
 S1b Would you like to ignore any director preferences?  
 U2 Yes, skip all directors  
 S2a Ok, I have a recommendation ready for you.  
 S2b I think you will like the movie *The Usual Suspects*.

Figure 2: Relaxing a specific constraint in a preference interview (S1b), which is followed by a delivery (S2b). Example from the movie domain. S = system, U = user.

requests on the interviewer’s agenda have been fulfilled and some *external resource*<sup>1</sup> responsible for reaching a solution, a delivery can be made. This depends on the task and the nature of the resource. In some cases the interviewer has a fixed agenda stating which attributes that need values supplied by the interviewee. The dialogue then progresses with repeated constrain requests in a “slot-filling” fashion. When all slots have been filled a delivery (typically in the form of a database result set) can be made. This strategy is standard for information-providing dialogue systems.

In other cases the requests for constraints are more dynamic. For instance, in recommender systems the number and nature of the “slots” that need to be filled depends on the interviewee’s preferences. For one interviewee (the user) it might be enough for the interviewer (the system) to ask for a handful constraints<sup>2</sup> if her preferences are narrow enough to quickly reach high-quality predictions to be used for recommendations. For another interviewee (e.g. one with “too normal” preferences that does not make her easy to place in a *collaborative filtering neighborhood*) the interviewer might have to keep constraining for several turns until the recommendation engine is ready to provide a recommendation.

<sup>1</sup>Such as a recommender engine, or database back-end resource.

<sup>2</sup>This depends on the nature of the recommendation engine. “Low” might mean a dozen or more items in a collaborative filtering engine; or perhaps one or two genre preferences and a few actor preferences in a content-based movie recommender engine.

## 2.2 The Delivery Strategy

Ensuing an interview is a *delivery*. The goal of a delivery is to present the *result* of the preceding interview. The roles in the delivery strategy are: a *deliverer* and a *receiver*.

In the basic case, a delivery simply consists of presenting a solution to the deliverer in one go. This is called a *direct delivery*. However, a delivery can be more elaborate. For instance, a deliverer could motivate her choice of recommendation with a motivation for it, as well as follow-up questions regarding the quality of the given recommendation. Typically, deliveries such as recommendations are delivered when the system is ready (i.e. has collected enough user preferences); and not necessarily at the turn immediately following a user request for a recommendation. Therefore, we call such deliveries *indirect*. Motivations are central for building trust (Swearingen and Sinha, 2002) and help explaining the inner workings of the recommendation algorithm (Höök, 2000). They are frequently used in recommendation situations and therefore desirable in recommender system interaction. Asking the receiver for feedback on the delivered item is also considered part of the delivery strategy, and could be used in both recommendations and more traditional information delivery, such as in information-providing dialogue systems. The initiative is mainly on the deliverer (i.e. the system in an implementation). Figure 5 exemplifies the resulting dialogue of the two kinds of delivery in the CORESONG system.

Deliveries may consist of *exceptional results*. Exceptional results arise when the user has provided too little, ambiguous, or erroneous information; and needs guidance in order to achieve her goal. The amount and quality of such information depends on what background resources the deliverer has at her disposal. A dialogue system that employs an ontological model of the domain can, for instance, “know” that certain concepts are not covered by an underlying database (out-of-domain concepts). Other examples of exceptional results are empty result sets, or under-specified queries.

In our view of recommendation and information-providing dialogue, exceptional results is what drives the interview strategy. There is a symmetry between under-specified queries and resulting clarification dialogues, and preference acquisition for recommendation delivery. This similar structure makes it possible to use the interview strategy for both preference interviews, and for clarification strategies for poorly understood database requests. The generality of the interview and delivery strategies makes them suitable to use for both phenomena, as exemplified in Section 3.

It is important to note that the occurrences of exceptional results increase in human-computer dialogue compared to human-human ditto. A typical example is infor-

mation about out-of-domain concepts which is a direct consequence of the limited domain descriptions in information systems to date.

## 3 Recommendation as Dialogue Behaviors

This section presents the model for the interview and delivery strategies that is used in the conversational music recommender system CORESONG. The model is based directly on results from the corpus analysis. It is presented as *dialogue behavior diagrams* based on UML activity diagrams which also makes it computationally attractive<sup>3</sup>.

CORESONG is a conversational recommender system implemented as a web application with typed chat-style interaction. The purpose of the system is to help users build music play lists for various situations, such as exercising or driving the car. CORESONG uses a relational database with song, genre, artist, album, and year information for over 8,000 songs (with more than 50 genres, 2,000 artists, and 650 albums). In order to produce recommendations the system also employs a content-based<sup>4</sup> recommendation engine (Burke, 2002).

### 3.1 The Dialogue Behavior Model

CORESONG’s dialogue strategy is realized using a set of dialogue behavior automata. Each automaton describes the computational behavior of the dialogue strategy manager of the system in a (mathematically) precise and accurate way.

The model is defined as *strata* (i.e. layers) of dialogue behavior diagrams. Each such diagram formalizes a coherent behavior of the system, in terms of an activity diagram with enhanced notation for this purpose.

In particular, to formalize user and system utterances the dialogue behavior diagrams use the Preference Conversational Query Language, PCQL (Wärnestål et al., 2007). PCQL action statements are used for representation of both user and system acts and treat questions and statements in a symmetric way, both in and out of the system.

A PCQL action statement has the form

$$\langle \text{TAG} \rangle [ \langle fp \rangle ] \{ \text{VALUES} [ \langle vlist \rangle ] \}^?$$

where  $\langle \text{TAG} \rangle$  is an action tag,  $\langle fp \rangle$  is an FP state formula and the third argument  $\langle vlist \rangle$  is an optional argument that holds a map of entity values (normally used for system

<sup>3</sup>Since activity diagrams naturally relate to state diagrams, the emerging standard of W3C’s State Chart XML should be suitable for specifying dialogue behavior diagrams.

<sup>4</sup>A version utilizing a hybrid engine that mixes collaborative filtering and content-based recommendations is under development.

answers only)<sup>5</sup>.

The FP state formula describes some aspects of the factual and preference state (the FP state) of the dialogue agent. The action tag expresses an *action* performed by the dialogue participant, a dialogue act, where the acting agent is doing something that will result in a response from the dialogue partner. For example, the utterance “What do you think about the genre techno?” can be represented as a the PCQL statement: ASK [⊙ Genre = Techno]<sup>6</sup>. PCQL makes no assumption on the constituents of the action tag set, since they are highly application-dependent. PCQL statements and FP states can express fragmented utterances, which (naturally) are common in on-going dialogue. A basic fragment resolution module, responsible for resolving anaphoric expressions, validates and completes PCQL statements prior to the dialogue behavior strata. This means that each dialogue behavior instance operates on complete PCQL statements.

Each dialogue behavior diagram can be used to produce one or several running dialogue behavior instances in the running system. A key point of the proposed model is to have several instances of the same behavior for different external resources (e.g. a database of products), where each behavior instance contributes to the complete dialogue system behavior.

### 3.2 Basic Behaviors

Figures 3 and 4 define two dialogue behaviors, interview and direct delivery. For a first version of CORESONG, these dialogue behaviors are instantiated for each of the two external resources, a music information database and a recommender engine.

The core symbol of the dialogue behavior diagram is an activity state where an action is carried out in the dialogue and labeled arrows are conditional transitions (*guards*) between states. Guards operate on FP state formulas that in turn may be manipulated by calls to back-end resources. Each **call** returns a result in the form of a PCQL action statement (stored in the % variable) that can be used by successive activities and guard functions.

Consider the direct delivery behavior in Figure 4. As the basic interview behavior it awaits input (the **in** node 21) and checks whether it is a request/ask move by using the *is-ask* function. If the check passes, the external resource connected to this particular diagram is queried (the **call** to the *LookUp* in node 24). The result of the query is presented to the user—if the size of the result set is within a pre-defined range—as a PCQL statement in node 25. The dialogue behavior diagram then informs the

<sup>5</sup>More details and elaborate PCQL statements are presented in (Wärnestål et al., 2007)

<sup>6</sup>The operator symbol ⊙ is used to request preference polarity and strength.

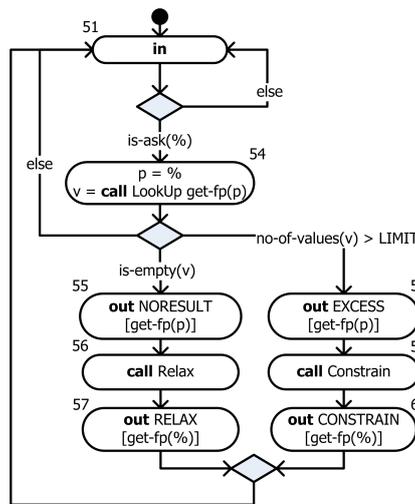


Figure 3: The interview dialogue behavior.

user that it is ready for a new turn (node 26), awaiting a new user move in node 21.

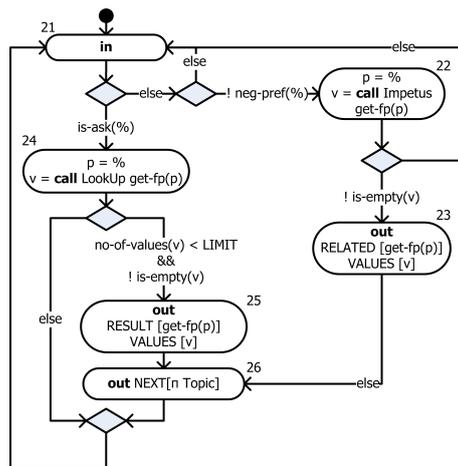


Figure 4: The direct delivery dialogue behavior.

Preferences are detected and recorded, annotated with user-defined *situations* (e.g. exercise, driving, work) in a preference model. Preference strengths are calculated (Carberry et al., 1999) and used by the external recommender engine resource, which in turn trigger node transitions in its connected dialogue behavior instances.

Figure 5 exemplifies how the interview and direct delivery dialogue behaviors of a database and a recommender engine resource collaborate in the dialogue. Utterance S1a is generated because the database has a valid result set based on the user’s *preference* statement in U1. The result set is created by a tailored **call** to the database (node 22 in Figure 4) which retrieves related information to encourage the user to react on domain informa-

Action Tag	Sample Output Database	Recommender Engine
RESULT	These artists belong to the genre <i>techno</i> :	You might like the song $x$
MOTIVATE		You might like song $x$ because it is liked by others who like $y$
RECOMMEND		Have you heard the song $x$ ?
NORERESULT	There were no matches to your query	There are no products that match your preferences
RELAX	Do you want to disregard artist constraint?	Do you have any other preferences? <sup>a</sup>
EXCESS	There were too many matches to your query	I need more preferences in order to give you recommendations
CONSTRAIN	Please provide a genre	Are there any albums you like in particular?

<sup>a</sup>The utterance needs to be tailored depending on the recommender engine type.

Table 1: Examples of action tags and sample output from CORESONG’s two back-end resources (the music database and the recommender engine). MOTIVATE and RECOMMEND are part of the indirect delivery behavior described in section 3.3 and only valid for the recommender engine.

tion (state 23). Since the preference model needs to be fleshed out with more preferences before the recommendation engine is ready to produce recommendations (transition  $54 \Rightarrow 58$  in Figure 3), additional constraints are needed. When accessing the preference model (transition  $59 \Rightarrow 60$  in Figure 3) the system is informed of which attribute that should be constrained (realized as a CONSTRAIN action in S1c). Constrain and Relax are back-end resources that keep track of the order in which domain entity types should be requested, depending on the content of the user’s preference model. U2 is a *factual* query that

- U1 I like the artist *Audioslave*.  
 S1a These are the albums that belong to the artist *Audioslave*: [list]  
 S1b I need more preferences in order to give you recommendations.  
 S1c Please provide a genre that you like.  
 U2 What albums have the genre *Alternative*?  
 S2a These albums belong to the genre *Alternative*: [list]  
 S2b You might like the song *Original Fire*.  
 S2c What else do you want to know?

Figure 5: Multiple deliveries from two back-end resources (database and recommender engine) and two dialogue behaviors (interview and direct delivery). S = system, U = user.

yet again renders the database with a valid result set, resulting in the S2a delivery (corresponding to node 25 in Figure 4). Since we model information requests as preferences (Carberry et al., 1999) the user’s utterance in U2 is also interpreted as a proper response to S1c in the behaviors connected to the recommendation engine. This turn, we find that the recommender engine has a recommendation ready (due to the additional genre preference

in U2). S2b and S2c are thus part of the direct delivery behavior connected to a recommender engine (nodes 34 and 35).

### 3.3 Extending the Recommendation Behavior

As the dialogue excerpt in Figure 5 shows, the direct delivery instances in Figure 4 accommodate basic deliveries of both recommendations and database results. Recommendations, according to the corpus study, are often not as “blunt” as the direct delivery version of Figure 5, however. To enhance CORESONG’s recommendation strategy and make it more human-like, we connect a new *indirect* delivery dialogue behavior to the recommender engine resource (see Figure 6). However, we leave the direct delivery dialogue behavior for the database resource intact, since it is still used as-is for factual questions, and conversational impetus.

By running this new dialogue behavior in parallel with the interviews and direct delivery dialogue behaviors we achieve dialogues such as the one exemplified in Figure 7 in cases when the recommender engine’s status is ready to deliver recommendations. This complete behavior mimics the distilled corpus more closely. The system appears more “modest” in its recommendations, since it states a *motivation* before presenting the actual recommendation (S1b/S3a). Since the system cannot know whether the suggestion is previously familiar to the user, it delivers the recommendations in the form of questions (S1c/S3b). A set of new action tags are needed for the **out** activity states (utterance examples are found in Figure 7 and Table 1): MOTIVATE (S1b), RECOMMEND (S1c), and ASKRATE (S2a).

## 4 The Emergent Dialogue Strategy

In CORESONG, seamless integration of factual and preference aspects of the dialogue is achieved by having the individual dialogue behaviors combine their resulting

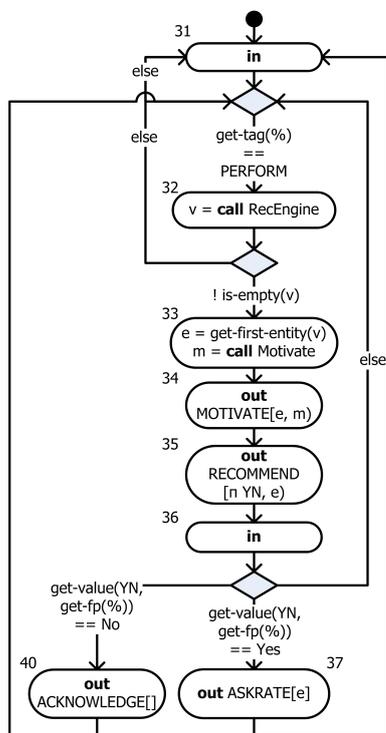


Figure 6: CORESONG’s indirect recommendation delivery dialogue behavior.

- U1 What albums belong to the genre *Alternative*?
- S1a These albums belong to the genre *Alternative*:  
[list]
- S1b You might like the song *Original Fire* because it is a song by Audioslave that belongs to the genre *Alternative*.
- S1c Have you heard it?
- U2 Yes
- S2a Ok. What do you think about that song?
- U3 It’s great
- S3a You might like the song *One* by Creed because it is liked by others who like Audioslave.
- S3b Have you heard it?

Figure 7: Two recommendation deliveries in a dialogue strategy following the direct database delivery (in Figure 4) and the indirect recommendation delivery (in Figure 6).

outputs. Thus, the resulting dialogue strategy emerges from such combinations in a robust way.

*Factual* dialogues use instances of the interview and direct delivery dialogue behaviors (Figures 3 and 4). *Preferential* (and recommendation) dialogues also use the interview dialogue behavior, but instances of the indirect recommendation delivery behavior instead of the direct delivery behavior (compare Figure 6).

As the corpus suggests, the dialogue has a distinct exploratory flavor where preference eliciting goes hand in hand with a gradually increasing understanding of the domain (Swearingen and Sinha, 2002; Wärnestål, 2005). It is therefore important that factual interviews from the user’s perspective are seamlessly blended with the preferential system interview as well as recommendation delivery.

Combining the behaviors’ output into a coherent system utterance is by no means trivial. In CORESONG the solution is pragmatic and divided into two constructs: *behavior priority* and *action tag heuristics*.

**Priority** We index behaviors with a priority and order the *out* PCQL statements accordingly (ascending order). For example, the indirect delivery has a higher priority than direct delivery. Technically, the interview has a higher priority than delivery, but in cases where a delivery and an interview are connected to the same resource, they are naturally mutually exclusive. However, if a delivery and an interview are connected to different resources, the higher interview priority comes into play.

**Heuristic** Secondly, we employ a simple heuristic when surface realizing the output. It states that informing action statements always precedes action statements that hold ask/requesting tags, and that the ask/request action with the highest priority is chosen. For example, this guarantees that utterance S1c always occurs *after* S1a in Figure 7 even though they origin from different behaviors.

**Conversational Impetus** On a final note, since most users are not aware of all their preferences at the outset of a dialogue session, it is important to trigger preference volunteering. It is mostly when exposed to information that the user brings her preferences into play (Carberry et al., 1999). This triggering we call *conversational impetus* since this is what drives the preference dialogue forward (Wärnestål, 2005). Conversational impetus in CORESONG is reached by accessing the database for all user input—not only information queries. The user is thus presented with related information that can encourage her to provide more preferences in a reactive manner (see U1 and S2a in Figure 1).

Our behavior-based approach supports such conversational impetus naturally since it is a direct consequence of