Natural Language Engineering **15** (3): 315–353. © 2009 Cambridge University Press
doi:10.1017/S1351324909005105 Printed in the United Kingdom

Automatic annotation of context and speech acts for dialogue corpora

KALLIRROI GEORGILA[1], OLIVER LEMON[2], JAMES HENDERSON[3] and JOHANNA D. MOORE[2]

[1]*Institute for Creative Technologies, University of Southern California, 13274 Fiji Way, Marina del Rey, CA 90292, USA*
[2]*School of Informatics, University of Edinburgh, 10 Crichton Street, Edinburgh, EH8 9AB, UK*
[3]*Department of Computer Science, University of Geneva, Battelle bâtiment A, 7 route de Drize, 1227 Carouge, Switzerland*
e-mails: kgeorgila@ict.usc.edu, olemon@inf.ed.ac.uk, james.henderson@cui.unige.ch, j.moore@ed.ac.uk

(*Received 14 June 2006; revised 2 August 2007, 5 March 2009*)

Abstract

Richly annotated dialogue corpora are essential for new research directions in statistical learning approaches to dialogue management, context-sensitive interpretation, and context-sensitive speech recognition. In particular, large dialogue corpora annotated with contextual information and speech acts are urgently required. We explore how existing dialogue corpora (usually consisting of utterance transcriptions) can be automatically processed to yield new corpora where dialogue context and speech acts are accurately represented. We present a conceptual and computational framework for generating such corpora. As an example, we present and evaluate an automatic annotation system which builds 'Information State Update' (ISU) representations of dialogue context for the COMMUNICATOR (2000 and 2001) corpora of human–machine dialogues (2,331 dialogues). The purposes of this annotation are to generate corpora for reinforcement learning of dialogue policies, for building user simulations, for evaluating different dialogue strategies against a baseline, and for training models for context-dependent interpretation and speech recognition. The automatic annotation system parses system and user utterances into speech acts and builds up sequences of dialogue context representations using an ISU dialogue manager. We present the architecture of the automatic annotation system and a detailed example to illustrate how the system components interact to produce the annotations. We also evaluate the annotations, with respect to the task completion metrics of the original corpus and in comparison to hand-annotated data and annotations produced by a baseline automatic system. The automatic annotations perform well and largely outperform the baseline automatic annotations in all measures. The resulting annotated corpus has been used to train high-quality user simulations and to learn successful dialogue strategies. The final corpus will be made publicly available.

1 Introduction

Richly annotated dialogue corpora are essential for new research directions in statistical learning approaches to dialogue management (Walker, Fromer and Narayanan

1998; Singh *et al.* 1999; Levin, Pieraccini and Eckert 2000; Henderson, Lemon and Georgila 2005, 2008), user simulation (Scheffler and Young 2001; Georgila, Henderson and Lemon 2005a, 2006; Schatzmann, Georgila and Young 2005a, 2006; Schatzmann, Thomson and Young 2007; Georgila, Wolters and Moore 2008b), context-sensitive interpretation, and context-sensitive speech recognition (Gabsdil and Lemon 2004). In particular, large dialogue corpora annotated with contextual information and speech acts are urgently required for training and testing dialogue strategies and user simulations. However, hand annotations are expensive and time consuming. In addition, they cannot be reused for the annotation of new corpora even if they share the same domain. We explore how existing dialogue corpora (usually consisting of utterance transcriptions) can be automatically processed to yield new corpora where dialogue context and speech acts are represented. We present a conceptual and computational framework for generating such corpora. In particular, we propose the use of dialogue system simulation for automatically annotating dialogue corpora.

Later, we present and evaluate an automatic annotation system which builds 'Information State Update' (ISU) representations of dialogue context (Larsson and Traum 2000; Bos *et al.* 2003; Lemon and Gruenstein 2003) for the COMMUNICATOR (2000 and 2001) corpora of spoken human–machine dialogues (2,331 dialogues) in the domain of telephone flight reservations (Walker *et al.* 2001a, 2002; Walker, Passonneau and Boland 2001b). Users of the COMMUNICATOR systems try to book a flight and they may also make hotel or car-rental arrangements. This is one instance of our approach to the problem of automatic annotation of large corpora.

1.1 The automatic annotation task

In general, spoken or written dialogue corpora consist of transcribed (or automatically recognised) speaker turns, with possibly some additional annotations, such as timing information, gestures, and perhaps some type of dialogue-act or speech-act tagging. For statistical learning approaches, we need to construct from these data sets, a more richly annotated version of the corpora where dialogue contexts and speech acts are represented.[1] In general we need to compute a function from $(Speaker_i, Utterance_j)$ to $(Context_j, Speech\_act_{i,j})$. That is, after every speaker utterance we desire a representation of the speech act of that utterance, and of the whole dialogue context after that utterance.

In the case of open-domain human–human corpora, this is a very challenging task, because computing the context relies on computing the speech act and content of each utterance. However, for human–machine corpora in limited domains, the problem becomes more tractable, because often the speech act and/or content of the

[1] The utterances of a dialogue are primarily communicative acts between the two conversants. For the specific case of natural language utterances the term *speech act* was first used by Searle (1969). Another term used for the same concept is *dialogue act* (Traum 2000). We will use the terms speech act and dialogue act interchangeably. *Dialogue context* is defined as what has been established so far in the conversation (Lemon and Gruenstein 2003), e.g. the status of the slots (whether they are filled or confirmed) in a slot-filling task, the history of speech acts, etc.

machine-generated utterances are known and logged, and because limited-domain dialogues can feasibly be parsed using keyword spotting or relatively simple semantic parsing techniques.

We thus distinguish six basic levels of the task in descending order of difficulty

(1) Human–human open-domain corpora.
(2) Human–machine open-domain corpora.
(3) Human–human closed-domain corpora.
(4) Human–machine closed-domain corpora consisting of transcribed and/or recognised utterances.
(5) Human–machine closed-domain corpora consisting of transcribed and/or recognised utterances, where machine speech acts and/or content are already tagged.
(6) Human–machine closed-domain corpora consisting of transcribed and/or recognised utterances, where both human and machine speech acts and/or content are already tagged.

Our approach relates to levels 3–6. We provide a tool (at level 5) for task-oriented dialogues that maps from a human–machine corpus (COMMUNICATOR) consisting of utterances and machine dialogue act tags to full context representations including speech act tags for user utterances (either transcribed or recognised). Note that our tool can reconstruct information that was not logged during the course of the dialogue. For example, many dialogue systems do not log information about the dialogue context or semantic interpretations of the user's utterances. In addition, in 'Wizard-of-Oz' experiments (where a human pretends to be a machine) (Georgila *et al.* 2008a; Rieser and Lemon 2008) usually only the wizard's actions are logged with semantic tags, with no information about the underlying wizard's strategy or the context in which these actions take place. Thus it is important to have a tool for post-processing such logs and the users' utterances in order to extract context information. In Section 8 we discuss how our tool has also been used for automatically annotating a corpus generated in a Wizard-of-Oz experiment (Georgila *et al.* 2008a).

The contribution of this work thus lies in several areas

- principles for context annotation,
- principles for speech act annotation,
- a proposed standard document type definition (DTD) for context and speech act annotations,
- extension of the DATE annotation scheme (Walker and Passonneau 2001),
- a computational tool and framework for automatic annotation of task-oriented dialogue data,
- a richly annotated dialogue corpus – the first dialogue corpus to be annotated with full 'Information State' context representations.

1.2 The 'Information State Update' approach

To provide us with a principled approach to defining the dialogue contexts which we annotate, we adopt the 'Information State Update' (ISU) approach to dialogue

modelling. The ISU approach supports the development of generic and flexible dialogue systems by using rich representations of dialogue context.

'The term *Information State* of a dialogue represents the information necessary to distinguish it from other dialogues, representing the cumulative additions from previous actions in the dialogue, and motivating future action' (Larsson and Traum 2000).

Technically, Information States represent dialogue context as a large set of features, e.g. speech acts, tasks, filled information slots (e.g. destination = Paris), confirmed information slots, speech recognition confidence scores, etc. Update rules then formalise the ways that information states or contexts change as the dialogue progresses. Each rule consists of a set of applicability conditions and a set of effects. The applicability conditions specify aspects of the information state that must be present for the rule to be appropriate. Effects are changes that are made to the information state when the rule has been applied. For full details see Larsson and Traum (2000) and Bos *et al.* (2003).[2]

By using these information states as our notion of dialogue context, a dialogue corpus annotated with contexts can be used in a number of ways

- data for training reinforcement learning (RL) approaches to dialogue management,
- data for training and testing user simulations,
- baseline for evaluating new dialogue strategies,
- data for training models for context-dependent interpretation and speech recognition.

In general, for such research we require data that has either been generated and logged by context-tracking dialogue systems (Gabsdil and Lemon 2004; Lemon, Georgila and Henderson 2006a; Lemon *et al.* 2006b) or that has been subsequently annotated (or a mixture of both). Both preliminary versions of our annotations and the version that we present here have been used successfully in Georgila *et al.* (2005a, 2006), Henderson *et al.* (2005, 2008), Schatzmann *et al.* (2005a, 2005b), Frampton and Lemon (2006). Note that prior work on dialogue context annotations (Poesio *et al.* 1999) was not automated, and was not suitable for large-scale annotations.

The outline of the paper is as follows: In Section 2 we survey basic principles for annotating dialogue data with feature values for learning approaches. In Section 2.1 we describe briefly the original COMMUNICATOR corpora which we take as our example. Section 3 describes the annotation system. In Section 4 a detailed example is provided. Section 5 focuses on specific methods required for the COMMUNICATOR data and Section 6 presents our evaluation of the automatic annotations. Section 7

[2] All dialogue systems have internal dialogue states for storing information required through the course of the dialogue. Information States provide a general theoretical framework for building dialogue systems and may include aspects of dialogue state as well as more mentalistic notions such as beliefs, intentions, plans, etc. It is very easy to model a dialogue state as an Information State, which makes our approach applicable to corpora derived from systems that were not based on the ISU approach. However, the opposite is not necessarily true. For a full discussion on the difference between information states and dialogue states see Larsson and Traum (2000).

describes how the annotation system was ported from the flight reservations domain to the city information domain and how it could be used in different types or genres of dialogue, such as tutorial dialogue. Section 8 discusses its limitations and Section 9 presents our conclusions.

2 Context annotation principles

In current research, the question arises of what types of information should ideally be logged or annotated for the purposes of building simulated users, optimising context-based dialogue systems via Reinforcement Learning (Walker *et al.* 1998; Singh *et al.* 1999; Levin *et al.* 2000; Young 2000), and training dialogue-context models. We focus on task-oriented dialogues and our approach is to divide the types of information required into five main levels (see Figure 2): *dialogue-level, task-level, low-level, history-level*, and *reward-level*. We also divide the logging and annotations required into information about *utterances* and information about *states*. Utterances (by humans or systems) will have dialogue-level, task-level, and low-level features, while dialogue states will additionally contain some history-level information (see Figure 2). Entire dialogues are assigned reward features, e.g. taken from questionnaires filled by users. This framework has also been adopted by Rieser, Kruijff-Korbayová and Lemon (2005a, 2005b), Andreani *et al.* (2006) and Rieser and Lemon (2006, 2008).

As discussed in Section 7 the structure of the information state may have to be modified for other types of dialogue. For example, for tutorial dialogues, there would be additional annotation levels and Information State fields to encode the progress of the student and the tutor's tutoring style. Furthermore, some Information State fields related to task-oriented slot-filling dialogues (e.g. 'FilledSlot', 'FilledSlotsHist', etc., depicted in Figure 2) would be redundant. Thus, our context annotation principles can be extended and/or modified to deal with more complex or different types of dialogue.

The dialogue annotation task has two main components: annotating utterances and annotating states. In the original COMMUNICATOR corpus, only the system utterances are annotated. Our annotation system adds annotations for the user utterances, and constructs context annotations for the states which follow each utterance.

2.1 *The original* COMMUNICATOR *corpora*

The COMMUNICATOR corpora (2000 and 2001) consist of spoken human–machine dialogues in the domain of telephone flight reservations. The users always try to book a flight but they may also try to select a hotel or rent a car. The dialogues are primarily 'slot-filling' dialogues, with information being presented to the user at the end of the conversation.

The COMMUNICATOR corpora have recently been released by the Linguistic Data Consortium (LDC). A particular problem is that although the COMMUNICATOR corpus is the largest publicly available corpus of speech-act-annotated dialogues,

```
turn start_time = "988306674.170"
     end_time = "988306677.510"
     speaker = "user"
     number = "5"
utterance start_time = "988306674.170"
          end_time = "988306677.510"
          number = "5"
asr = october three first late morning
ne_asr = <DATE_TIME>october three first late morning</DATE_TIME>
transcription = october thirty first late morning
ne_transcription = <DATE_TIME>october thirty first late morning</DATE_TIME>
```

Fig. 1. Example user turn from the original COMMUNICATOR corpus, simplified from the original XML format; *asr* is the output of the speech recogniser, and *ne_asr* is the output of the speech recogniser tagged with named entity information.

it does not meet our requirements on corpus annotation for dialogue strategy learning, user simulation, and representation of dialogue context. For example, the user dialogue inputs were not annotated with speech act classifications, and no representation of dialogue context was included. Moreover, there was no information about the status of the slots, which is critical for learning dialogue strategies and user simulations.

The original COMMUNICATOR corpora have previously been annotated (but only for the system's side of the dialogue) using the DATE (Dialogue Act Tagging for Evaluation) scheme (Walker and Passonneau 2001) described in Section 2.2. Figure 1 shows an extract from the 2001 collection. For user utterances both the speech recognition output and the human transcription of the user's input are provided but there is no speech act tagging. Also, for each dialogue there is information about the actual and perceived task completion and user satisfaction scores, based on the PARADISE evaluation framework (Walker, Kamm and Litman 2000). For the user satisfaction scores, users had to answer questions in a Likert scale (1–5) about the ease of the tasks they had to accomplish, whether it was easy or not to understand the system, their expertise, whether the system behaved as expected, and if they would use the system again in the future or not.

The 2000 collection contains 648 dialogues recording the interactions of humans with nine systems, and the 2001 collection contains 1,683 dialogues with eight systems. Table 1 shows some statistics of the two collections. In the 2000 collection each turn contains only one utterance but in the 2001 corpus a turn may contain more than one utterance. More details about the COMMUNICATOR corpora can be found in Walker *et al.* (2001a, 2001b, 2002).

We now present the DATE scheme used in the original COMMUNICATOR corpus and then our extension of it, including the dialogue information state annotations. This annotation scheme has become known as the 'TALK context annotation framework' and it has an associated XML document type definition (DTD).[3]

[3] Available at http://homepages.inf.ed.ac.uk/olemon/talk2005v2.dtd

Table 1. *Statistics of the 2000 and 2001* COMMUNICATOR *data*

	2000	2001	Total
Number of dialogues	648	1,683	2,331
Number of turns	24,728	78,718	103,446
Number of system turns	13,013	39,419	52,432
Number of user turns	11,715	39,299	51,014
Number of utterances	24,728	89,666	114,394
Number of system utterances	13,013	50,159	63,172
Number of user utterances	11,715	39,507	51,222
Number of system dialogue acts	22,752	85,881	108,633

2.2 The DATE annotation scheme

The system utterances in the original COMMUNICATOR corpus are annotated using the DATE scheme (Walker and Passonneau 2001). The DATE scheme was developed for providing quantitative metrics for comparing and evaluating the nine different DARPA COMMUNICATOR spoken dialogue systems. The scheme employs the following three orthogonal dimensions of utterance classification:

- *conversational domain*: ABOUT_TASK, ABOUT_COMMUNICATION, SITUATION_FRAME,
- *task–subtask*: TOP_LEVEL_TRIP (ORIG_CITY, DEST_CITY, DEPART_ARRIVE_DATE, DEPART_ARRIVE_TIME, AIRLINE, TRIP_TYPE, RETRIEVAL, ITINERARY), GROUND (HOTEL, CAR),
- *speech act*: REQUEST_INFO, PRESENT_INFO, OFFER, ACKNOWLEDGEMENT, STATUS_REPORT, EXPLICIT_CONFIRM, IMPLICIT_CONFIRM, INSTRUCTION, APOLOGY, OPENING_CLOSING.

The conversational domain dimension categorises each utterance as belonging to a particular 'arena of conversational action.' Here 'ABOUT_TASK' refers to the domain task (in COMMUNICATOR this is air travel, hotel, and car-rental booking), and 'ABOUT_COMMUNICATION' refers to conversational actions managing the communication channel (e.g. 'are you still there?'). 'SITUATION_FRAME' utterances manage the 'culturally relevant framing expectations' in the dialogue (e.g. that the conversation will be in English, or that the system cannot issue airline tickets).

The task–subtask dimension relates to a model of the domain tasks that the dialogue system is designed to support. In COMMUNICATOR there were two main tasks: booking a flight (TOP_LEVEL_TRIP), and 'GROUND' which was to determine whether the user also wanted to book a car rental and/or a hotel. The subtasks were elements such as finding the dates and times of the flights.

The speech act dimension relates to the utterance's communicative goal. The speech acts used are relatively standard, and are described in detail in Walker and Passonneau (2001). Note that in the COMMUNICATOR data only the system's side of the dialogue is already annotated using the DATE scheme.

2.3 Extending DATE: The new utterance annotation scheme

Given these system utterance annotations, the first part of our task of producing the new annotations is to interpret the user's input and find its effect on the dialogue context. In other words, we need to associate the user utterances with the correct speech acts and tasks. This process is complicated by the fact that each utterance may involve multiple tasks and that each task could include multiple speech acts.

To accommodate the annotation of user utterances, we needed to extend the DATE scheme in several ways. One problem was that the original annotation of the COMMUNICATOR data does not distinguish between the different origin and destination cities for different legs of a multiple-leg trip. Thus the tag 'DEST_CITY' could be used for any type of destination, regardless of whether the trip is single or multiple-leg. However, we believe that it is important to annotate these distinctions so that there is no overwriting of the values in filled slots such as 'DEST_CITY,' 'DEPART_DATE,' etc. For this reason we extended the set of subtasks to include the distinction between different journey legs. Another problem is that the original annotation of the COMMUNICATOR data does not distinguish between departure and arrival dates or times, and sometimes it contains times which have been labelled as dates. Thus we also made this distinction, and fixed these mislabellings.

We use the following extended tasks (the equivalent of subtasks in the DATE scheme) and speech acts for annotating user utterances. These are in addition to the DATE scheme (Walker and Passonneau 2001) used for the system prompts annotation

- User speech acts: PROVIDE_INFO, REPROVIDE_INFO, CORRECT_INFO, REJECT_INFO, YES_ANSWER, NO_ANSWER, QUESTION, COMMAND.
- Tasks which take values: CONTINUE_DEST_CITY, DEPART_DATE, CONTINUE_DEPART_DATE, RETURN_DEPART_DATE, ARRIVE_DATE, CONTINUE_ARRIVE_DATE, RETURN_ARRIVE_DATE, DEPART_TIME, CONTINUE_DEPART_TIME, RETURN_DEPART_TIME, ARRIVE_TIME, CONTINUE_ARRIVE_TIME, RETURN_ARRIVE_TIME, HOTEL_CITY, HOTEL_LOCATION, HOTEL_NAME, CAR_CITY, CAR_RENTAL, RENTAL_COMPANY, ID_NUMBER, NUMBER.
- Tasks which are either true or false: CONTINUE_TRIP, NO_CONTINUE_TRIP, RETURN_TRIP, NO_RETURN_TRIP, ACCEPT_FLIGHT_OFFER, REJECT_FLIGHT_OFFER, ACCEPT_FLIGHT_SUMMARY, REJECT_FLIGHT_SUMMARY, ACCEPT_CAR_OFFER, REJECT_CAR_OFFER, ACCEPT_GROUND_OFFER, REJECT_GROUND_OFFER, ACCEPT_HOTEL_OFFER, REJECT_HOTEL_OFFER, CAR_INTEREST, NO_AIRLINE_PREFERENCE, CHANGE_AIRLINE, FLIGHT_INTEREST, SEND_ITINERARY, PRICE_ITINERARY, CONTINUE, REQUEST_HELP, REQUEST_REPETITION, REQUEST_STOP, NONSTOP_FLIGHT, BYE, START_OVER.

Assigning ⟨speech act, task⟩ pairs to user utterances is basically a dialogue act recognition task[4] where each dialogue act tag is a ⟨speech act, task⟩ pair. Our

[4] Other equivalent terms are dialogue act classification, dialogue act detection, and dialogue act tagging.

case is very similar to dialogue act tagging with multidimensional tag sets, which assigns a combination of tags to each utterance segment (Lesch, Kleinbauer and Alexandersson 2005). Although only one tag can be assigned per utterance segment ('from Boston' is unambiguously a 'PROVIDE_INFO(ORIG_CITY)' and 'to Orlando' a 'PROVIDE_INFO(DEST_CITY)'), each user utterance may consist of more than one segment, the boundaries of which are not defined, thus possibly leading to disagreements in the number of tags assigned to that utterance by different annotation systems or human annotators. This makes the task of our automatic annotation system even more challenging.

Several dialogue act recognition techniques have been described in the literature so far, e.g. *n*-grams (Reithinger and Maier 1995; Reithinger and Klesen 1997; Webb, Hepple and Wilks 2005), hidden Markov models (Ries 1999; Stolcke *et al.* 2000), Bayesian networks (Keizer and op den Akker 2007), neural networks (Kipp 1998; Ries 1999), transformation-based learning (Samuel, Carberry and Vijay-Shanker 1998), etc. All these techniques assume that some part of the corpus is already annotated with dialogue acts and can be used for training statistical models. None of these statistical methods can be applied in our case because the original COMMUNICATOR corpus does not include any annotations for the user's side of the dialogue; in other words, there is no complete corpus that could be split for training and testing purposes. Therefore we are restricted to use traditional parsing techniques for each user utterance taking into account information from the previous system utterances. In particular, the speech acts and tasks are computed using a phrase spotting semantic parser which we describe in Section 3.1.

As already mentioned, because a given user utterance may contain multiple speech acts and tasks, each user utterance is potentially annotated with lists of these features. These lists are kept synchronised so that the lengths and ordering of the features are kept equal. For example, if the user gives the departure date, reprovides information about the origin city and also provides new information about the destination city then 'SpeechAct' will be '[PROVIDE_INFO, REPROVIDE_INFO, PROVIDE_INFO]' and 'Task' will be '[DEPART_DATE, ORIG_CITY, DEST_CITY].' This representation allows the pairing of speech acts and tasks to be extracted from the annotation.

2.4 The TALK context annotation scheme

Using these annotations of the system and user utterances, our system then computes context annotations for the states which follow each utterance. Figure 2 shows an example information state as it has been annotated by this automated annotation system. It corresponds to the state following the user utterance shown in Figure 1. Dialogue annotations are saved in XML format but here the information state is presented in plain text for reasons of readability. Several of these features simply specify the utterance annotations for the previous utterance. Others specify various book-keeping features such as turn number, or low-level features (discussed in Section 5.2). The most interesting features from the point of view of context annotations are those specifying which slots have been filled or confirmed, and those which accumulate information about the whole dialogue history.

DIALOGUE LEVEL
Turn: user
TurnStartTime: 988306674.170
TurnEndTime: 988306677.510
TurnNumber: 5
Speaker: user
UtteranceStartTime: 988306674.170
UtteranceEndTime: 988306677.510
UtteranceNumber: 5
ConvDomain: [about_task]
SpeechAct: [provide_info]
AsrInput: (date_time) october three first late morning (date_time)
TransInput: (date_time) october thirty first late morning (date_time)
System Output:

TASK LEVEL
Task: [depart_time]
FilledSlot: [depart_time]
FilledSlotValue: [late morning]
ConfirmedSlot: [dest_city]

LOW LEVEL
WordErrorRatenoins: 20.00
WordErrorRate: 20.00
SentenceErrorRate: 100.00
KeyWordErrorRate: 50.00

HISTORY LEVEL
FilledSlotsStatus: [orig_city],[dest_city],[depart_time]
FilledSlotsValuesStatus: [hartford connecticut],[orlando florida],[late morning]
ConfirmedSlotsStatus: [orig_city],[dest_city]
SpeechActsHist: [yes_answer],opening_closing,[ ],opening_closing,instruction,
request_info,[provide_info],implicit_confirm,request_info,[provide_info],implicit_confirm,
request_info,[provide_info]
TasksHist: [null],meta_greeting_goodbye,[ ],meta_greeting_goodbye,meta_instruct,
orig_city,[orig_city],orig_city,dest_city,[dest_city],dest_city,
depart_arrive_date,[depart_time]
FilledSlotsHist: [null],[ ],[orig_city],[dest_city],[depart_time]
FilledSlotsValuesHist: [yes],[ ],[hartford connecticut],[orlando florida],[late morning]
ConfirmedSlotsHist: [ ],[ ],[ ],[orig_city],[dest_city]

Fig. 2. Example dialogue context/Information State. User-provided information appears between [ ] brackets.

Computing context annotations for the state which follows each utterance means computing the internal state of the system at that point in the dialogue. Thus our automatic annotation system can be thought of as a dialogue system running in a reverse mode (reading backwards from the system prompts and recognised user utterances) and therefore there is no guarantee that the state produced by the automatic annotation system will be an accurate simulation of the real state that the dialogue system was in. Nevertheless, as will be shown in Section 6, the states produced by the automatic annotation system are good approximations of the real

states. We must also keep in mind that the purpose of the automatic annotations is to generate corpora to be used in machine learning experiments for learning dialogue strategies, user simulations, and models for context-dependent interpretation and speech recognition. Machine learning methods are good at compensating for errors in the training data.

The most difficult problem to solve in annotating dialogue contexts is determining (via processing discussed in Section 3) what slots have been filled, confirmed, or even emptied, by a user utterance. We define a piece of information as 'confirmed' (according to the system's perspective) only if it has been positively confirmed. Thus confirmation processing can only take place after system utterances labelled as explicit or implicit confirmation. There is no need to have a separate field for the value of the confirmed slot because the value which is confirmed must be the same as the value with which the slot has been filled.

One concern about labelling confirmation using only the information available in the original COMMUNICATOR corpus (and not the real state of the dialogue system itself) is that it assumes that the COMMUNICATOR systems had some notion of confirmation in their algorithms. For this reason, we only specify slot confirmation when the systems attempt confirmation via an explicit or implicit confirmation action. Moreover, only the speech recognition output is used for processing and deciding on the slots that will be filled or confirmed. The human transcription of the user's input is only considered for computing error rates, as explained in Section 5.2. This also ensures that we do not base our annotation on information that the systems did not have at runtime.

Note that the history-level annotations/logs of states should be computable from the other levels over the history of utterances. Therefore the features of the dialogue, task, and low levels can be expanded after the annotation/logging is completed. With some dialogue managers, e.g. those by Larsson and Traum (2000) and Bos *et al.* (2003), this can be achieved by logging whole Information States.

Note also in Figure 2 the difference between the groups of Information State fields {FilledSlotsHist, FilledSlotsValuesHist, ConfirmedSlotsHist} and {FilledSlotsStatus, FilledSlotsValuesStatus, ConfirmedSlotsStatus}. The former fields give us information about the exact order in which slots have been filled or confirmed and may contain several instances of the same slot, e.g. slot 'ORIG_CITY' could be confirmed twice. The latter fields ('FilledSlotsStatus,' etc.) inform us about the current status of the slots and thus may only contain one instance per slot. This is very important because if a confirmed slot is refilled with a new value it will remain in the 'ConfirmedSlotsHist' field even though its new value has not been confirmed yet. The history of speech acts and tasks is also included in our annotations.

Regarding the reward level, task completion (actual and perceived) metrics and user satisfaction scores, e.g. based on the PARADISE evaluation framework (Walker *et al.* 2000), may be used. As mentioned in Section 2.1 the original COMMUNICATOR corpora included such metrics, and these features are used in Henderson *et al.* (2005, 2008). Other reward features that could be computed are the dialogue duration, the number of turns, whether the user hung up, the number of slots that were filled and confirmed in a slot-filling dialogue, etc.

The TALK project context annotation framework was initially developed specifically for annotating the COMMUNICATOR data. It has also been used for a corpus of in-car MP3 player dialogues (Rieser *et al.* 2005a, 2005b; Rieser and Lemon 2006, 2008) and has been adopted for the DiSCoH project (Andreani *et al.* 2006), but in general there are several other features one might want to include in dialogue context annotations. For the dialogue level, such features include changes to issues/questions under discussion (Ginzburg 1996), changes to common ground (Clark and Brennan 1991; Traum 1994), obligations (Traum and Allen 1994), system and user intentions (Grosz and Sidner 1986), syntactic and semantic parses, and salient NPs. For the task level, such features include the number of database query results (for slot-filling/information-seeking dialogues), user goals, and confidence in each slot (e.g. low/medium/high). For the low level, such features include the amplitude of the speech signal, and word-based confidence scores. For multimodal systems, one might also want to log information about the input modalities, available output modalities, and XY co-ordinates of mouse or touch-screen gestures.

A notable constraint on the information to be useful for machine learning of dialogue strategies and context-dependent interpretation and speech recognition is that all captured features should in principle be available to a dialogue system at runtime – so that, for example, a dialogue system using a learnt policy can compute its next action in any state. This excludes, for example, word error rate from the state information usable for Reinforcement Learning, since it can only be computed after transcription of user speech. In this case, for example, automatic speech recognition (ASR) confidence scores should be used instead. It also means that we need to automatically annotate the ASR hypotheses of the systems, rather than the transcribed user utterances.

3 The automated annotation system

The annotation of the COMMUNICATOR data with information states was implemented using the DIPPER dialogue management system (Bos *et al.* 2003) and the Open Agent Architecture (OAA) (Cheyer and Martin 2001), a hub-based architecture for asynchronous communication between separate processes or 'agents.' Several OAA agents have been developed, with DIPPER being responsible for controlling their interaction. Figure 3 illustrates the components of the automatic annotation system.

In Figure 3, the OAA agent labelled *dme* is the DIPPER dialogue manager. The DIPPER tools support building (multimodal) dialogue systems, by offering a Dialogue Move Engine and interfaces to speech recognisers, speech synthesisers, parsers, and other agents. DIPPER is built on top of the Prolog or Java language depending on the DIPPER version[5] and all the processing that is not handled by the agents is performed by Prolog or Java clauses. Here we use the Prolog version. The flow chart of the automatic annotation task is depicted in Figure 4.

The OAA agent labelled *readXMLfile* is used for reading the original COMMUNICATOR corpus XML files, which contain information about dialogues, turns, utterances,

[5] DIPPER is freely available at http://www.inf.ed.ac.uk/research/isdd/

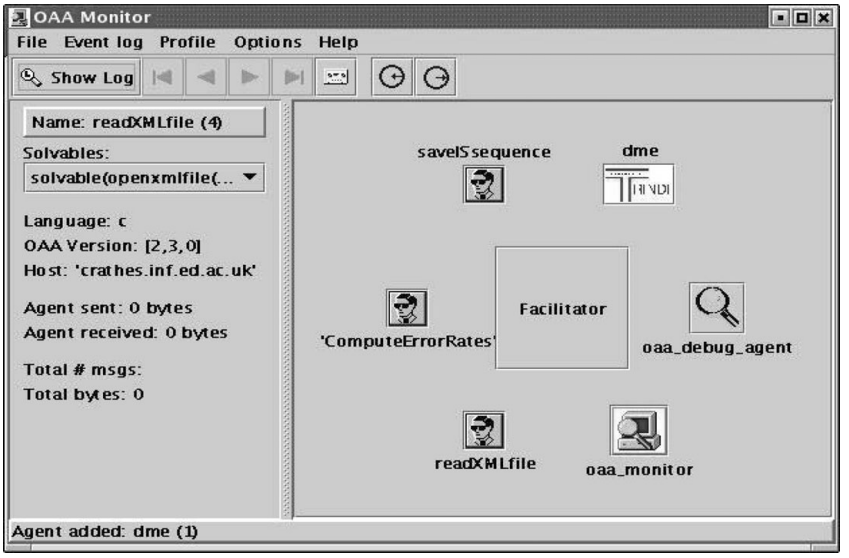

Fig. 3. The automatic annotation system viewed using the OAA monitor (the agent labelled 'dme' is the DIPPER dialogue manager).

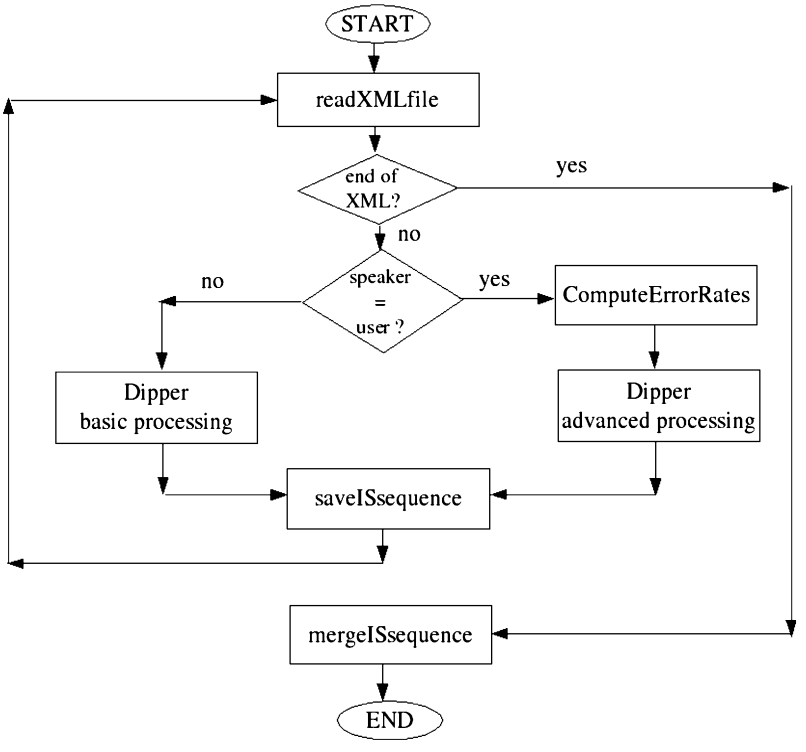

Fig. 4. The flow chart of the automatic annotation task ('basic processing' refers to tagging of system utterances, 'advanced processing' refers to contextual parsing of user utterances).

transcriptions, and so on (see Figure 1). When the agent reads information from an XML file, a corresponding DIPPER update rule fires and the dialogue information state is updated accordingly. Each information state corresponds to an utterance in the COMMUNICATOR data and a turn may contain several utterances. Most of the time one turn includes utterances of the same speaker. Though there are cases in which both the user and the system speak within the same turn. This does not cause problems in annotation since everything is computed on the utterance level. The *readXMLfile* agent reads the COMMUNICATOR XML file line by line. For each line, an information state is created by DIPPER. However, in the XML file there are lines that contain no information and just signal the end of the utterance or the turn. Thus at the end of each dialogue some state merging is required to ensure that the resulting annotations contain only information states that correspond to system or user actions.

In Figure 4 the boxes labelled 'basic' and 'advanced' DIPPER processing refer to tagging of system utterances and contextual parsing of user utterances, respectively. They will be discussed later.

An OAA agent labelled *saveISsequence* appends the current information state values to the output XML file that will finally contain the whole sequence of information states (the format of context representations is defined by the TALK DTD). The information state of Figure 2 corresponds to the user utterance depicted in Figure 1.

3.1 Parsing utterances in context

Note that in the original COMMUNICATOR corpus only the system's side of the dialogue was already annotated using the DATE scheme. The first job of the automatic annotation system is thus to compute speech acts and content for the user's side of the dialogue. In order to do this multiple stages of parsing are required – to determine the content of the user's utterance and then to determine the speech act that it performs in context.

These computations are performed using Prolog clauses in DIPPER. In the original COMMUNICATOR corpus, the user utterances (both speech recognition output and human transcription) are enriched with tags indicating the semantics of some words, e.g. cities, dates, airlines, etc. (see Figure 1). This, in addition to the fact that user utterances tend not to be very complex in this type of dialogue, facilitates parsing and thus a keyword-based parser seems to be adequate for this first stage of parsing in our task. Thus if the user says 'from Los Angeles' it is straightforward for the parser to fill the slot 'ORIG_CITY' with the value 'Los Angeles.' If however, the user said 'Los Angeles' the parser would need information from the previous context. If the previous system utterance was tagged as 'REQUEST_INFO(ORIG_CITY)' then the parser would understand that 'Los Angeles' is related to the 'ORIG_CITY' slot. As we will see in the confirmation strategies (Section 3.2) and the example that follows (Section 4), the inferences required here are not always so simple. Since some system dialogue moves in the COMMUNICATOR data were not always reliably annotated, the automatic annotation system does not always rely only on the original speech act and task tags of the system utterances but also parses the system utterances.

3.2 Confirmation strategies

When computing confirmation, it is important to take into account the different ways in which dialogue systems use various types of confirmation. In general, the systems in the COMMUNICATOR corpus typically follow one of three general confirmation strategies.

In the first strategy the system asks the user to fill a slot, then asks for confirmation (explicit or implicit), and moves to the next slot if the user confirms, or may keep asking for confirmation if the user does not cooperate. In the second strategy the system asks the user to fill several slots and then attempts to confirm them in one single turn. That means that the system's turn could consist of several utterances labelled as 'EXPLICIT_CONFIRM' or 'IMPLICIT_CONFIRM.' A third strategy, which is a variation of the second strategy is when the system tries to confirm several slots in a single utterance, e.g. 'EXPLICIT_CONFIRM(TRIP),' 'IMPLICIT_CONFIRM(ORIG_DEST_CITY).' Before confirmation the slots could be filled either in a single turn or in multiple turns.

For the first and third confirmation strategies it proves adequate to look only one or two steps backwards in the history of system utterances, whereas for the second strategy looking further back is required. To cover all these cases, when the annotation system parses the user input, it looks backwards to all the system utterances between the last user utterance and the current one (not including the last user utterance). From these system utterances, we take into account only utterances with the following speech acts: REQUEST_INFO, EXPLICIT_CONFIRM, IMPLICIT_CONFIRM, and OFFER. Other utterances (e.g. instructions) are not taken into account because they do not affect whether a slot will be filled or confirmed.

Note that the annotation system first extracts the speech acts and possible tasks related to the current user utterance and then attempts to detect confirmation based on this information. Any kind of disambiguation required, e.g. to decide whether the speech act should be tagged as 'PROVIDE_INFO' or 'REPROVIDE_INFO,' is done before confirmation. However, if speech act or task ambiguity remains unresolved it will be dealt simultaneously with confirmation, e.g. if the user uttered a city name we cannot be sure whether it refers to an origin or destination city until we consider the context. The reason for this sequential procedure is that it could be the case in a different corpus that the user's speech acts and tasks were already annotated. In that case the annotation system would only have to compute the confirmed slots. Thus the separation of parsing and confirmation computations serve the purpose of modularisation.

For the first two confirmation strategies, the annotation system should check whether the tasks extracted by parsing the user's utterance are included in the task labels of the previous explicit or implicit confirmation system prompts. Then according to the type of speech act (YES_ANSWER, NO_ANSWER, REPROVIDE_INFO, PROVIDE_INFO, CORRECT_INFO, etc.) the system confirms one or more previously filled slots or fills one or more new ones.

The third confirmation strategy involves system utterances such as 'You are travelling from Boston to Chicago on the 5th of May. Is that correct?' If the user says 'yes' then this is a confirmation of three different slots, so the value of the 'Task' feature is '[ORIG_CITY, DEST_CITY, DEPART_DATE].' To infer this the annotation

```
(S1) what city are you leaving from?
                                (request_info, orig_city)
(U1) <CITY>hartford connecticut</CITY>
(S2) a flight from <CITY>hartford</CITY>,
                                (implicit_confirm, orig_city)
(S3) where would you like to go?
                                (request_info, dest_city)
(U2) <CITY>orlando florida</CITY>
(S4) traveling to <CITY>orlando</CITY>,
                                (implicit_confirm, dest_city)
(S5) on what date would you like to travel?
                                (request_info, depart_arrive_date)
(U3) <DATE_TIME>october three first late morning</DATE_TIME>
(S6) traveling <DATE_TIME>late morning</DATE_TIME>,
                                (implicit_confirm, depart_arrive_date)
(S7) and what date would you like to travel?
                                (request_info, depart_arrive_date)
(U4) <DATE_TIME>october thirty one</DATE_TIME>
```

Fig. 5. Original COMMUNICATOR data extract.

system has to parse the system utterance as well. Because the 'SpeechAct' feature needs to be aligned with the 'Task' feature, its value is '[YES_ANSWER, YES_ANSWER, YES_ANSWER].'

In the Appendix, the processes for assigning speech acts and tasks to each utterance, and calculating confirmations and dialogue context, are presented in the form of pseudocode.

4 Example context annotation

We now examine an extract from the original COMMUNICATOR 2001 data (see Figure 5) and its new context annotation (see Figure 2). System utterances are marked with 'S(n)' and user utterances as 'U(n)' where n is the number of the utterance. For the system utterances the speech act and task pairs are given, for the user utterances only the speech recognition output is provided.[6]

In utterance (U3) the user gives the departure date and time. However, the speech recognition output 'october three first' was not considered by the original COMMUNICATOR system to be a valid date, so the system understands only the time 'late morning' and tries to confirm it in (S6). As we see in (S6) the speech act is 'IMPLICIT_CONFIRM' and the task is tagged as 'DEPART_ARRIVE_DATE' instead of 'DEPART_ARRIVE_TIME.' Similar phenomena cause problems for correctly annotating the dialogues. In this example, in (U3) our automatic annotation system fills slot 'DEPART_TIME' with the value 'late morning' and confirms the 'DEST_CITY' slot. Then it reads the next system utterance (S6). Note that if it considers only the task label 'DEPART_ARRIVE_DATE' it will attempt to confirm the wrong slot 'DEPART_ARRIVE_DATE,'

[6] The human transcription of the user input is also available but not used for the reasons discussed in Section 2.4.

or in other words it will try to confirm a slot that has not been filled yet. Therefore routines have been implemented so that the system can distinguish between valid dates or times. Moreover, date/time mislabellings have been corrected.

In Figure 2 we can see the automatically annotated Information State[7] corresponding to the dialogue context after U3 (the actual system output is in XML, but we do not show it over here for reasons of readability). Note especially the confirmation of 'DEST_CITY' information in this move, and the history level of the annotation, which contains the sequences of speech acts and filled and confirmed slots for the entire dialogue.

In order to further explain how we compute confirmation, consider a variation of the above example. Generally, in common types of dialogue, after a system request for confirmation the user may decide to (1) ignore this request and proceed to provide new information (especially after system requests for implicit confirmation); (2) proceed to ask for clarification or help, request repetition, etc.; or (3) accept or reject the system request. Imagine that in U3 the user does not give the departure date (which would be case 1) but instead only replies to the confirmation prompt about the destination city (S4), i.e. chooses to reply to the system request for confirmation (case 3). In the COMMUNICATOR corpus we have observed six general ways the user accepts or rejects a system confirmation request[8]: YES-CLASS, e.g. 'yes'; NO-CLASS, e.g. 'no'; YES-CLASS, CITY, e.g. 'yes, Orlando'; NO-CLASS, CITY, e.g. 'no, Boston'; NO-CLASS, CITY, CITY, e.g. 'not Orlando, Boston'; or CITY, e.g. 'Orlando.'

In the first five cases it is easy for the annotation system to infer that there is positive or negative confirmation and thus confirm the slot or not accordingly because of the appearance of 'YES-CLASS' or 'NO-CLASS.' However, in the last case the annotation system should compare the user's utterance with the previous system's prompt for confirmation in order to decide whether the slot should be confirmed or not. If the user says 'Orlando' he/she *re-provides* information and the slot 'DEST_CITY' is confirmed whereas if the user utters 'Boston' he/she corrects the system (CORRECT_INFO), which means that the slot 'DEST_CITY' is not confirmed and therefore its current value will be removed. In the 'NO-CLASS, CITY, CITY' case the user rejects the value of the slot and corrects it at the same time. These are examples of the patterns used to compute confirmation.

5 Specific methods required for the COMMUNICATOR data

Up to now, all the methods we have presented are generally applicable for the case of annotating human–machine dialogue corpora for limited-domain information-seeking applications. In this section, for the sake of completeness, we note the particular methods that we used to deal with the peculiarities of the COMMUNICATOR data.

[7] Items appearing between [ ] brackets are user inputs (sometimes not annotated) and other items are system actions.

[8] The 'YES-CLASS' corresponds to words or expressions like 'yes,' 'okay,' 'right,' 'correct,' etc. In the same way 'NO-CLASS' stands for 'no,' 'wrong,' and so on.

5.1 *Specific parsing rules for the* COMMUNICATOR *data*

As discussed in Section 2.3 when the annotation system parses the user input it has to decide whether the information provided refers to a single or a multiple-leg trip. For a continuation trip, user input parsed as DEST_CITY, DEPART_DATE, DEPART_TIME, ARRIVE_DATE and ARRIVE_TIME will be tagged as CONTINUE_DEST_CITY, CONTINUE_DEPART_DATE, CONTINUE_DEPART_TIME, CONTINUE_ARRIVE_DATE, and CONTINUE_ARRIVE_TIME. In the same way, for a return trip, DEPART_DATE, DEPART_TIME, ARRIVE_DATE, and ARRIVE_TIME will be tagged as RETURN_DEPART_DATE, RETURN_DEPART_ TIME, RETURN_ARRIVE_DATE, and RETURN_ARRIVE_TIME. For a continuation trip the origin city of one leg is the destination city of the previous leg. For a return trip, the origin city of the inward leg is the destination city of the outward leg and the destination city of the inward leg is the origin city of the outward leg. Thus we do not need tags such as CONTINUE_ORIG_CITY, RETURN_ORIG_CITY, RETURN_DEST_CITY.

5.2 *Confidence scoring for the* COMMUNICATOR *data*

Ideally, in future dialogue corpora, we would have dialogue data that contains ASR confidence scores. Unfortunately the COMMUNICATOR data does not have this information. However, the COMMUNICATOR data contains both the output of the speech recognition engine for a user utterance and a manual transcription of the same utterance carried out by a human annotator. We consider the word error rate (WER) to be strongly related to confidence scores and thus each time a user utterance is read from the XML file a third agent is called to estimate error rates (the *ComputeErrorRates* agent). Four different error rates are estimated: classic WER, WER-noins, sentence error rate (SER), and keyword error rate (KER).

The classic WER, is defined by the following formula:

$$\mathrm{WER} = 100\left(\frac{Nins + Ndel + Nsub}{N}\right)\%$$

where N is the number of words in the transcribed utterance, and *Nins*, *Ndel*, *Nsub* are the number of insertions, deletions, and substitutions, respectively, in the speech recognition output. WER-noins is WER without taking into account insertions. The distinction between WER and WER-noins is made because WER shows the overall recognition accuracy whereas WER-noins shows the percentage of words correctly recognised. The sentence error rate (SER) is computed on the whole sentence, based on the principle that the speech recognition output is considered to be correct only if it is exactly the same as the manually transcribed utterance. All the above error estimations have been performed using the HResults tool of HTK (Young *et al.* 2005), which is called by the *ComputeErrorRates* agent. Finally the keyword error rate (KER) is also computed by *ComputeErrorRates* (after the utterance has been parsed by DIPPER) and shows the percentage of the correctly recognised keywords (cities, dates, times, etc.). This is also a very important metric regarding the efficiency of the dialogues. Similarly to WER, KER cannot be computed at runtime but we assume that it is strongly correlated with the confidence score of the parser.

It should be noted that speech phenomena such as pauses, fillers, noises, etc. that are transcribed by human annotators are not taken into account when error rates are estimated because most speech recognisers do not include them in their outputs, even though they are considered by their acoustic models. Therefore if such phenomena were included while estimating errors there would not be a fair comparison between the speech recognition output and the human transcription of the utterance.

6 Evaluating the automatic annotations

We pursued two types of evaluations of the automatically annotated data. First, automatic evaluation using the task completion metrics in the corpus, and second, comparison with human hand annotation of the same corpus.

We also developed a baseline automatic annotation system that tags a user utterance with the ⟨speech act, task⟩ pair that best matches the previous system request. Thus, if the previous system utterance is tagged as 'REQUEST_INFO(ORIG_CITY)' the baseline system will tag the user utterance as 'PROVIDE_INFO(ORIG_CITY)' regardless of what the user actually said. Confirmed slots are computed in the same way as the automatic annotation system. Also after system utterances tagged as 'EXPLICIT_CONFIRM' if the user says 'yes' the user utterance will be tagged as 'yes_answer' and the task will depend on the previous system prompt, e.g. after 'EXPLICIT_CONFIRM(ORIG_DEST_CITY)' a 'yes' answer will be tagged as 'YES_ANSWER(ORIG_CITY), YES_ANSWER(DEST_CITY).' Similarly, after a system action tagged as 'EXPLICIT_CONFIRM(TRIP)' the baseline system will parse the system utterance to infer the tasks associated with the 'YES_ANSWER' speech act of the user. The same applies to 'no' answers. This forms a strong baseline since 79 per cent of the time in the corpus users tend to reply to system prompts by providing exactly the information requested by the system. Obviously, a majority class baseline system (i.e. a baseline that always produces the most frequent ⟨speech act, task⟩ pair) or a random baseline system (i.e. a baseline that randomly generates ⟨speech act, task⟩ pairs) would be much weaker than our baseline. As will be shown in the sequel, our baseline system did not perform as well as the advanced automatic annotation system that we developed.[9]

6.1 Automatic evaluation

We evaluated our automatic annotation system by automatically comparing its output with the actual (ATC) and perceived (PTC) task completion metrics as recorded in the original COMMUNICATOR corpus. Our evaluation is restricted in the 2001 corpus because no such metrics are available for the 2000 data collection. Systems in the 2001 collection were generally more complex and there was a large performance improvement in every core metric from 2000 to 2001 (Walker *et al.* 2002). Therefore if the 2001 annotations are evaluated as successful, it is expected

[9] Both systems are automatic but from now on and for clarity we will refer to the advanced one as 'automatic annotation system' and to the baseline one as 'baseline system.'

that the same will be true for the 2000 annotations. If the final state of a dialogue – that is, the information about the filled and confirmed slots – agrees with the ATC and PTC for the same dialogue, this indicates that the annotation is consistent with the task completion metrics. We consider only dialogues where the tasks have been completed successfully – in these dialogues we know that all slots have been correctly filled and confirmed[10] and thus the evaluation process is simple to automate.[11] For example, if a dialogue that is marked as successful consists of a single leg and a return leg then the expected number of slots to be filled and confirmed is six ('ORIG_CITY,' 'DEST_CITY,' 'DEPART_DATE,' 'DEPART_TIME,' 'RETURN_DEPART_DATE,' and 'RETURN_DEPART_TIME'). If the automatic annotation system filled three and confirmed two slots then the accuracy for this particular dialogue for filled slots and confirmed slots would be 50 per cent and 33.3 per cent, respectively. For dialogues that are marked as unsuccessful or not marked at all (as in the 2000 corpus) there is no straightforward way to calculate the number of slots that should have been filled or confirmed. This automatic evaluation method cannot give us exact results – it only indicates whether the dialogue is annotated more or less correctly.

We have applied our automatic evaluation method on the flight-booking portions of the automatically annotated COMMUNICATOR corpus. The results are that, for dialogues where ATC or PTC is marked as '1' or '2' (i.e. where the flight-booking portion of the dialogue was successful or was perceived by the user to be successful),[12] the current automatic annotations for the whole corpus showed 93.9 per cent of the required slots to be filled (filled slots accuracy) and 75.2 per cent of the slots to be confirmed (confirmed slots accuracy). For the baseline annotations the accuracy for the filled and confirmed slots was 65.4 per cent and 50.9 per cent, respectively. Therefore when we use the advanced automatic annotations the absolute increase in accuracy of the filled and confirmed slots over the baseline annotations is 28.5 per cent and 24.3 per cent, respectively. Detailed results are depicted in Table 2.

The IBM system did not confirm and therefore we could not obtain results for the 'confirmed slots accuracy.' In cases where the system attempts to confirm more than one slot in a single turn (second and third confirmation strategies), if the user gives a simple 'NO_ANSWER' there is no way for the annotation system to detect the slot that the 'NO_ANSWER' refers to. The system assumes that the 'NO_ANSWER' refers to all the slots under confirmation. This can lead to fewer slots being confirmed. One of the rules that the annotation system used for confirmation calculation in the version described in Georgila, Lemon and Henderson (2005b) was that only filled

[10] Error analysis showed that this assumption that the successful dialogues had all slots confirmed (not just filled) is too strong.

[11] Note that the only reason we do not include unsuccessful dialogues in our automatic evaluation is that there is no way to automatically calculate the expected number of filled or confirmed slots. The uncompleted dialogues are not necessarily more difficult to annotate. It is often the case that successfully completed dialogues are very complex, e.g. there are several misunderstandings and the dialogue flow later becomes smooth after several user requests for restarting the dialogue.

[12] ATC is marked as '1' for actually completed dialogues and '0' otherwise. PTC is marked as '1' for dialogues in which only the air requirements were perceived as completed, '2' for dialogues in which both the air and ground requirements were perceived as completed and '0' otherwise.

Table 2. *Automatic ISU annotation accuracy for the* COMMUNICATOR *2001 data (automatic evaluation, AUTO: automatic annotation system, BASE: baseline system)*

System name	Total number of dialogues	Number of evaluated dialogues (ATC or PTC = 1 or 2)	Filled slots accuracy (%)	Confirmed slots accuracy (%)
			AUTO/BASE	AUTO/BASE
ATT	258	123	94.2/66.5	62.6/38.0
BBN	162	131	89.2/63.1	83.5/58.9
CMU	151	121	85.5/68.7	72.6/56.0
COL	217	152	97.8/63.3	66.6/35.0
IBM	240	166	95.4/58.1	NA/NA
LUC	214	128	98.6/76.6	84.4/68.0
MIT	213	162	96.8/54.8	79.4/39.4
SRI	228	99	92.1/84.2	81.2/76.9
ALL	1683	1082	93.9/65.4	75.2/50.9

slots could be confirmed, mostly to ensure that the system policies trained with the COMMUNICATOR annotated corpus (e.g. using Reinforcement Learning) would be reasonable. This rule caused problems in cases where for example the system knew the user's residence and therefore did not ask for the 'ORIG_CITY' but in the sequel tried to confirm it, or when the user gave a negative confirmation to a filled slot value (thus the filled slot was emptied) but the system performed a second confirmation request with an alternative slot value. In that case even if the user gave a 'YES_ANSWER' the slot would not be confirmed because it was not filled anymore. The above observations explain the low scores of the COL and ATT systems (and to a lesser extent CMU) for 'confirmed slots accuracy' in the earlier work of Georgila *et al.* (2005b). In the current version of the automatic annotation system the rule that caused problems (of not being able to confirm a slot unless it had been filled first) has been removed and the COL and CMU scores for confirmed slots accuracy have risen significantly (especially COL). The results presented in Table 2 show that the baseline annotations are generally poor and that the user does not always behave as expected. Therefore user utterances should be properly parsed in their context instead of just being tagged with the ⟨speech act, task⟩ pair that best matches the previous system request. In particular, IBM, MIT, and also COL had very general opening questions in their dialogues, e.g. 'What are your travel plans?' or 'Please tell me about the first leg of the trip.' that prompted the user to provide much information at the same time, which was beyond the capabilities of the baseline system.

6.2 Evaluation with respect to hand-annotated data

The results of the automatic evaluation tell us whether the system succeeded in filling and confirming the right slots but not whether their values were correct. By inspection of the annotated COMMUNICATOR dialogues it appears that most of

the time slots are filled and confirmed with the correct values or they are not filled/confirmed at all. However, in order to have more accurate results, evaluation with respect to hand-annotated data was also performed. We randomly selected six dialogues from each system of the 2001 collection, making forty-eight dialogues in all for hand annotation. There were no constraints regarding whether the dialogues were actually completed or perceived by the user as completed. The only constraint was not to have more than one dialogue from the same user so that more variety was captured.

We extracted the final information state of the flight-booking portion of the automatically annotated dialogues and examined the 'FilledSlotsStatus,' 'FilledSlotsValuesStatus,' and 'ConfirmedSlotsStatus' fields. As mentioned in Section 2.4, these fields give us information about the current status of the slots, i.e. whether they are filled or confirmed, and their values. For example, if 'ORIG_CITY' is filled with the value 'Boston' and then with the value 'Orlando,' only the value 'Orlando' will be saved in the 'FilledSlotsValuesStatus' field. Then we manually inspected the flight-booking portion of the dialogue from beginning to end and we noted down the slots that should be filled or confirmed and their values, according to what happened in the dialogue and with respect to the ASR hypotheses, rather than the utterance transcriptions. By comparing the status of the slots as produced by our automatic annotation system with the status that the slots should have according to the manual annotation, we can measure the precision, the recall, and the F-score of our automatic annotations.

Precision and recall are defined by the following formulae:

$$\text{Precision} = \frac{C}{C + I1 + I2}$$

$$\text{Recall} = \frac{C}{C + I1 + MC}$$

$$F\text{-score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

where C is the number of correctly tagged slots, $I1$ is the number of incorrectly tagged slots that should have been tagged with a different value, $I2$ is the number of incorrectly tagged slots that should not have been tagged at all, and MC is the number of slots that were missed by the automatic annotation system or the baseline system (i.e. not tagged at all). Using the above formulae, precision, recall, and consequently F-score can be computed for filled slots and confirmed slots.

In our manual annotation a slot is considered as correctly filled if it is filled with the correct value. Moreover, a slot is considered as confirmed only if the system attempted to confirm it. It will not be considered as confirmed just because the user accepted the flight offer. This is in order to have a fair comparison with our automatic annotations. Both the automatic annotation system and the baseline system tag a slot as confirmed only after the dialogue system has explicitly or implicitly tried to confirm that particular slot. In addition, a slot is confirmed if its value in the 'FilledSlotValue' field has been confirmed.

Table 3. *Precision, recall, and F-score of filled slots (including their values) for the* COMMUNICATOR *2001 data (manual evaluation, AUTO: automatic annotation system, BASE: baseline system)*

System name	Precision (%)	Recall (%)	*F*-score (%)
	AUTO/BASE	AUTO/BASE	AUTO/BASE
ATT	88.9/92.6	86.5/67.6	87.7/78.1
BBN	93.3/95.5	82.3/61.8	87.5/75.0
CMU	91.7/100	86.8/81.6	89.2/89.9
COL	95.6/93.8	95.5/68.2	95.5/79.0
IBM	82.4/86.1	90.4/67.4	86.2/75.6
LUC	46.1/36.8	34.3/20.0	39.3/25.9
MIT	92.1/92.3	87.5/60.0	89.7/72.7
SRI	88.6/89.3	81.6/69.5	85.0/78.2
ALL	85.8/87.8	81.8/62.6	83.8/73.1
ALL (except LUC)	89.5/92.6	87.6/68.0	88.5/78.4

Table 4. *Precision, recall, and F-score of confirmed slots (including their values) for the* COMMUNICATOR *2001 data (manual evaluation, AUTO: automatic annotation system, BASE: baseline system)*

System name	Precision (%)	Recall (%)	*F*-score (%)
	AUTO/BASE	AUTO/BASE	AUTO/BASE
ATT	95.7/100	81.5/55.6	88.0/71.5
BBN	95.8/100	67.7/47.0	79.3/63.9
CMU	93.3/95.5	80.0/62.9	86.1/75.8
COL	96.6/100	93.3/56.7	94.9/72.4
IBM	90.0/83.3	81.8/50.0	85.7/62.5
LUC	50.0/40.0	34.5/20.7	40.8/27.3
MIT	90.3/94.7	75.7/48.6	82.4/64.2
SRI	96.0/100	85.7/77.8	90.6/87.5
ALL	89.6/90.8	74.5/52.0	81.4/66.1
ALL (except LUC)	94.2/97.4	80.2/56.5	86.6/71.5

Detailed results of precision, recall, and *F*-score for filled and confirmed slots are given in Tables 3 and 4, respectively.

The results for precision, recall, and *F*-score for the automatic annotation system are relatively high for all systems apart from LUC. For ATT, CMU, and COL the scores computed for confirmed slots from the evaluation with regard to the hand-annotated data are much higher than the automatically derived scores of Section 6.1, which proves that the assumption that the successful dialogues had all slots confirmed is too strong. Precision of confirmed slots is much higher than

Table 5. *Precision and recall of ⟨speech act, task⟩ pairs for the* COMMUNICATOR *2001 data (manual evaluation, AUTO: automatic annotation system, BASE: baseline system)*

Source of error	Precision (%)	Recall (%)
	AUTO/BASE	AUTO/BASE
	70.5/64.0	76.4/64.6
(null-null)-(provide-null)	1.8/1.1	2.0/1.2
(null-null)-(yes-null or no-null)	1.7/2.2	1.9/2.2
(provide_info)-(reprovide_info)	2.4/0.7	2.6/0.7

recall, which means that the automatic annotation system decides to tag a slot as confirmed only if there is strong evidence for doing so.

As with the automatic evaluations reported earlier, the recall and the F-score of the baseline system for the confirmed slots are low for ATT, BBN, COL, IBM, MIT, and to a lesser extent CMU. The performance of the baseline for LUC is poor for the same reasons that the automatic annotation fails to produce good precision and recall values (details are given in Section 6.4). The precision of the baseline system is very high, some times even higher than the precision of the automatic annotation system, due to the fact that the baseline is very conservative in its predictions. However, the recall is always considerably lower than the recall of the automatic annotation system for both filled and confirmed slots.

By using the automatic annotations instead of the baseline ones the absolute gain in the F-score for filled and confirmed slots is approximately 10 per cent and 15 per cent, respectively, with or without including the poor results of LUC.

For these forty-eight dialogues we also manually tagged each user utterance with ⟨speech act, task⟩ pairs and used this as our gold standard with which we compared both the automatic and the baseline annotations. The results are depicted in Tables 5 and 6.

Table 5 shows that the precision and recall of the automatic annotation system for all ⟨speech act, task⟩ pairs are 70.5 per cent and 76.4 per cent, respectively. The precision is a little lower than the recall because the automatic annotation system had the tendency to overgenerate. For example if the ASR output was 'yes I'd no I'm travelling to London' the automatic annotation system would tag it as '[YES_ANSWER(NULL), NO_ANSWER(NULL), PROVIDE_INFO(DEST_CITY)]' instead of '[PROVIDE_INFO(DEST_CITY)].' This is due to the fact that the keyword-based parser cannot handle speech repairs. Each row in Table 5 gives the gain we can get in precision and recall by ignoring some unimportant sources of error. Thus, if we ignore cases that are tagged with empty values for both the speech act and the task by the system but the human annotators tag them as 'PROVIDE_INFO(NULL)' or the opposite, precision rises to 72.3 per cent and recall to 78.4 per cent. Also, both the automatic annotation system and the baseline system tag user utterances which contain the words 'yes' or 'no' as 'yes_answer' or 'no_answer,' respectively even though the utterance is not necessarily an answer. That of course does not cause

Table 6. *Precision, recall, and F-score per slot type for the* COMMUNICATOR *2001 data (manual evaluation, AUTO: automatic annotation system, BASE: baseline system)*

Slot type	Precision (%)	Recall (%)	*F*-score (%)
	AUTO/BASE	AUTO/BASE	AUTO/BASE
orig_city	68.9/69.0	84.3/56.1	75.8/61.9
dest_city	74.8/73.7	79.2/58.3	76.9/65.1
depart_date	74.3/72.3	82.1/64.5	78.0/68.2
depart_time	63.9/71.7	67.1/54.4	65.5/61.9
arrive_time	71.4/75.0	23.8/14.3	35.7/24.0
continue_dest_city	69.7/70.2	65.3/62.1	67.4/65.9
continue_depart_date	52.0/53.3	66.7/61.5	58.4/57.1
continue_depart_time	79.2/87.5	67.9/50.0	73.1/63.6
return_depart_date	82.0/87.2	83.7/83.7	82.8/85.4
return_depart_time	82.1/86.4	67.7/79.2	74.2/82.6
airline	95.0/71.4	70.4/18.5	80.9/29.4

problems with the status of the filled and confirmed slots, and if it is ignored the precision of the automatic annotation system rises from 70.5 per cent to 72.2 per cent and the recall from 76.4 per cent to 78.3 per cent. If the distinction between 'provide_info' and 'reprovide_info' is ignored the gains for precision and recall are 2.4 per cent and 2.6 per cent, respectively. As we can see the results for the baseline are consistently lower. By ignoring all of the three above-mentioned sources of error the best results that can be obtained are 76.4 per cent and 82.9 per cent for the automatic annotation system (precision and recall) and 68.0 per cent and 68.7 per cent for the baseline system. The initial *F*-scores for the automatic annotation system and the baseline system are 73.3 per cent and 64.3 per cent, respectively and the best *F*-scores we can obtain by ignoring the three sources of error are 79.5 per cent and 68.3 per cent. All the values in Table 5 have been calculated without taking into account cases where both the human annotators and the systems tagged the user utterance with empty lists for both the speech act and the task, i.e. when they agreed that the utterance did not contain relevant information to the dialogue task. If we want to give some credit to the automatic annotation system and the baseline system for successfully detecting such cases then the results for precision, recall, and *F*-score become for the automatic annotation system 71.0 per cent, 76.3 per cent, and 73.5 per cent and for the baseline 65.7 per cent, 66.2 per cent, and 66.0 per cent. The improvement becomes higher if we also ignore the problems of distinguishing between 'provide_info' and 'reprovide_info,' etc.

Table 6 shows precision, recall, and *F*-score per slot type. Again the automatic annotation system outperforms the baseline one. The recall for 'ARRIVE_TIME' (both in the automatic and baseline annotations) and for 'AIRLINE' (only in the baseline annotations) is very low because there are not many instances of these slots and thus the effect of each error on the total score is inflated. Furthermore, the 'AIRLINE' slot is optional and the dialogue systems rarely prompt the user to provide this slot, which explains the low recall of the baseline.

Manual annotation has also shown that the hotel and car rental bookings have been well annotated.

6.3 Indirect evaluation via user simulations

As a form of indirect evaluation we build user simulation models. For learning dialogue strategies it is rarely (if ever) the case that enough training data from real dialogues with human users is available to sufficiently explore the vast space of possible dialogue states and strategies. Thus, simulated users are critical for training stochastic dialogue systems (often of the order of 100,000 training dialogues are required), and for evaluating candidate dialogue policies. Simulated users simulate real-user behaviour, i.e. the way users would interact with the system in order to accomplish their goals (e.g. book a flight, get tourist or product information, etc.). The basic idea is to use small corpora to train stochastic models for simulating real-user behaviour. Once such a simulated user is available, any number of dialogues can be generated through interaction between the simulated user and the dialogue policy, and these simulated dialogues can be used with statistical optimisation methods such as RL (Georgila *et al.* 2005a, 2006, 2008b; Schatzmann *et al.* 2005a). Also user simulations make it feasible to test the performance of different dialogue policies against simulated users in an efficient and inexpensive way.

We train n-grams of ⟨speech act, task⟩ pairs from both the automatic system annotations and the baseline annotations. Our reasonable assumption is that the higher the quality of the user simulation model the better the training data it was trained on. To assess our user models we need to compare their output with real responses given by users in the same context. For this purpose we run our user simulation models through the ⟨speech act, task⟩ pairs of the hand-annotated data.[13] As explained previously, user simulations are to be used for automatic dialogue strategy learning. For this reason we desire user simulations which do not always act in the same way in identical states – we need to explore the policy space, so some reasonable amount of variation is required. Our n-gram models produce actions based on a probability distribution learnt from the training data and not always the action with the highest probability. Therefore our evaluation metrics should take into account this distribution. We use expected precision (EP) and expected recall (ER) to quantify how closely the synthetic turn resembles the real-user turn. Unlike precision and recall that are calculated always choosing the action with the highest probability (Schatzmann *et al.* 2005a), EP and ER are calculated taking into account the probability distribution of actions that the user model can produce at a given state. Thus we measure the expected value of precision and recall if we ran the user model for a large number of times through the same states found in the data, each time generating an action according to the distribution produced by the model. We also use perplexity (PP) as a measure of the number of possible actions that the user model has to choose between at a given state. For a full discussion on the above metrics see Georgila *et al.* (2005a, 2006) and

[13] The hand-annotated data was not enough for building user simulation models.

Table 7. *Expected precision, expected recall, and perplexity obtained when n-gram user simulation models trained on the automatic and baseline annotations run through the hand-annotated data (AUTO: user simulations trained on the automatic annotations, BASE: user simulations trained on the baseline annotations)*

	Expected precision (%)	Expected recall (%)	Perplexity
	AUTO/BASE	AUTO/BASE	AUTO/BASE
5-gram	49.3/45.7	49.2/42.9	10.0/14.4
4-gram	47.3/45.2	47.5/42.5	8.9/14.2
3-gram	40.6/40.1	41.0/37.7	8.6/16.8
2-gram	27.4/28.5	27.8/26.8	10.3/21.7

Schatzmann *et al.* (2005a, 2006). Results are depicted in Table 7. The user model based on the automatic annotations performs better than the one trained on the baseline annotations, which once more proves that the automatic annotation system is superior to the baseline one. The automatic annotations are especially better with regard to expected recall and for higher values of *n*. For low values of *n*, the baseline user model suffers less from data sparsity issues since the baseline annotations do not produce the variety of the automatic ones. With regard to perplexity again the user model trained on the automatic annotations performs better.

Another indirect indication of the good quality of the annotations and proof of their usefulness is that they have been used to train successful dialogue strategies (Henderson *et al.* 2005, 2008; Frampton and Lemon 2006) and user simulations in Georgila *et al.* (2005a, 2006) and Schatzmann *et al.* (2005a, 2005b).

Furthermore, in Lemon *et al.* (2006a) we have demonstrated that a policy learnt from our COMMUNICATOR 2001 automatically annotated dialogues performs better than a state-of-the-art hand-coded dialogue system in experiments with real users. The experiments were done using the 'TownInfo' multimodal dialogue system of Lemon *et al.* (2006b). The learnt policy (trained on the COMMUNICATOR data) was ported to the city information domain, and then evaluated with human subjects. The learnt policy achieved an average gain in perceived task completion of 14.2 per cent (from 67.6 per cent to 81.8 per cent at $p < .03$) compared to a state-of-the-art hand-coded system.

6.4 Additional error analysis

We now explore some additional patterns of error in the automatic annotations, which are specific to the flight reservations domain. In all systems there are system prompts asking the user whether he/she would like to have a continuation or return trip. Unfortunately the DATE tag 'CONTINUE_TRIP' or 'RETURN_TRIP,' respectively, is the same regardless of whether the user chooses to have a continuation/return trip or not. The automatic annotation system has to parse the system prompt and the user reply and decide on the type of trip. For all systems except LUC, these system prompts always have a consistent form. For example, for continuation trips ATT

Table 8. *Precision, recall, and F-score for filled and confirmed slots (including their values) and ⟨speech act, task⟩ pairs, automatic annotation of the TALK corpus*

	Precision (%)	Recall (%)	*F*-score
Filled slots	88.4	98.7	93.3
Confirmed slots	92.0	89.2	90.6
⟨speech act, task⟩ pairs	80.4	82.3	81.3

always uses the following type of question 'Is Detroit your final destination?' Thus if the user replies 'yes' there will be no continuation trip. On the other hand, for CMU the structure of the question is different: 'Would you like to go to another city after Los Angeles?' Here 'yes' means the opposite, that is, there will be a continuation trip. LUC switches between the two types of questions, which means that if the annotation system fails to parse the system output there is no default solution to follow as in the other systems. In addition, in the LUC dialogues sometimes there is no question about continuation or return trips and the annotation system has to deduce that from the context, which makes the annotation task even harder. As a result, sometimes the new values from continuation or return legs overwrite the initial values of the outward trip. This mistake was not captured by the automatic evaluation because the annotation system did not recognise the existence of a continuation/return trip, and thus the number of slots that it computed as expected to be filled or confirmed did not include continuation and return slots.

7 Porting to other domains and applications

The current implementation is generic enough to be easily ported from the flight-reservations domain to other slot-filling applications in different domains. As a proof of concept we modified the automatic annotation system in order to automatically annotate the TALK city information corpus. The TALK corpus consists of human–machine dialogues in the city-information domain collected by having real users interact with the TALK 'TownInfo' dialogue system (Lemon *et al.* 2006a, 2006b). Users are asked to find a hotel, bar, or restaurant in a town. The dialogue system is mixed-initiative, i.e. users are allowed to provide information about multiple slots at the same time or provide information different from what the system has requested.

We compare the annotations produced by the modified automatic annotation system with the log files of the 'TownInfo' dialogue system. Table 8 shows precision, recall, and *F*-score for filled and confirmed slots and for ⟨speech act, task⟩ pairs. For filled and confirmed slots we take into account not only whether the correct slot has been filled or confirmed but also whether its value is correct or not. The values for ⟨speech act, task⟩ pairs have been calculated without taking into account cases that are tagged with empty lists both in the automatic annotations and the dialogue system logs, i.e. when the utterance did not contain relevant information to the dialogue task. If we want to give some credit to the automatic annotations for

detecting such cases (as in Section 6.2), the results for precision, recall, and *F*-score become 84.7 per cent, 86.2 per cent, and 85.4 per cent, respectively.

The results are good and prove that the automatic annotation system can be easily ported to other slot-filling tasks. In particular, it took one day to modify the automatic annotation system and run the evaluation tests. All values are high, even higher than the scores of the automatic annotations for the COMMUNICATOR corpus because our city information domain required fewer slots to be filled and confirmed and did not have cases similar to multiple leg or return trips that accounted for a high percentage of the errors in the COMMUNICATOR automatic annotations.

The question also arises as to what extent our automatic annotation system is portable to other types or genres of dialogue, such as tutorial dialogue (Zinn, Moore and Core 2002; Litman and Fobes-Riley 2006). Obviously the structure of tutorial dialogues is very different from the structure of task-oriented applications. Therefore to process tutorial dialogues the Information States and the annotation tool would have to be extensively modified.

Depending on the tutoring style (e.g. didactic versus Socratic), the automatic annotation task could be easier or harder. Core, Moore and Zinn (2003) hypothesised that didactic tutoring corresponds to system-initiative dialogue management whereas Socratic tutoring is related to mixed-initiative dialogue strategies. However, their findings from analysing a corpus in the domain of basic electricity and electronics showed that the opposite was true, i.e. students had initiative more of the time in the didactic dialogues than in the Socratic dialogues. Furthermore, Socratic dialogues were more interactive than didactic dialogues. The students produced a higher percentage of words in Socratic dialogues whereas tutor turns and utterances were shorter. On the contrary, in didactic dialogues tutors tended to give longer explanations. It is hard to say which type of tutoring would be easier for an automatic annotation tool to deal with. Again the success of the automatic annotations would depend heavily on the parser employed. In didactic dialogues with long tutor explanations having the tutor's part already annotated would certainly help. With respect to Socratic dialogues and given that the tutor's part was already tagged with speech act information, it would be easier to interpret the student input in cases where the tutor asked sequences of targeted questions with strong expectations about plausible answers (Core *et al.* 2003).

In terms of multimodal dialogue systems where the user can provide input to the system through different modalities at the same time (e.g. gestures and speech), much of the success of the automatic annotation tool will depend on the parser and its ability to process and combine asynchronous information from different input modalities. The Information States would have to be updated with fields related to multimodality. For slot-filling multimodal dialogue systems no other major changes would be required.

8 Discussion

We have shown that the automatic annotation system can be successfully ported to similar slot-filling applications with minimal effort. The current system has

automatically annotated two corpora in both of which the dialogue system speech act annotations were already available. However, the annotation tool is designed not to rely on the available annotations of the system's utterances but will also parse them to compensate for possible errors in the available annotations. This is also important for cases such as 'EXPLICIT_CONFIRM(TRIP)' where the annotation system has to parse the system utterance to infer the tasks associated with the general tag 'TRIP.' In addition, to decide on whether a user speech act should be 'PROVIDE_INFO' or 'REPROVIDE_INFO' the annotation tool also has to parse the system utterances (see Section 3.2). Thus the tool as it is currently implemented is appropriate also for annotating closed-domain human–human or human–machine slot-filling dialogues in which neither side of the dialogue is tagged with speech acts (levels 3 and 4 in Section 1.1) – including dialogues collected with Wizard-of-Oz experiments. As is the case with a dialogue system, the more complex the dialogue structure the more advanced the parser that is required. For the COMMUNICATOR corpora and the TALK corpus a keyword-based parser proved adequate, which would not be the case for human–human dialogues.

Recently, we also used the automatic tool to process dialogues collected from a Wizard-of-Oz experiment where both older and younger human users interacted with a human wizard to book an appointment at a hospital. The idea was that human annotators would be provided with the automatic annotations as a starting point and correct them instead of having to annotate the corpus from scratch (Georgila *et al.* 2008a). It took approximately three days to modify the automatic annotation tool to process this corpus (more than the time required for the corpus in the city information domain). This is due to the fact that dialogues were more complex than the COMMUNICATOR dialogues or the ones in the city information domain because older people produced longer dialogues than younger users, had a richer vocabulary, and used a larger variety of speech acts (Georgila *et al.* 2008a). Nevertheless, the automatic annotation system performed well and significantly accelerated the manual annotation process.[14]

Our annotation tool was not designed to handle open-domain dialogues, e.g. spontaneous dialogues between humans or dialogues between a human and a 'chatbot' that do not aim to accomplish a particular task. Chatbots are conversational agents designed to simulate open-domain conversations with humans. Instead of parsing and interpreting user inputs, and planning next dialogue acts so as to achieve domain goals, chatbots use shallow pattern-matching techniques to retrieve a next system move from a dataset. Chatbots therefore have no understanding of the contents of the user utterances or of what they are saying in response, or indeed of why they are saying it. Even if our system could sufficiently interpret (e.g. perhaps by using a wide-coverage semantic parser) the utterances of humans and chatbots, it would still be very challenging to maintain a well-structured Information State representing

[14] We cannot compute the actual gain in time by using the automatic annotation system to bootstrap the manual annotations compared to annotating manually from scratch because we did not follow the latter approach. However, based on the annotators' comments, having the automatic annotations to start with proved to be very helpful.

the conversants' beliefs, goals, etc. for open-domain dialogues. This constitutes a significant open problem for future research.

Regarding the resulting annotated corpus, it is certainly true that such corpora as COMMUNICATOR are likely to aid further research in automatically learning dialogue strategies, user simulations, and context-dependent interpretation and speech recognition, only for slot-filling applications rather than more complex types of dialogue. Indeed the COMMUNICATOR corpus only includes dialogues for slot-filling tasks of limited complexity. However, the COMMUNICATOR corpus is the largest publicly available corpus and there are currently no other large and richly annotated corpora available for more complex human–machine dialogues. In particular, for automatically learning dialogue strategies and user simulations it is clear that the type of the corpus used will influence the resulting models. One way to deal with this problem is to learn user simulation models that allow for some deviation from the user behaviour observed in the corpus and subsequently use these simulated user models to learn more complex dialogue strategies. On the other hand, learning dialogue strategies and user simulations is a difficult problem with several open research questions even for simple slot-filling tasks. The state space is very large (Henderson *et al.* 2005, 2008) and the required computations can easily become intractable. For example, Williams and Young (2005) and Schatzmann *et al.* (2007) use a 'summary-space' mapping technique to deal with intractability and their task is less complex than the COMMUNICATOR task. Therefore it is very important that we show that machine learning techniques work well for simple slot-filling tasks first, before we move to more complex applications where the problems of intractability will become even more severe.

9 Conclusions

We provide a general framework and tools for the automatic annotation of dialogue corpora with additional contextual and speech act information. These more richly annotated corpora are essential for emerging statistical learning techniques in dialogue systems research. We explained the concepts behind context and speech act annotation, and the general methods for converting a standard corpus (of utterance transcriptions) into a corpus with context representations.

We then focused on the automatic annotation system applied to the COMMUNICATOR corpus of human–machine flight booking dialogues. The original COMMUNICATOR data (2000 and 2001) is not sufficient for our wider purposes (of learning dialogue strategies and user simulations from a corpus, and context-sensitive interpretation and speech recognition) since it does not contain speech act annotations of user utterances or representations of dialogue context. We briefly reviewed the DATE annotation scheme, and our extensions to it. We then described the automatic annotation system which uses the DIPPER dialogue manager. This annotates user inputs with speech acts and creates dialogue 'Information State' context representations. We discussed confirmation strategies, presented examples, and evaluated our annotations with respect to the task completion metrics of the original corpus and in comparison to hand-annotated data.

Both automatic and manual evaluation results establish the good quality of the automatic annotations. More evidence of the adequacy of the automatic annotations is that the resulting data has been used to learn successful dialogue strategies (Henderson *et al.* 2005, 2008; Frampton and Lemon 2006; Lemon *et al.* 2006a), and high quality user simulations (Georgila *et al.* 2005a, 2006; Schatzmann *et al.* 2005a, 2005b). The final annotated corpus will be made publicly available, for use by other researchers.[15] We expect that the conceptual framework and the tools described here can be used to convert other existing dialogue corpora into rich, context-annotated dialogue corpora useful for statistical learning techniques. In future work we intend to make the required adjustments to the tool so that it can also be publicly distributed.

Acknowledgements

Georgila was supported by SHEFC HR04016 – Wellcome Trust VIP Award. Lemon and Henderson were partially supported by EPSRC grant number: EP/E019501/1. This work was partially funded by the EC under the FP6 project 'TALK: Talk and Look, Tools for Ambient Linguistic Knowledge' http://www.talk-project.org, IST-507802 and under the FP7 project 'CLASSiC: Computational Learning in Adaptive Systems for Spoken Conversation' http://www.classic-project.org, IST-216594. The TALK city information corpus collection was conducted in collaboration with Steve Young, Jost Schatzmann, Blaise Thomson, Karl Weilhammer, and Hui Ye at Cambridge University and their work is gratefully acknowledged. We would also like to thank the anonymous reviewers for their helpful comments.

References

Andreani, G., Di Fabbrizio, G., Gilbert, M., Gillick, D., Hakkani-Tür, D., and Lemon, O. 2006. Let's DiSCoH: collecting an annotated open corpus with dialogue acts and reward signals for natural language helpdesks. In *Proceedings of the IEEE/ACL Workshop on Spoken Language Technology*, Aruba, 2006, pp. 218–21.

Bos, J., Klein, E., Lemon, O., and Oka, T. 2003. DIPPER: description and formalisation of an Information-State Update dialogue system architecture. In *Proceedings of the 4th SIGdial Workshop on Discourse and Dialogue*, Sapporo, Japan, pp. 115–24.

Cheyer, A., and Martin, D. 2001. The open agent architecture. *Journal of Autonomous Agents and Multi-Agent Systems*, **4**(1/2): 143–8.

Clark, H. H., and Brennan, S. E. 1991. Grounding in communication. In L. Resnick, J. Levine, and S. Teasely (eds.), *Perspectives on Socially Shared Cognition*, pp. 127–49. American Psychological Association.

Core, M. G., Moore, J. D., and Zinn, C. 2003. The role of initiative in tutorial dialogue. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Budapest, Hungary, pp. 67–74.

[15] There is currently an example of a complete annotated dialogue available both in XML (http://homepages.inf.ed.ac.uk/kgeorgil/annot_comm_sample_dialogue.xml) and text (http://homepages.inf.ed.ac.uk/kgeorgil/annot_comm_sample_dialogue.txt) formats.

Frampton, M., and Lemon, O. 2006. Learning more effective dialogue strategies using limited dialogue move features. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics (ACL)*, Sydney, Australia, pp. 185–92.

Gabsdil, M., and Lemon, O. 2004. Combining acoustic and pragmatic features to predict recognition performance in spoken dialogue systems. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL)*, Barcelona, Spain, pp. 344–51.

Georgila, K., Henderson, J., and Lemon, O. 2005a. Learning user simulations for Information State Update dialogue systems. In *Proceedings of the 9th European Conference on Speech Communication and Technology (INTERSPEECH–EUROSPEECH)*, Lisbon, Portugal, pp. 893–6.

Georgila, K., Henderson, J., and Lemon, O. 2006. User simulation for spoken dialogue systems: learning and evaluation. In *Proceedings of the 9th International Conference on Spoken Language Processing (INTERSPEECH–ICSLP)*, Pittsburgh, PA, pp. 1065–68.

Georgila, K., Lemon, O., and Henderson, J. 2005b. Automatic annotation of COMMUNICATOR dialogue data for learning dialogue strategies and user simulations. In *Proceedings of the 9th Workshop on the Semantics and Pragmatics of Dialogue (SEMDIAL: DIALOR)*, Nancy, France, pp. 61–8.

Georgila, K., Wolters, M., Karaiskos, V., Kronenthal, M., Logie, R., Mayo, N., Moore, J. D., and Watson, M. 2008a. A fully annotated corpus for studying the effect of cognitive ageing on users' interactions with spoken dialogue systems. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, Marrakech, Morocco, pp. 938–44.

Georgila, K., Wolters, M., and Moore, J. D. 2008b. Simulating the behaviour of older versus younger users when interacting with spoken dialogue systems. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL–HLT)*, Columbus, OH, pp. 49–52.

Ginzburg, J. 1996. Dynamics and semantics of dialogue. In Jerry Seligman and Dag Westerstahl (eds.), *Logic, Language, and Computation*, Vol. 1. CSLI Publications, Stanford, CA.

Grosz, B. J., and Sidner, C. L. 1986. Attention, intentions, and the structure of discourse. *Computational Linguistics*, **12**(3): 175–204.

Henderson, J., Lemon, O., and Georgila, K. 2005. Hybrid reinforcement/supervised learning for dialogue policies from COMMUNICATOR data. In *Proceedings of the 4th Workshop on Knowledge and Reasoning in Practical Dialogue Systems, International Joint Conference on Artificial Intelligence (IJCAI)*, Edinburgh, UK, pp. 68–75.

Henderson, J., Lemon, O., and Georgila, K. 2008. Hybrid reinforcement/supervised learning of dialogue policies from fixed datasets. *Computational Linguistics* **34**(4): 487–511.

Keizer, S., and op den Akker, R. 2007. Dialogue act recognition under uncertainty using Bayesian networks. *Journal of Natural Language Engineering* **13**(4): 287–316.

Kipp, M. 1998 The neural path to dialogue acts. In *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI)*, Brighton, UK, pp. 175–9.

Larsson, S., and Traum, D. 2000. Information state and dialogue management in the TRINDI Dialogue Move Engine Toolkit. *Journal of Natural Language Engineering* **6**(3–4): 323–40.

Lemon, O., Georgila, K., and Henderson, J. 2006a. Evaluating effectiveness and portability of reinforcement learned dialogue strategies with real users: the TALK TownInfo evaluation. In *Proceedings of the IEEE/ACL Workshop on Spoken Language Technology*, Aruba, pp. 178–81.

Lemon, O., Georgila, K., Henderson, J., and Stuttle, M. 2006b. An ISU dialogue system exhibiting reinforcement learning of dialogue policies: generic slot-filling in the TALK in-car system. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Trento, Italy, pp. 119–22.

Lemon, O., and Gruenstein, A. 2003. Multithreaded context for robust conversational interfaces: context-sensitive speech recognition and interpretation of corrective fragments. *ACM Transactions on Computer–Human Interaction (ACM TOCHI)*, **11**(3): 241–67.

Lesch, S., Kleinbauer, T., and Alexandersson, J. 2005. A new metric for the evaluation of dialog act classification. In *Proceedings of the 9th Workshop on the Semantics and Pragmatics of Dialogue (SEMDIAL: DIALOR)*, Nancy, France, pp. 143–6.

Levin, E., Pieraccini, R., and Eckert, W. 2000. A stochastic model of human–machine interaction for learning dialog strategies. *IEEE Transactions on Speech and Audio Processing* **1**: 11–23.

Litman, D., and Forbes-Riley, K. 2006. Correlations between dialogue acts and learning in spoken tutoring dialogues. *Journal of Natural Language Engineering* **12**(2): 161–76.

Poesio, M., Cooper, R., Larsson, S., Matheson, C., and Traum, D. 1999. Annotating conversations for information state update. In *Proceedings of the 3rd Workshop on the Semantics and Pragmatics of Dialogue (SEMDIAL: AMSTELOGUE)*, Amsterdam, Netherlands.

Reithinger, N., and Klesen, M. 1997. Dialogua act classification using language models. In *Proceedings of the 5th European Conference on Speech Communication and Technology (EUROSPEECH)*, Rhodes, Greece, pp. 2235–8.

Reithinger, N., and Maier, E. 1995. Utilizing statistical dialogue act processing in VERBMOBIL. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL)*, Cambridge, MA, pp. 116–21.

Ries, K. 1999. HMM and neural network based speech act detection. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Phoenix, AZ, pp. 497–500.

Rieser, V., Kruijff-Korbayová, I., and Lemon, O. 2005a. A corpus collection and annotation framework for learning multimodal clarification strategies. In *Proceedings of the 6th SIGdial Workshop on Discourse and Dialogue*, Lisbon, Portugal, pp. 97–106.

Rieser, V., Kruijff-Korbayová, I., and Lemon, O. 2005b. A framework for learning multimodal clarification strategies. In *Proceedings of the 7th International Conference on Multimodal Interfaces (ICMI)*, Trento, Italy.

Rieser, V., and Lemon, O. 2006. Using machine learning to explore human multimodal clarification strategies. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics (ACL)*, Sydney, Australia, pp. 659–66.

Rieser, V., and Lemon, O. 2008. Learning effective multimodal dialogue strategies from Wizard-of-Oz data: Bootstrapping and evaluation. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL–HLT)*, Columbus, OH, pp. 638–46.

Samuel, K., Carberry S., and Vijay-Shanker, K. 1998. Dialogue act tagging with transformation-based learning. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (ACL–COLING)*, Montreal, Quebec, Canada, pp. 1150–6.

Schatzmann, J., Georgila, K., and Young, S. 2005a Quantitative evaluation of user simulation techniques for spoken dialogue systems. In *Proceedings of the 6th SIGdial Workshop on Discourse and Dialogue*, Lisbon, Portugal, pp. 45–54.

Schatzmann, J., Stuttle, M. N., Weilhammer, K., and Young, S. 2005b. Effects of the user model on simulation-based learning of dialogue strategies. In *Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop*, San Juan, Puerto Rico, pp. 220–5.

Schatzmann, J., Thomson, B., and Young, S. 2007. Statistical user simulation with a hidden agenda. In *Proceedings of the 8th SIGdial Workshop on Discourse and Dialogue*, Antwerp, Belgium, pp. 273–82.

Schatzmann, J., Weilhammer, K., Stuttle, N., and Young, S. 2006. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *Knowledge Engineering Review* **21**(2): 97–126.

Scheffler, K., and Young, S. 2001. Corpus-based dialogue simulation for automatic strategy learning and evaluation. In *Proceedings of the Workshop on Adaptation in*

Dialogue Systems, North American Chapter of the Association for Computational Linguistics (NAACL), Pittsburgh, PA, pp. 64–70.

Searle, J. 1969. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press: Cambridge, UK.

Singh, S., Kearns, M., Litman, D., and Walker, M. 1999. Reinforcement learning for spoken dialogue systems. *Advances in Neural Information Processing Systems* **12**: 956–62.

Stolcke, A., Coccaro, N., Bates, R., Taylor, P., Ess-Dykema, C. V., Ries, K., Shriberg, E., Jurafsky, D., Martin, R., and Meteer, M. 2000. Dialogue act modelling for automatic tagging and recognition of conversational speech. *Computational Linguistics* **26**(3): 339–74.

Traum, D. 1994. A computational theory of grounding in natural language conversation. *PhD Thesis*, Department of Computer Science, University of Rochester.

Traum, D. 2000. Twenty questions for dialogue act taxonomies. *Journal of Semantics* **17**(1): 7–30.

Traum, D. R., and Allen, J. 1994. Discourse obligations in dialogue processing. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, Las Cruces, NM, pp. 1–8.

Walker, M., Aberdeen, J., Boland, J., Bratt, E., Garofolo, J., Hirschman, L., Le, A., Lee, S., Narayanan, S., Papineni, K., Pellom, B., Polifroni, J., Potamianos, A., Prabhu, P., Rudnicky, A., Sanders, G., Seneff, S., Stallard, D., and Whittaker, S. 2001a. DARPA Communicator dialog travel planning systems: The June 2000 data collection. In *Proceedings of the 7th European Conference on Speech Communication and Technology (EUROSPEECH)*, Aalborg, Denmark, pp. 1371–4.

Walker, M. A., Fromer, J. C., and Narayanan, S. 1998. Learning optimal dialogue strategies: a case study of a spoken dialogue agent for email. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (ACL–COLING)*, Montreal, Quebec, Canada, pp. 1345–51.

Walker, M. A., Kamm, C. A., and Litman, D. J. 2000. Towards developing general models of usability with PARADISE. *Journal of Natural Language Engineering* (Special Issue on Best Practice in Spoken Dialogue Systems) **6**(3): 363–77.

Walker, M. A., Passonneau, R. J., and Boland, J. E. 2001b. Quantitative and qualitative evaluation of DARPA Communicator spoken dialogue systems. In *Proceedings of the 39th Meeting of the Association for Computational Linguistics (ACL)*, Toulouse, France, pp. 515–22.

Walker, M., and Passonneau, R. 2001. DATE: a dialogue act tagging scheme for evaluation of spoken dialogue systems. In *Proceedings of the Human Language Technologies Conference*, San Diego, CA, pp. 1–8.

Walker, M., Rudnicky, A., Aberdeen, J., Bratt, E., Garofolo, J., Hastie, H., Le, A., Pellom, B., Potamianos, A., Passonneau, R., Prasad, R., Roukos, S., Sanders, G., Seneff, S., Stallard, D., and Whittaker, S. 2002. DARPA communicator evaluation: progress from 2000 to 2001. In *Proceedings of the 7th International Conference on Spoken Language Processing (ICSLP)*, Denver, CO, pp. 273–6.

Webb, N., Hepple, M., and Wilks, Y. 2005. Dialogue act classification based on intra-utterance features. In *Proceedings of the AAAI Workshop on Spoken Language Understanding*, Pittsburgh, PA.

Williams, J. D., and Young, S. 2005. Scaling up POMDPs for dialog management: the "summary POMDP" method. In *Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop*, San Juan, Puerto Rico, pp. 177–82.

Young, S. 2000. Probabilistic methods in spoken dialogue systems. *Philosophical Transactions of the Royal Society (Series A)* **358**(1769): 1389–402.

Young, S., Evermann, G., Gales, M., Hain, T., Kershaw, T., Moore, G., Odell, J., Ollason, D., Povey, D., Valtchev, V., and Woodland, P. 2005. *The HTK Book* (for HTK version 3.3). Cambridge University Engineering Department.

Zinn, C., Moore, J. D., and Core, M. G. 2002. A 3-tier planning architecture for managing tutorial dialogue. In *Proceedings of the Intelligent Tutoring Systems, Sixth International Conference (ITS)*, Biarritz, France, pp. 574–84. Lecture Notes in Computer Science vol. 2363, Berlin: Springer.

Appendix: The algorithm of the automatic annotation system

```
function main( )
{

  indx_utt = 0;
  while not EOF {
    read utterance;
    utter[indx_utt] = utterance;

    if speaker[indx_utt]=system {
      save_information_state;
      go to beginning of loop;
    }


    /* in the following loop the annotation system looks backwards to all
       the system utterances between the last user utterance and the
       current one (not including the last user utterance) and parses them */
    k = indx_utt-1;
    indx_prev_utt = 0;
    while true {
      if speaker[k]=user
        exit loop;

      prev_utt[indx_prev_utt] = utter[k];
      extract prev_utt_speechact[indx_prev_utt];
      /* consider only request_info, explicit_confirm, etc. */
      res = member(prev_utt_speechact[indx_prev_utt], considered_speechacts);
      if res=false {
        k = k-1;
        go to beginning of loop;
      }
      extract prev_utt_task[indx_prev_utt];

      parse((i)prev_utt[indx_prev_utt], (o)prev_utt_keywords[indx_prev_utt],
                                        (o)prev_utt_values[indx_prev_utt]);

      correct_date_time_tags;

      indx_prev_utt = indx_prev_utt+1;
      k = k-1;
    }

    /* parse the current user utterance */
    parse((i)utt[indx_utt], (o)utt_keywords[indx_utt], (o)utt_values[indx_utt]);
    extract utt_speechact[indx_utt];
    extract utt_task[indx_utt];

    /* this is specific to the COMMUNICATOR corpus */
    decide_on_leg_of_trip;

    disambiguate_speechact_task((i)prev_utt_keywords, (i)prev_utt_values,
        (i)prev_utt_speechact, (i)prev_utt_task, (i)utt_keywords[indx_utt],
        (i)utt_values[indx_utt], (i)utt_speechact[indx_utt], (i)utt_task[indx_utt],
        (o)utt_speechact_new[indx_utt], (o)utt_task_new[indx_utt]);
```

```
    fill_slots_and_values((i)utt_keywords[indx_utt], (i)utt_values[indx_utt],
        (i)utt_speechact_new[indx_utt], (i)utt_task[indx_utt],
        (o)FilledSlots[indx_utt], (o)FilledSlotsValues[indx_utt]);

    detect_confirmedslots((i)prev_utt_keywords, (i)prev_utt_values,
        (i)prev_utt_speechact, (i)prev_utt_task,
        (i)utt_keywords[indx_utt], (i)utt_values[indx_utt],
        (i)utt_speechact_new[indx_utt], (i)utt_task_new[indx_utt],
        (o)confirmed_slots[indx_utt], (o)confirmed_slots_values[indx_utt],
        (o)unconfirmed_slots[indx_utt], (o)unconfirmed_slots_values[indx_utt],
        (o)utt_speechact_final[indx_utt], (o)utt_task_final[indx_utt]);

    remove unconfirmed_slots[indx_utt] from CurrentlyFilledSlots;
    remove unconfirmed_slots_values[indx_utt] from CurrentlyFilledSlotsValues;

    further update CurrentlyFilledSlots, CurrentlyFilledSlotsValues, CurrentlyConfirmedSlots;

    save_information_state;

    indx_utt = indx_utt+1;
  }

}


function disambiguate_speechact_task((i)prev_utt_keywords, (i)prev_utt_values,
         (i)prev_utt_speechact, (i)prev_utt_task,
         (i)utt_keywords, (i)utt_values,
         (i)utt_speechact, (i)utt_task,
         (o)utt_speechact_new, (o)utt_task_new)
{

  for i=0 to length(utt_speechact) {
    if utt_speechact[i]=provide_reprovide_info {
      res1 = member(explicit_confirm, prev_utt_speechact);
      res2 = member(implicit_confirm, prev_utt_speechact);

      if (res1=true or res2=true) {
        res = compare(utt_keywords, utt_values,
                      prev_utt_keywords, prev_utt_values);
        if res=true
          utt_speechact_new[i] = reprovide_info;
      }
      else if utt_speechact[i-1]=reject_info
        utt_speechact_new[i] = correct_info;
      else
        utt_speechact_new[i] = provide_info;
    }
    else
      utt_speechact_new[i] = utt_speechact[i];
  }

  for i=0 to length(utt_task)
    if utt_task[i] is marked as ambiguous
      compare(prev_utt_keywords, prev_utt_task,
              utt_keywords[i], utt_task[i], utt_task_new[i]);
    else
      utt_task_new[i] = utt_task[i];

}

function fill_slots_and_values((i)utt_keywords, (i)utt_values,
          (i)utt_speechact, (i)utt_task,
          (o)FilledSlots, (o)FilledSlotsValues)
```

```
{

  for i=1 to length(utt_speechact) {
    if utt_speechact[i]=provide_info or utt_speechact[i]=correct_info
          or utt\_speechact[i]=reprovide_info {
        associate_keywords_with_slots_and_values((i)utt_keywords,
            (i)utt_values, (i)utt_task, (o)Slots, (o)SlotsValues);
      add Slots to FilledSlots;
      add SlotsValues to FilledSlotsValues;
    }
  }

}

function detect_confirmedslots((i)prev_utt_keywords, (i)prev_utt_values,
          (i)prev_utt_speechact, (i)prev_utt_task,
          (i)utt_keywords, (i)utt_values, (i)utt_speechact, (i)utt_task,
          (o)confirmed_slots, (o)confirmed_slots_values,
          (o)unconfirmed_slots, (o)unconfirmed_slots_values,
          (o)utt_speechact_res, (o)utt_task_res)
{

  /* initialise the output speech act and task lists as empty lists */
  utt_speechact_res = [ ];
  utt_task_res = [ ];

  for i=1 to length(prev_utt_speechact) {
    res1 = member(explicit_confirm, prev_utt_speechact[i]);
    res2 = member(implicit_confirm, prev_utt_speechact[i]);

    if (res1=true or res2=true) {
      if length(utt_speechact)=1 {
        if utt_speechact[1]=yes_answer {
          associate_keywords_with_slots_and_values((i)prev_utt_keywords[i],
              (i)prev_utt_values[i], (i)prev_utt_task[i],
              (o)Slots, (o)SlotsValues);
          add Slots to confirmed_slots;
          add SlotsValues to confirmed_slots_values;

          add yes_answer to utt_speechact_res;
          add prev_utt_task[i] to utt_task_res;
        }
        else if utt_speechact[1]=no_answer {
          associate_keywords_with_slots_and_values((i)prev_utt_keywords[i],
              (i)prev_utt_values[i], (i)prev_utt_task[i],
              (o)Slots, (o)SlotsValues);
          add Slots to unconfirmed_slots;
          add SlotsValues to unconfirmed_slots_values;

          add no_answer to utt_speechact_res;
          add prev_utt_task[i] to utt_task_res;
        }
      }
      else {
        for k=1 to length(utt_speechact) {
          if utt_speechact[k]=provide_info or
              utt_speechact[k]=reject_info or
              utt_speechact[k]=correct_info {
            associate_keywords_with_slots_and_values((i)prev_utt_keywords[i],
                (i)prev_utt_values[i], (i)prev_utt_task[i],
                (o)Slots, (o)SlotsValues);
            add Slots to unconfirmed_slots;
            add SlotsValues to unconfirmed_slots_values;
          }
```

```
          else if utt_speechact[k]=reprovide_info {
            associate_keywords_with_slots_and_values((i)prev_utt_keywords[i],
                (i)prev_utt_values[i], (i)prev_utt_task[i],
                (o)Slots, (o)SlotsValues);
            add Slots to confirmed_slots;
            add SlotsValues to confirmed_slots_values;
          }

          add utt_speechact[k] to utt_speechact_res;
          add utt_task[k] to utt_task_res;
        }
      }
    }
  }

}
```