Discourse Motivated Constraint Prioritisation For Task-Oriented Multi-Party Dialogue Systems

Petra-Maria Strauß, Simon Friedmann, Tobias Heinroth
Institute of Information Technology
University of Ulm
Germany
{petra-maria.strauss;tobias.heinroth}@uni-ulm.de

Abstract

This paper presents a new algorithm to prioritise user constraints for problem solving in task-oriented multi-party dialogues. The situation of (at least) two users pursuing a common goal supersede the need for exhaustive semantic analysis which is commonly used in dialogue systems to prioritise user constraints. Instead, we suggest to use the ongoing discourse, especially the order of occurrence of the constraints for prioritisation. In this paper, we describe our algorithm and the scenario in which it was applied. We further present a first evaluation in which we compared our approach to semantic prioritisation which showed very promising results. It proofed that for our domain our simple algorithm outperformed its opponent by far.

1 Introduction

In this paper we present a new algorithm to prioritise user constraints for problem solving in task-oriented multi-party dialogue systems. Most prevailing spoken language dialogue systems (SLDS) are single-user systems. One user is interacting with the computer while the computer collects the information provided by the user. Multi-party systems pose new challenges, however, also new opportunities considering their characteristics. The multi-party SLDS interacts with more than one user, the users interact with the computer and additionally with each other. Thus, the system not only needs to understand the requests posed directly towards it but additionally has to follow the dialogue between the users in order to comprehend the entire conversation and grasp the context.

Naturally, the conversation partners often have different preferences which complicate the problem solving process immensely. The course of the dialogue also depends on the sort of dialogue and domain. In our example domain of restaurant selection two human dialogue partners and a computer interact with each other. The dialogue partners are generally not interested in a long discussion but rather in coming to a quick consensus. Besides uttering their own preferences and dislikes, the dialogue partners evaluate each other's preferences against their own and react accordingly.

The system therefore does not model the preferences of each user independently but collects all information relevant for the task to form a set of so-called constraints for the data base queries. If the query does not yield any results ('over-constraint situation'), an intelligent system is expected to provide an alternative solution. The common approach to that is to relax less important constraints. In single-user systems the prioritising of constraints is mainly performed by semantically analysing the constraint-bearing utterances in terms of keywords that denote the importance of the constraint to the user. However, the collection of valid constraints is also more straight-forward, depending on the preferences of the single user and on data-base constraints.

In the multi-party case, in the course of the dialogue each introduced constraint is discussed, rejected or accepted by the other dialogue partner. This makes automatic semantic analysis very complicated. We claim that in this case the prioritisation

process can be a lot simpler. We take the content of the discourse into account, i.e. the longer a constraint is valid in the dialogue, the more important it gets.

Various research groups have been working on the same domain of restaurant selection. For the system to cooperatively find a suitable restaurant, it has been found to be of utmost importance to consider the users' preferences, as well as also the strengths of these preferences (e.g. (Carberry et al., 1999)). Work on the closely related matter of presenting information and options in a SLDS can be found e.g. in (Demberg and Moore, 2006), (Walker et al., 2004), and (Carenini and Moore, 2001).

However, all of these surveyed dialogue systems are single-user systems. We state in the following sections why and how the multi-party situation differs from prevalent single-user systems and why it gives rise to new approaches. We briefly present the scenario and dialogue system in which we deploy the presented approach in Section 2. Section 3 focuses on different aspects of multi-party interaction. Section 4 introduces our approach to prioritisation exploiting multi-party characteristics to enhance the constraint based problem solving used in our system. The evaluation is described in Section 5 before Section 6 concludes the paper.

2 Multi-Party Dialogue System

Two human dialogue partners interact with a computer which acts as an independent dialogue partner in the scenario of restaurant selection (Strauss, 2006). In the beginning of the dialogue, the users talk about an optional topic while the system passively observes and captures the relevant conversational context. As soon as the users come to speak of the specified domain the system starts to "listen" attentively. When required by the conversational situation, it takes the initiative to get meaningfully involved in the communication and to help solve the task, i.e. help the users to find a suitable restaurant.

The analyses presented in this paper were performed on a set of dialogues from a corpus obtained through Wizard-of-Oz recordings (Strauss et al., 2008). The dialogues were transcribed and annotated with a simple tagset of 10 dialogue acts (refer to Section 3.2).

3 Multi-Party Interaction

Multi-party dialogue systems differ in many ways from single-user systems. The counterparts of the system are at least two people who interact with the system and additionally with each other. The face-to-face interaction with human dialogue partners is for humans still the most natural and comfortable way of communicating. Presumably, it is also faster and more efficient, e.g. due to the human ability of dissolving ambiguity by interpreting paralinguistic phenomena of communication such as emotions and facial expressions of the other dialogue partner. Thus, multi-party dialogue systems can be very advantageous in terms of that humans still communicate with each other and additionally turn towards the system only when necessary. The system is acting only as a side-participant of the conversation when the users don't need it.

Consequently, the design of our system is convenient as the users are able to first come to an initial agreement among themselves before the system gets involved in the conversation. This seems more efficient and faster than if they would have been interacting with the system during the entire process.

A further point crucial for the system's usability is the process of problem solving itself and how the results are presented to the users. Consideration and prioritisation of the users' preferences is an important issue in this respect. The research addressing this problem has so far been only considering single-user situation (e.g. (Carberry et al., 1999)). Before we present a new approach for the multi-party situation that utilises all the benefits that come with the additional dialogue partner, we discuss a few more points relevant for multi-party interaction.

3.1 Communication Roles

The situation in which the communication takes place has a substantial impact on the conversation. Next to the conversational roles such as speaker, addressee and overhearer (Clark, 1996), the roles the participants take on socially in the conversation play an important role in dialogues. (Bunt, 1994) introduced the social context as part of the dialogue context. (Traum, 2004) talks about the specific task roles which relate dialogue participants in certain ways.

Utterance	DA, Reference
A5: Let's go to an Italian restaurant.	(sugg,{})
B6: An Italian restaurant?	(check,A5)
A7: Yes.	(ack,B6)
B8: Ok.	(acc,A5)

Table 1: Dialogue snippet

During the Wizard-of-Oz recordings we randomly assigned different roles and scenarios to the dialogue partners. These included e.g. employer and employee, lovers, business colleagues, or friends. This way, we tried to obtain a wide range of different (such as superior / inferior) behaviour in our corpus which is important to be able to evaluate our approach on a broad variety of dialogues.

3.2 Interaction Phenomena

Task-oriented human-human dialogue shows a certain pattern which needs to be understood by the system to be able to model the conversation. The users mainly exchange proposals, introducing their preferences into the conversation. A proposal from one of the dialogue partners induces a reaction from the other dialogue participant. This response may consist of a simple acknowledgement, an accept or reject, a response with further content, or possibly a counter-proposal. Sometimes, the dialogue partner repeats the proposal which can have the function of acknowledgement, of checking if it was understood correctly or as a way of deferring the dialogue in order to win time to think. The response does not necessarily follow up a proposal but can also occur various turns later in the conversation with possibly even talking about a different topic in the meantime.

Table 1 shows a short example dialogue snippet labelled with the according dialogue act and the number of the utterance it refers to. We deploy a tagset of 10 basic dialogue acts which satisfies our domain and dialogue system requirements: *request, suggest, inform, acknowledge, check, accept, reject, stall, greet, other*.

User A proposes to go to an Italian restaurant. Instead of accepting right away, *User B* repeats *A*'s proposal whereupon *A* acknowledges *B*'s repetition. In this case, the repetition is to be interpreted as a request for clarification (check act).

4 Discourse Motivated Constraint Prioritisation

During the course of the conversation, the system collects all information relevant for the task which forms the basis for the database query and thus narrows down the result set in terms of positive or negative constraints.

If no results are obtained, i.e. an over-constraint situation occurred, the system should offer the users an alternative result. For this, we deploy constraint prioritisation in order to take user preferences into account. Different approaches to user preferences have been introduced (e.g. (Carberry et al., 1999)), however, only for single user dialogue systems. In a single-user system, finding out user preferences can be done using different methods. One is to analyse the semantic content of an utterance looking for specific words that show some kind of sign of importance, e.g. 'maybe', 'definitely', etc.[1] Another way is to simply ask the user about which constraint is more important in case that the system encounters an over-constraint situation.

In contrast, a multi-party system has one big advantage over all single-user systems: The additional - human - dialogue partner who already analyses the utterances from the other dialogue partner. When a suggestion is introduced with a 'maybe', this low priority is recognised by the dialogue partner, who can then accept the suggestion, or, being aware of the low priority of the proposal, make his or her own counter-suggestion which might be more precise.

A request to the computer is generally expressed only when the dialogue participants have found their highest common priorities. In the following, we describe the algorithm in detail.

4.1 Prioritisation Scheme

For prioritisation, information is extracted of each utterances according to three categories: Changing Categories, Current Preferences, and Prioritisation Values.

[1] Actually, it is not that simple as e.g. 'inferior' words are also more likely uttered by 'inferior' dialogue participants. However, a further elaboration on this aspect goes beyond the scope of this paper.

Changing Categories (CC) indicate the topic(s) of the current utterance. For instance, if one of the participants makes the statement of wanting to eat Italian food, the CC field is tagged with category (**F**) which stands for food or cuisine.

The other distinguishable categories are location (**L**), ambiance (**A**), category (**C**), price range (**P**), specials (**S**) and opening hours (**O**).

Current Preferences (CP) lists all currently valid constraints represented by individuals of the respective category and is thus used for a database query. In the example above, 'Italian' would be categorised as food (F) and individual **F1** (taken it is the first F-subject in this conversation). A second F-value later on in the dialogue, e.g. Mexican food, would then be tagged **F2**, etc. This is applied analogously to all other categories (L1, L2, P1 etc.).

Prioritisation Values (PV) assign a priority value to every individual. With every recalculation (induced by a change in the CP section) all currently valid values rise by '1 point'. A new individual is introduced with the value '1', i.e. it has risen '1 point' from the default value of '0'. Negative constraints or dislikes are represented with negative values accordingly (starting at '-1').

4.2 Executing the Prioritisation Scheme

In the following, the prioritisation algorithm is applied to a dialogue.

Introducing Preferences

At the beginning of a dialogue, the table contains no entries. As soon as a topic is raised, it is displayed in the CC section. The corresponding individual is inserted into the CP column of the table and the PV value is '1' (or '-1' in case of negation).

During the Dialogue

Every time the users modify their constraints, e.g. by proposing or dismissing one, a change in the CP section occurs and the PV are recalculated: The values of all individuals that are currently represented as valid preferences (in CP) are raised by '1' (or lowered by '1' for negative values).

Thus, the longer a subject stays valid, the higher its priority value becomes, which is obviously the desired effect. That means, as long as a subject is not explicitly abandoned or replaced by a different value due to incompatibility between constraints, it is considered valid and part of the current preferences. If a constraint is dismissed it is taken out of CP, its PV stays at the current value. Should it be re-introduced in the dialogue with the same polarity, it is reinserted into CP and the priority calculation starts at the former value. A change in the polarity of a valid constraint is performed by simply adding or removing the '-'.

System Involvement

All currently valid individuals are listed in the CP section which serves as the basis for the system's database queries. Every change in the constraint set induces a database query so the system is always up-to-date and ready to interact. Generally, the system interacts for the first time after the users have already come to an initial agreement. As also noted by (Carberry et al., 1999), this first request to the computer deserves special attention as it displays the users' original preference. Thus, all valid individuals at the time of the first computer request receive a *first request bonus* of '2'. This number provides an adequate trade-off between raising the priorities enough to stand out, but at the same time not too high so they can still be 'overruled', if necessary.

Table 2 shows a short part of a dialogue. At the beginning of this dialogue snippet, new individuals are introduced, namely 'Spanish' and 'Italian', 'F2' and 'F3'. It can be seen that the highest priority for the users at that point have the location 'L1' and the ambiance 'A1'. Both are at value '5', which means they assumably were the first constraints to be introduced in the dialogue and also received 2 points bonus. The second displayed utterance does not induce a recalculation of new preferences, due to the fact that nothing has changed. The request is simply repeated by the other dialogue participant. In the following utterance, the introduction of German cuisine as a negative value induces a change in CP and triggers a recalculation of the priorities.

4.3 Applying Prioritisation to Constraint Based Problem Solving

At present, the prioritisation only comes into play in the case of an over-constraint situation, i.e. if the database query does not yield any results. In order to offer the users a best possible alternative result

Utterance	CC	CP	PV	
...				
B: I would like something Italian or Spanish.	F	F2 F3 C1 L1 A1	F2 = 1 F3 = 1 C1 = 4 L1 = 5 A1 = 5	(italian) (spanish) (restaurant) (river) (beergarden)
A: Italian or Spanish is fine with me.				
A: I just don't want German food.	F	F4 F2 F3 C1 L1 A1	F4 =-1 F2 = 2 F3 = 2 C1 = 5 L1 = 6 A1 = 6	(german) (italian) (spanish) (restaurant) (river) (beergarden)
...				

Table 2: Prioritisation scheme applied to an extract of a dialogue.

the system has to decide which constraint(s) to relax. We deploy the following (simplified) algorithm:

```
while overconstraint OR resultset ==
previous_resultset do
    if onto_check(relax_candidate).succeed then
        present_results();
        break;
    else
        if relax(relax_candidate).succeed then
            present_results();
            break;
        end
    end
    relax_candidate++;
end
```

Algorithm 1: Simplified relaxation algorithm

The constraint with the lowest priority value is chosen as the first relaxation candidate. The result of the following query is analysed in terms of another over-constraint situation. The result set is further compared to the result set that was presented to the users in the system's last turn before the initial over-constraint situation occurred. If the result sets are the same, i.e. the same result set that obviously had just been rejected or further constrained by the users would be presented again. Thus, the relaxation algorithm proceeds at this point. If again no result was obtained, the relaxed value is reinserted before the next relaxing candidate is considered for relaxation. After another unsatisfying result, both values are relaxed etc. The presented algorithm is simplified in this matter and also in the way that it assumes that each time there is exactly one constraint with minimal priority value which, however, is not always the case. The implemented algorithm handles this by trying out each of the potential relaxation candidates and taking the one with the best results.

Before relaxing, the relaxation candidate is inspected in the context of the ontology to take related values into account. If e.g. no restaurant can be found near the town-hall before relaxing this constraint, it will be checked if there would possibly be one around the cathedral which is the adjacent area. This kind of ontology check can be performed for all exclusive categories (L, F, P, C, and A). However, the observation of the recorded dialogues showed that some values should not be relaxed if possible. They include e.g. the values of category *S* (i.e. 'specials', such as cocktails), or 'expensive' of category *P*, as well as negative constraints. No matter at what point these values were introduced in the dialogue, they were very important to the users and therefore not relaxed.

5 Evaluation

We performed evaluation on a set of dialogues from our corpus (Strauss et al., 2008). In the normal course of a dialogue all constraints are considered in each database query regardless of their priority. Therefore, evaluation can only be performed on di-

alogues where over-constraint situations occurred. This resulted in a set of 14 dialogues.

At recording time, the system simply told the users that there were no results found. The users then modified their query according to their preferences. For evaluation we compared the outcome of our algorithm to the users' reaction to the over-constraint situation and how they proceeded in the dialogue, i.e. which constraints they relaxed or modified. The relaxation algorithm performed equally well or better in 13 out of 14 dialogues, i.e. the algorithm leaded to relaxing the same constraints as the users did. By conducting the ontology check, in 5 of the 13 cases the outcome would have even been better as the system would have suggested a result closer to the original preferences than what was obtained in the dialogue.

We further compared the performance of our algorithm to semantic prioritisation. For this, we hand-annotated the constraints (mainly by considering keywords that denote importance) using a weighting scheme from '1' (little interest) to '5' (strongest interest). The same range is applied to dislikes ('-5' to '-1', with '-5' meaning strongest dislike). Weights were dynamically adapted during the course of the dialogue, if necessary. The semantic algorithm performed as well as ours in 6 cases. In most cases it relaxed the constraints in a different order which mostly also lead to a different result set. The semantic algorithm repeatedly tried to relax one or more of the users' main preferences which e.g. becomes apparent in one of the dialogues just after the over-constraint situation when the users tried to rephrase their main preferences which at this point the system already would have had relaxed. In 1 case, the semantic algorithm performed better than ours in the way that it relaxed the same constraint as the users when ours did not.

The overall result is therefore very affirmative: Our algorithm represents user preferences equally well or better than a similar method using semantic analysis for prioritising user constraints in all but one evaluated cases.

6 Conclusion

In this paper we presented a new algorithm to prioritise user preferences in a task-oriented multi-party dialogue system. We use the ongoing dialogue to assign priority values to the constraints, i.e. the longer a constraint is valid in the dialogue the more important it gets. The evaluation of our simple approach showed auspicious performance. We compared it to a semantic prioritisation approach as well as to how the users actually proceeded after an over-constraint situation had occurred in the analysed dialogues. In all but one case, our algorithm performed equally well or better.

Future work includes further evaluation, also using different domains. Additionally, we are planning to take the frequency of changes in a certain category into account. For instance, if the users switch many times between different kinds of cuisine, the value for this category would be rather high and imply a sort of uncertainty and flexibility in this aspect.

References

Harry C. Bunt. 1994. *Context and Dialogue Control*. THINK Quarterly, vol.3, pp.19-31.

S. Carberry and J. Chu-Carroll and S. Elzer. 1999. *Constructing and Utilizing a Model of User Preferences in Collaborative Consultation Dialogues*. Journal of Computational Intelligence, vol.15, no.3, pp.185-217.

G. Carenini and J.D. Moore. 2001. *An Empirical Study of the Influence of User Tailoring on Evaluative Argument Effectiveness*. IJCAI, pp.1307-1314. Seattle, Washington, USA.

H.H. Clark. 1996. *Using Language*. Cambridge University Press. Cambridge, England.

V. Demberg and J.D. Moore. 2006. *Information Presentation in Spoken Dialogue Systems*. EACL, pp.65-72. Trento, Italy.

P.-M. Strauss. 2006. *A SLDS for Perception and Interaction in Multi-User Environments*. 2nd International Conference on Intelligent Environments (IE06). Athens, Greece.

P.-M. Strauss and H. Hoffmann and W. Minker and H. Neumann and G. Palm and S. Scherer and H. C. Traue and U. Weidenbacher. 2008. *The PIT Corpus Of German Multi-Party Dialogues*. International Conference on Language Resources and Evaluation (LREC). Marrakech, Morocco.

D. Traum. 2004. *Issues in Multiparty Dialogues*. Advances in Agent Communication Ed. F. Dignum. Springer-Verlag LNAI 2922, pp.201-211.

M.A. Walker and S.J. Whittaker and A. Stent and P. Maloor and J.D. Moore and M. Johnston and G. Vasireddy. 2004. *Generation and evaluation of user-tailored responses in multimodal dialogue*. Journal of Cognitive Science, vol.28, pp.811-840.