Proactive Systems and Influenceable Users: Simulating Proactivity in Task-oriented Dialogues

Vevake Balaraman[1,2], **Bernardo Magnini**[1]
[1]Fondazione Bruno Kessler, Trento, Italy
[2]ICT Doctoral School, University of Trento, Italy
{balaraman, magnini}@fbk.eu

Abstract

We investigate proactivity, the capacity of a dialogue system to provide relevant information even when not explicitly requested, in the context of task-oriented dialogues. We propose to extend the current task-oriented framework, and have investigated four aspects of proactivity: (i) the degree of proactivity provided by the system during a dialogue; (ii) the propensity of the user to be influenced by the system proactivity; (iii) the complexity of the domain ontology; (iv) the relation between user needs and application domain, in terms of expected failure situations. Under the hypothesis that proactivity helps to increase effectiveness and efficiency of dialogues, we set up a framework based on dialogue simulations, and experimented the four aspects mentioned above. Although the current implementation allows to simulate a limited amount of dialogue phenomena (e.g., system initiative only), we are able to show that proactivity might have strong effects on dialogues, reducing up to 60% of dialogue turns in an application domain of medium complexity.

1 Introduction

Proactivity is a fundamental characteristic of human-human collaborative dialogues, consisting in the attitude of the speakers to provide information that can be used to achieve the goal of the dialogue, even when such information is not explicitly requested. This attitude obeys to the so called *principles of cooperative dialogue*, which have been summarized in the popular Grices's maxims (Grice, 1975). Among the others, proactivity is very common in instruction-giving dialogues. For instance, assuming the following dialogue between speakers A and B,

(A) *What time is the next train to London?*
(B) *The next train is at 1015. It arrives at 1245.*

we say that *It arrives at 1245* is a proactive response, because information given by B is more than was asked for by A, and because B guesses that this is the sort of information A might also need, and so offers it unsolicited. The major effect of proactivity is that, by anticipating user needs (e.g., a question like *What time this train will arrive in London?* in our example), it avoids long dialogues, making communication more efficient.

Proactive behaviours share some aspects with other communicative devices. Persuasion (Fogg, 2002; Wang et al., 2019) provides not requested information too, although not necessarily in response to a user request (i.e., we can try to persuade even without a dialogue). Recommendation (Chen et al., 2019), has assumed an application connotation, i.e., suggesting items that match the user profile during a search process. We prefer to use *proactivity* as a general term to indicate any situation in a task-oriented dialogue where the system takes the initiative to provide a piece of non requested information with the goal of better achieving the user needs.

Finally, proactivity goes beyond verbal communication, and may involve multimodal modalities. A well known experience has been Microsoft Clippy, launched as an helper for the Microsoft operative system. The substantial difficulties encountered by Clippy to be well accepted by users, mostly because of its perceived invasiveness, should be considered as a signal that effective proactivity is actually a very complex phenomenon.

Although very common in human-human dialogues, proactivity is almost absent from the current research in task-oriented dialogue systems. To the best of our knowledge, proactivity is largely under represented in most of the datasets (e.g., ATIS (Price, 1990), WoZ (Mrkšić et al., 2017), MultiWoz (Budzianowski et al., 2018), Snips (Coucke et al., 2018)) that are used to train either natural language understanding or dialogue state tracker components

Non-proactive Dialog	Proactive Dialog
User: Are there any Eritrean restaurants in town? **Sys:** No, there are no Eritrean restaurants in town. Would you like a different restaurant?	**User:** Are there any eritrean restaurants in town? **Sys:** No, there are no eritrean restaurants in town. *Most of the restaurants are Chinese, particularly in the East area of the town*
User: How about Chinese food? **Sys:** There is a wide variety of Chinese restaurants. Do you have an area preference or a price preference to narrow it down?	**User:** Chinese food in the East area is fine. **Sys:** Yu Garden is a Chinese restaurant in the east area, *with mid level price*.
User: I would like the East part of town. **Sys:** Yu Garden is a Chinese restaurant in the east area.	
User: Could I get the address, phone number, and postcode of Yu Garden? **Sys:** Phone is 01223 248882, address and postcode are 529 Newmarket Road Fen Ditton C.B 5, 8 P.A	**User:** Could I get the address, phone number, and postcode of Yu Garden? **Sys:** Phone is 01223 248882, address and postcode are 529 Newmarket Road Fen Ditton C.B 5, 8 P.A
User: Thank you. That is all the information I needed. Bye bye!	**User:** Thank you. That is all the information I needed. Bye bye!

Figure 1: On the left side: a non proactive sample dialog from the WoZ dataset. On the right side: a proactive version of the same dialogue (arranged by us). Turns are separated by dotted-lines; proactive contributions of the system are in *italic*.

of conversational agents. A more detailed analysis on proactivity in existing task-oriented datasets is presented in Section 3).

Figure 1, left side, shows a typical task-oriented dialogue from the WoZ dataset (Wen et al., 2017), collected through Wizard of Oz and focused on booking restaurants. Here we notice the behaviour of the system in case of a failure situation (i.e., no match with the user needs), at turn 1: *No, there are no Eritrean restaurants in town. Would you like a different restaurant?*, and compare it with a proactive version of the same dialogue, arranged by us, showed in Figure 1, right side. In this proactive version the systems responds: *No, there are no Eritrean restaurants in town. Most of the restaurants are Chinese, particularly in the East area of the town.*, offering to the user potentially useful domain information to reorient the search. At the next turn the user may accept or not the proactive contribution of the system. In case the user accepts it (as in Figure 1, right side), there is no need of a further question about food type and the area of the restaurant, and the system can directly move to propose a specific restaurant, saving one turn in the dialogue. In case the user does not accept the proactive contribution, the dialogue will follow the non-proactive path, and the system can implicitly assume a low match with the user needs for the facts involved in the proactive contributions, which will not be proposed to the user anymore. In both of the cases, we see that a proactive behaviour has brought the system to a more precise definition of the user needs (i.e., the belief state of the system).

In the paper we aim at a first step toward extending the current task-oriented framework to accommodate proactive behaviours. We investigate the following research questions: (i) which are the main features that affect proactivity in task-oriented dialogues? and (ii) can we model such features within the architecture of current dialogues systems? and (iii) how can we show the impact of proactivity on task-oriented dialogues, e.g., in terms of effectiveness and efficiency of the dialogues?. The code used for the simulation is made publicly available.[1]

2 Related Work

In this section we report works that are related to proactive dialogues in both task-oriented and open domain (i.e., chit-chat) dialogue systems.

2.1 Proactivity in Task-oriented Dialogues

Task-oriented dialogues, usually distinguished from so called chit-chat dialogues, cover a broad range of applications (Allen et al., 2001; Bohus and Rudnicky, 2009), including giving instructions,

[1] https://github.com/vevake/Proactive_dialog_simulation

question answering, executing commands, conversational search, recommendation dialogues, and also explanatory dialogues. In this paper we focus on task-oriented dialog systems whose aim is to help the uses to retrieve items (i.e., entities) in a certain domain of interest, through *conversational search* (Radlinski and Craswell, 2017). Such systems (Young et al., 2013; Bordes et al., 2017; Lei et al., 2018; Wen et al., 2017) are modelled to achieve the task by actively questioning the user for slot values in a knowledge base, and then to retrieve entity instances satisfying the user needs. We report some work on recommendation, dialogue initiative, and persuasion, which we believe is related to proactivity.

Recommendation dialogues. (Thompson et al., 2004) propose ADAPTIVE PLACE ADVISOR, an adaptive conversational recommendation system designed to help people to select an item from a Knowledge Base. (Sun and Zhang, 2018) propose a conversational recommender system based on user's past ratings and the current interactions to make personalized recommendations to the user. The cold-start problem for recommendations in task-oriented dialogue systems was investigated by (Christakopoulou et al., 2016), proposing a preference elicitation framework to learn the preferences of a new user in fewer interactions and then use preferences to make recommendations. (Yoshino and Kawahara, 2015) propose a news navigation system that proactively presents users with information by tracking on the user focus.

Mixed initiative dialogues. Collaboration is particularly investigated in the context of mixed initiative dialogues, with the general goal of modelling dialogue policies for several situations (e.g., negotiation and argumentation dialogues). (Guinn, 1996) examined how to incorporate the initiative in a task-oriented dialogue system and proposed attaching a initiative level to each goal and as the goal of the dialogue changes, so does the dialogue initiative. (Yang and Heeman, 2007) investigated human-human conversations, proposing that initiative normally belongs to the speaker who initiates the task. They support the claim though simulations with two variations of mixed-initiative strategy. (Nouri and Traum, 2014) addressed mixed-initiative dialogue patterns in a negotiation dataset and the relationship between such patterns and the goal/outcome of the dialogue. They propose an annotation schema for dialogue initiative that allow to identify key features in a negotiation dialogues.

Persuasive dialogues. Unlike recommendation and dialogue initiative, where system and user cooperate to achieve a common goal, persuasive dialogues aims at convincing the user to achieve the systems goal (e.g., buying a target item or donating for charity). (Hiraoka et al., 2014) analyse human persuasion and the relation with user satisfaction. They propose an annotation scheme for persuasive dialogues and analyze a set of dialog acts by collecting a human-human dialog dataset for camera sales, showing that the main dialogue acts are information exchange and argumentation. (Hiraoka et al., 2013) propose a dialogue manager based on Bayesian network framework. They construct a knowledge-base that captures the relationship between various topics, which is then used to guide the user from topic to topic. (Wang et al., 2019) work on identifying correct persuasion strategies and analyse which strategies were effective based on user backgrounds.

While recommendation, user initiative and persuasion address collaborative phenomena in dialogue, they also tend to influence the user toward a specific target goal/item. Although the border between recommendation and proactivity is fuzzy, we intend proactivity as a general collaborative strategy aiming at improving the conversation quality and effectiveness, particularly in case of failure situations.

2.2 Proactivity in Open-Domain Dialogue Systems

Proactivity is also addressed for open domain dialogue systems due to high user expectations in open domain conversations. (Li et al., 2016) propose STALEMATEBREAKER, a system that can proactively introduce new content during a conversation, when a stalemate situation (e.g., "..." or "Errr", etc.) in the conversation is detected. The new content is dependent on the context of the conversation and is retrieved from a knowledge graph. (Yan and Zhao, 2018) address proactivity combining the response generation and a suggestion generation, and proposing a hybrid approach that provides a smart response (in addition to the system response), which the user can use as the input for the next turn. While this approach is feasible for open domain systems, for task-oriented dialogue systems such as smart responses, this is inefficient because of the nature of the task. (Wu et al., 2019) propose a view on proactivity where the system is actively leading the conversation to achieve a specific goal. This requires the system to generate plans over a knowledge graph.

3 Proactivity in Existing Task-oriented Datasets

Modeling proactivity for task-oriented dialogues requires the system be aware of the content of domain and be able to provide responses that help the user to better achieve his/her goals. In general, a proactive system should be guided by two strategies: (i) try to avoid failure situations, providing the user with domain information before the point of failure; (ii) in case a failure is reached, use proactivity to recover the dialogue. In this Section we are interested to analyse whether the two strategies mentioned above are represented in current available datasets.

Existing datasets for task-oriented dialogue systems are widely collected using a Wizard-of-Oz (WoZ) approach. According to the WoZ methodology dialogues are collected following a script that is provided both to the user, who believes to interact with a system, and to the wizard, a person who simulates the behavior of a system. Typically, the wizard extracts information to be provided to the user searching in a domain Knowledge Base (e.g., restaurants in a town). However, since the dialogue follows a pre-defined script with a description of the actions that the wizard can take, if proactive behaviours are not included in the script, the responses of the Wizard will be quite standard, resulting in a general lack of proactivity. To support this intuition, we analyse two of the most widely used multiturn datasets for task-oriented dialogue systems, namely WoZ (Wen et al., 2017; Mrkšić et al., 2017) and MultiWoZ (Budzianowski et al., 2018), searching for system proactivity. In both the datasets proactivity is not marked in any way (i.e., there is no a specific dialogue act for it). In order to analyse proactivity we focused on failure situations, where there is no match between the constraints posed by the user and the content of the system domain (see example reported in Figure 1). These situations are both easy to be individuated in the dialogue corpus through simple patterns (e.g., *There are no...*), and, according to our expectations, are those where the Wizard should be mostly proactive to help the user. As a consequence, we assume that restricting our search to those situations can give a good approximation of proactivity in the corpus.

In the WoZ dataset we found over 200 failure situations, i.e., turns where the system was not able to find any matching with respect to the user constraints. In all such turns, except four of them, the response of the Wizard was a non proactive behaviour of the type "*There are no <SLOT-VALUE> restaurants in town. Would you like a different restaurant?*" where <SLOT-VALUE> corresponds to a combination of slot values requested by the user (e.g., Eritrean, as in our example in Figure 1). We can conclude that proactivity was not a relevant design criteria for the WoZ dataset.

As for the MultiWoZ dataset, we have individuated nearly 3000 turns where the user constraints do not yield any matching result. Out of them, we noticed around 10% of turns providing a proactive response to the user, such as "*There are no hotels that fit your criteria in the <AREA>, but there are two Guesthouses. Would you like to book one of those?*". Such responses in MultiWoZ were explicitly mentioned in the dialog task description of the Wizard, with the goal of modeling more realistic conversations (Budzianowski et al., 2018). We conclude that in MultiWoZ proactivity, at least in case of search failure, has been considered by design. However, we think that: (i) the amount of proactivity is still under represented, and (ii) maybe more important, being not annotated in any way, there is virtually no difference between proactive and non proactive behaviours.

4 Modeling Proactivity

In this section first we introduce the relevant background about modeling task-oriented dialogues, and then we attempt to extend such framework to integrate proactivity.

4.1 Task-oriented Dialogue Framework

According to most of the recent literature (Budzianowski et al., 2018; Bordes et al., 2017; Mrkšić et al., 2017), we assume a task-oriented dialogue between a system and an user, composed of a sequence of turns $\{t_1, t_2, ...t_n\}$. The goal of the dialogue is to retrieve a set of entities (possible empty) in a domain knowledge base (KB), which satisfy the user needs. A domain ontology O provides a schema for the KB, and typically represents entities (e.g., RESTAURANT, HOTEL, MOVIE) according to a pre-defined set of slots S

(e.g., FOOD, AREA, PRICE, for the RESTAURANT domain), and values that a certain slot can assume (e.g., EXPENSIVE, MODERATE and CHEAP, for the slot PRICE). On the base of the entities defined in the domain ontology, the application knowledge base, KB, is then populated with instances of such entities. As in most of the literature, we distinguish *informable slots*, which the user can use to constraint the search (e.g., AREA), and *requestable slots* (e.g., PHONENUMBER), whose values can be asked only when a certain entity has been retrieved. We say that the *entity space* of a domain ontology O is the combinatorial product of the slot values defined for each informable slot in O.

$$Entity\ Space_{O_d}(E) = \prod_{i=1}^{|S|} |V_i|$$

where $|V_i|$ is the number of possible values for an informable slot S_i in a given domain ontology O_d. At each turn in the dialogue, both the user and the system may refer to facts in the KB, the user with the goal of retrieving entities matching his/her needs, and the system to pro-actively propose entities that can help the user to achieve the dialogue goals.

Finally, in such task-oriented systems a dialogue state tracker (DST) maintains a distribution over the dialogue states based on the dialogue history. A dialogue state d_i for a turn t_i is typically represented as a set of slot-value pairs, such as {PRICE=MODERATE, FOOD=ITALIAN}, meaning that at t_i the system assumes that the user is looking for an Italian restaurant with a moderate price.

4.2 Proactive Units

We define a proactive behaviour as any information that: (i) is introduced by the system; (ii) was not previously introduced in the dialogue by the user; and (iii) it is assumed to be relevant to achieve the user needs. We call a piece of proactive content *proactive unit*. Intuitively, a proactive unit pu_i relative to a certain dialogue state d_i is a sort of guess of the user need given the information available at turn t_i. For instance, assuming
d_i= {AREA=CENTER}, then both:
{AREA=CENTER, FOOD=ITALIAN} and
{AREA=CENTER, FOOD=ITALIAN,
PRICE=MODERATE}
are possible proactive units for d_i.

Although in principle there might be different ways to identify a proactive unit during a dialogue, in this paper we focus on proactive units generated through *completion* of the dialogue state. Given a dialogue state $d_i = \{s_1 = v_1, ...s_n = v_n\}$, a completion for d_i is defined as an extension of d_i with slot-value pairs defined in the Ontology O, with the constraint that only one slot-value pair for each slot is admitted. Notice that, according to the task-oriented framework we adopted (Budzianowski et al., 2018), a dialogue state can only contain informable slots (see Section 4.1) and we apply the same constraint to its completions. This means that proactive units only contain informable slots, and that suggestions containing requestable slots (e.g., "Bella Napoli offers a large variety of pizzas") are not allowed. This is the way we capture the distinction between proactivity and recommendation mentioned in the Section 1.

Depending on the complexity of the domain ontology (i.e., number of slots and slot values), the number of proactive-units generated through completion for a dialogue state may be high, and it becomes useful the capacity to rank the units according to their relevancy, so that the more relevant are selected as proactive suggestions to the user. As a first approximation to a relevance function, we consider the probability that the entities described in the proactive units are actually present in the domain KB. Intuitively, the more the chances that a proactive unit have occurrences in the KB, the easier for the dialogue system will be to accomplish the user goals. To compute the probability of a proactive unit we consider the fraction of KB entities that matches the slot-value pairs out of the KB entities that match the dialogue state. More precisely:

$$P(pu_i|d_i) = \frac{\#matches(pu_i)}{\#matches(d_i)}$$

where the number of matches corresponds to the number of entities (i.e., instances) in the KB that satisfy all the slot-value pairs in pu_i or d_i.

For instance, if the user asks for Italian restaurants (d_i={FOOD=ITALIAN}), the following could be a partial rank of the proactive units that complete the information in the dialogue state:

([FOOD=ITALIAN], [AREA=CENTRE], $p = 0.8$)
([FOOD=ITALIAN], [AREA=SOUTH], $p = 0.2$)

assumed that 80% of the ITALIAN restaurants in the KB are in CENTRE, while 20% are in the SOUTH). According to this ranking, the first proactive unit will have more chances to be selected in that dialogue turn.

Notice that the definition of completion provided above is constrained by the slots defined in the domain ontology, and does not consider other possible features, such as those related to the user identity (e.g., gender, age), nor situational features (e.g., time of the day).

4.3 System Proactivity

Having now clear our definition of proactive units, a first aspect that needs to be modelled is the attitude of the system to inject such proactive units during the dialogue. Based on the fact that proactive units are "completions" of a dialogue state, we consider those turns in the dialogue involved in the search through informable slots, and exclude turns where the choice is already made and the user asks additional information on requestable slots (e.g., telephone number of the selected restaurant).

Intuitively, at each (search) turn the system has to take two decisions: first, whether to be proactive or not, and, second, in case the system decides to be proactive, the amount of proactive units to be displayed to the user. We regulate the two decisions using two parameters. a is a value in the range [0,1] that defines the apriori attitude of the system to be proactive. It is interpreted as the proportion of turns in the dialogue that show proactivity. For instance, $a = 0.5$ means that 50% of the turns in the dialogue are expected to be proactive. A second parameter is k, indicating the average number of proactive units to be used by the system at each proactive turn. k ranges from 1 (i.e., an average of one proactive unit) to $|E|$, the cardinality of the entity space (see Section 4.1) defined over the domain ontology O. In practice, k is typically very small, as high numbers of proactive units will not be cognitively accepted by users.

The model we propose defines a and k as average behaviours of the system, and it is agnostic about how the average is actually achieved. However, in most situations it seem reasonable that both a and k distribute, respectively their proactive turns and proactive units, quite uniformly through the dialogue, avoiding dialogues where, e.g., all proactive turns are at the end, or all the proactive units are used in a single turn. In our experiments (Section 6) we adopt a similar strategy.

4.4 User Influenceability

A second aspect that needs to be considered while modeling proactive dialogues is the propensity of the user to accept the system suggestions. Intuitively, in order to produce the expected benefit, the user has to somehow "accept" the proactive content provided by the system. Remind that we defined proactivity as a neutral behaviour, i.e., general information about the domain of the task (e.g., "Most Italian restaurants are in the city center"), rather than specific recommendations (e.g., "Bella Napoli offers a large variety of pizzas"). While acceptance may depend on several reasons (e.g., quality of the proactive content), we approximate the propensity of the user to accept a proactive information, i.e., *user influenceability*, with a probability value $[0, 1]$, where 0 corresponds to a user that never accepts the system information (i.e., a non influenceable user), and 1 to a user that always accepts proactive content (i.e., a very influenceable user). For instance, a user influenceability set to 0.5 means that 50% of the proactive content provided by the system will be "accepted" by the user, while the remaining 50% will be ignored.

4.5 Complexity of the Application Domain

A third aspect that may affect proactivity is the complexity of the application domain. The intuition is that solving a conversational search task (e.g., retrieving a restaurant) in a complex domain might require more interactions than solving the same task in a less complex domain. According to this intuition, the more the domain complexity, the more proactivity is necessary to help the user to make efficient dialogues. We define domain complexity considering the number of entities defined in the domain ontology, their slots and their slot values. Practically, domain complexity is approximated with the domain entity space, defined in Section 4.1, and calculated as the combinatorial product of the slot value sets, for each slot of the domain.

4.6 Failure Situations

The last aspect that we aim to capture is that proactivity is particularly useful to recover failure situations, when the user needs do not match with any actual content of the domain KB, and some form of "adjustment" of the initial needs is typically attempted in order to successfully conclude the dialogue. This is the situation shown in Figure

1, where initially the user is looking for an Eritrean restaurant, and, after being informed that there are no such restaurants (i.e., a failure), he/she makes use of the system information and modifies the initial goals. Intuitively, the more failure situations happen in a dialogue, the more the role of system proactivity, and the more the expected benefits in term of task accomplishment.

We denote the probability of a failure situation as *failure expectation* and estimate it, given a certain application KB, considering the prior probability that a certain user need (e.g., a restaurant with certain characteristics) actually occurs in the domain KB, and average such probability over all the potential user needs (i.e., the entity space defined over the domain ontology O).

$$\begin{matrix} Failure \\ expectation \end{matrix} = 1 - \frac{1}{|E|} \sum_{i=1}^{|E|} \begin{cases} 1 \text{ if } E_i \text{ in } KB \\ 0 \text{ else} \end{cases}$$

where E is the entity space. The more the failure expectation (e.g., "we do not have such restaurant"), the higher the need to help the user with proactivity. We practically estimate the probability of a user need to occur in the KB with the number of entity instances in the KB: the more the instances, the more the probability that the user need can be found in the KB.

Note that the size of the KB and the domain complexity described in section 4.5 are independent: a domain can be very complex (i.e., many slots and many slot values) although very poorly populated (i.e., few instances), and vice versa.

5 Simulating Proactivity

As there are no available datasets that can be used to test our model of proactive dialogues, we base our investigation on dialogues generated through simulation. We used an existing multi-domain dialog generator, *SimDial*[2] (Zhao and Eskenazi, 2018), and extended it by incorporating both the system proactive behaviour and user influenceability, as defined in Section 4. SimDial is based on a set of *domain specifications* describing the ontology of the dialogue task and a set of *complexity specification* that describe dialogue policies. As for domain we focus on restaurant booking defining its possible slots and values, while for dialogue policies we used the default specifications provided by SimDial.

[2]https://github.com/snakeztc/SimDial

5.1 Domain KB and Proactive Units

Our dialogue simulator is based on a default domain ontology for restaurants, consisting of three informable slots (i.e., AREA, PRICE, FOOD) with, respectively, 12, 10, and 3 pre-defined values, and two requestable slots (i.e., PARKING, OPEN). We also populated a default KB with 200 instances, randomly using the set of pre-defined slots.

Then, we extracted all the proactive units (see Section 4.2) from the default Ontology, considering both single and combinations of multiple slot-value and pairs, and then we computed their probability distribution in the KB. Such proactive units are stored in a repository called *KB-stats*, and made available as possible proactive information that the system can choose to convey to the user. The *KB-stats* is re-computed for each different configuration of the KB of our experiments.

5.2 Proactivity Module

The proactivity module that we incorporate to SimDial has access to *KB-stats*, which contains information about proactive units for the domain, as defined in Section 5.1. In order to keep the efficiency of the simulator under control, the content of the *KB-stats* is pre-computed. Specifically, we pre-compute the complete set of proactive units (see Section 4.2) considering the slots and the slot values in the ontology, as well as the probability of each proactive units on the base of the entities of the populated KB. We use the *KB-stats* to efficiently generate the pro-active-units for a certain dialogue state. The query to the *KB-stats* is the current dialog-state and the result is the list of proactive units that "complete" the dialogue state with their probability, according to relevancy definition provided in Section 4.2. For example, if the current dialog state is FOOD=ITALIAN, the proactive information could be (FOOD=ITALIAN, AREA=WEST, 0.8), which informs the system that 80% of the *Italian* restaurants are in the *west*.

5.3 Dialog Agent

The SimDial dialog-agent has a default policy which consists to request the user-agent for all the informable slots and, after that, to perform a domain query to retrieve entities from the KB. The order in which the informable slots are requested to the user-agent is randomized. This is the non proactive policy of SimDial.

The dialog-agent with the proactive policy requests the informable slot to the user-agent, as in the default policy, and, in addition, provides the proactive information (i.e., the proactive units) that may help the user to achieve his goal. To convey the proactive information, we have defined proactive templates, which are used to transform the abstract information of the proactive units into a natural language utterance.

In particular, we define two kinds of templates that show proactive information: i) the first template indicates that all the proactive units for the current dialogue state lie in a single search space (e.g., All *Italian* restaurants are in *Palo Alto* and are *expensive*); ii) the second template indicates that most of the proactive units for the current dialogue state lie in a single search space (e.g., Most *Italian* restaurants are in *Palo Alto*). For example as seen in Figure 2, the system at turn 7 requests the user-agent for the PRICE slot and then provides proactive information that all restaurants are cheap. The selection of the template is carried on based on the proactive information retrieved from the *KB-stats*. If the selected proactive unit for the current dialog state has probability 1.0, then the first template is chosen otherwise second template is preferred.

We also added an additional policy to the dialog agent: if, for a particular dialog state, there is no proactive information retrieved, then it implies that there are no entities in the KB matching the dialogue state. In this case, instead of the dialog-agent continuing to query the user-agent for additional slots, we directly inform the user-agent to change his goal, as there exists no result for the current goal.

5.4 User agent

The default SimDial user-agent responds to the slot requested by the dialog-agent with a slot value based on his current goal. When the dialog-agent returns a restaurant match for the user-agent's goal, the user-agent may ask for any additional information on the available requestable slots (i.e., PARKING and OPEN).

In case of a proactive dialogue agent, the user-agent receives an additional proactive information at each turn, and has the option to either accept or ignore this suggestion. We simulate this user behaviour through the user influenceability probability value, as defined in Section 4.4.

An interesting situation happens when the selected proactive unit contains multiple slots (e.g., Most *Italian* restaurants are in *Palo Alto* and are *expensive*). In this case we designed a policy that allows the user-agent both to accept the entire proactive unit, or just a part of it. This policy, partial acceptance, in the current version is not explicit as a dialogue act, rather it is implemented by the user-agent repeating the slot value suggested by the dialog-agent. The reason for that is due to the limitations of the natural language expressions that can be handled by the simulator.

Finally, the maximum number of turns in a dialogue is set to 100 and, if the user-agent can not achieve its goal in 100 turns, the user-agent decides to quit the dialogue.

5.5 Evaluating Proactive Dialogues

The working hypothesis of this paper is that proactive behaviors help to improve both the *effectiveness* of the dialogue, i.e., the capacity of the user to accomplish his goals, and the dialogue *efficiency*, i.e., accomplishing the task with shorter dialogues. In our simulation framework we say that a task has been accomplished when a match with the user needs is found within a certain number of turns (fixed to 100), and we measure efficiency with the number of turns of the dialogue.

5.6 Limits of the Current Simulator

The current implementation of the proactive simulator has a number of limitations, which affect our capacity to carry on simulations. The most relevant is that SimDial allows for system initiative dialogues only. Secondly, it is not trivial to integrate a proactive response within the simulator dialog policy. As an example, it happens that while the proactive response for the current dialog state shows that there is no match in the KB, the system would still continue to ask for further informable slot values from user before it makes the KB query.

An example of a simulated proactive dialogue is shown in Figure 2. Although with the above limitations, the current simulator allowed to run a number of experiments, reported in the next Section.

6 Experiments and Results

In this section we describe the dialogue configurations that we have simulated, we report the obtained results, and finally we discuss our achievements.

6.1 Experimental Setting

We simulated task-oriented proactive dialogues for restaurant booking. The default configuration is

Sys: Welcome to restaurant recommendation system. How can I help?

User: Hi. I need a restaurant.
Sys: What type of restaurant?. Most of the restaurants are Korean

User: Chinese food.
Sys: Which city are you interested in?. Most of the Chinese restaurants are in Austin and are cheap.

User: At San Jose.
Sys: What's the average price? All Chinese restaurants in San Jose are in cheap pricerange.

User: cheap price.
Sys: Restaurant 62 is a good choice. Are you happy about my answer?

User: No more questions. Thank you.
Sys: See you next time..

Figure 2: A simulated dialog using the simulator for the restaurant domain.

as described in Section 5.1. In addition, we simulated four levels of domain complexity (see Section 3.5) varying the number of slot values for the three slots of the ontology. As for the size of the KB, we simulated six configurations (including the default), ranging from 50 to 1000 instances. For each configuration the set of proactive units included in the *KB-stats* has been fully recalculated.

As for system proactivity, we set the average amount of proactive units per turn k to 1 and vary the system proactive attitude a in the range $[0, 1]$. User influenceability has been simulated considering ten values in the probability range $[0, 1]$ with increments of 0.1.

The estimation of a failure expectation, see Section 4.6, given that the simulated KB has been populated randomly, the failure expectation has been simply calculated considering the KB size for a fixed ontology: the bigger the KB the less failure expectations are expected.

Dialogue effectiveness is calculated on the number of turns of the dialogue. A simulated dialogue stops either when a restaurant is found, or when a maximum of 100 turns is reached. Finally, for each experimented configuration, we simulate 1000 dialogues.

6.2 Results of the Simulations

The goal of the simulation is to highlight the dependencies between the proactivity aspects described in Section 4. We are interested to test how dialogue effectiveness changes as a function of system

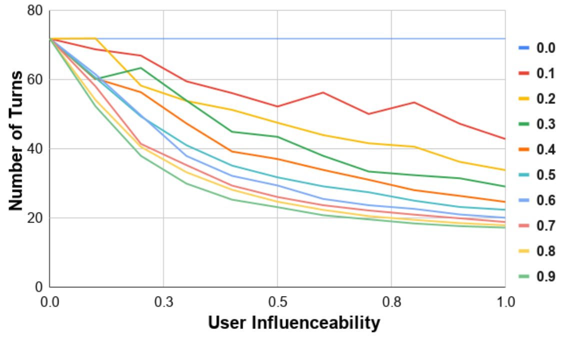

Figure 3: Dialogue effectiveness w.r.t the system proactivity attitude and the user influenceability. Numbers in legend indicate the system proactivity rate.

proactive attitude, user influenceability, domain complexity and failure expectations.

Dialogue effectiveness as a function of user influenceability and proactivity attitude. In this experiment we keep constant the ontology and the KB (using the default configurations), and we vary user influenceability and system proactivity attitude. Figure 3 shows for each proactivity attitude against the user influenceability values in the range $[0, 1]$. We notice that dialogue effectiveness increases (i.e number of turns decreases) with the increase of both proactive attitude and user influenceability. As expected, it can be noticed that a highly influenceable user (probability 1) results in the least number of turns in the conversation. While such highly influenceable users are quite unrealistic, and maybe undesired behaviour, we can infer that a moderately influenceable user (probability 0.5), would nevertheless result in a considerable reduction of unwanted interactions, a situation that helps the user to achieve the dialogue goals more effectively.

Dialogue effectiveness as a function of failure expectation and user influenceability. In this experiment we keep constant the ontology (using the default configuration) and the proactivity attitude (set to 1), and we vary the size of the KB and the user influenceability. Figure 4 shows the effectiveness of the proactive attitude over different failure expectation values (i.e., KB size). We can infer that, in case of a small KB, a user would struggle to find the desired results (i.e., 24 dialogue turns on average), as there is high probability that his/her constraints do not match with the KB instances. However, we notice that the number of turns considerably decreases even if the user con-

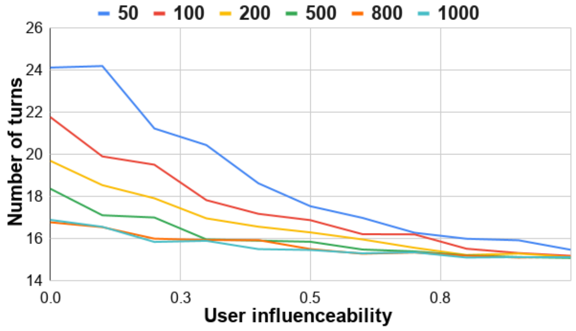

Figure 4: Dialogue effectiveness w.r.t the complexity of domain KB and the user influenceability. Values in the legend indicate the number of instances populated in the KB.

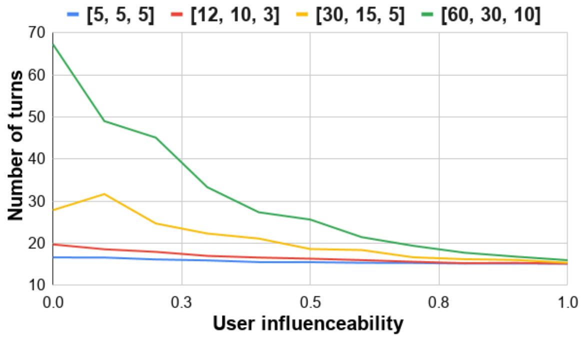

Figure 5: Dialogue effectiveness w.r.t the complexity of domain ontology and the user influenceability. Numbers in legend indicate the number of values for each of the three informable slots.

siders only half of the system proactive responses (i.e., users influenceability of 0.5). This is promising particularly for domains with a very small KB, although with a range of possible user constraints.

Dialogue effectiveness as a function of domain complexity and user influenceability. In this experiment we keep constant the KB size (using the default configuration of the KB with 200 instances) and the proactivity attitude (set to 1.0), and we vary the ontology complexity (varying the number of values for each slot), and the user influenceability.

We recall that for certain domains the number of values for a given slot could be unbounded (e.g., *time of departure*) and it could be exhausting for a user to find the KB result that matches his/her constraints without having any feedback or proactive behaviour from the system. Figure 5 shows that increasing the domain complexity, the number of turns necessary to achieve a goal increases, and this number can be drastically reduced by following a proactive behaviour.

Both proactive attitude and the user influenceability have the same effect on the dialogue effectiveness: the first generates proactive units, while the second filters them out, both with a certain probability. For this reason we do not present the corresponding experiments.

6.3 Discussion

In this section we discuss the two research questions at the core of the paper, in the light of the simulations that we carried on.

Which are the features that affect a proactive behaviour in task-oriented dialogues? The aspects proposed in Section 4 proactive unit, system proactivity rate, user influenceability and complexity of the application domain are all relevant for describing proactive dialogues. Particularly, the notion of proactive unit has been shown to be essential to make operative our simulation of proactivity.

How can we model such features within the architecture of current dialogue systems? While the current modeling of task-oriented dialogues provides a general framework, specific extensions were necessary to model our intuitions about the attitude of the user (i.e., influenceability) and the role of the system (i.e., proactivity). However, simulating proactivity revealed itself as more complex than initially expected, and, due to the issues described in Section 5.6, we had to adopt quite simplistic solutions.

How can we show the impact of proactivity on task-oriented dialogues, in terms of effectiveness and efficiency of the dialogues? Although we are aware that an effective evaluation should involve human judgements, the simulations that we conducted allow to show that the proactivity attitude of the system and the user influenceability have similar effect on the dialogue, and that they act as multipliers of the dialogue effectiveness. On the other side, failure expectation and domain complexity capture orthogonal properties of proactivity. Through our simulations we obtained quantitative evidences that small KBs and complex ontologies are the main motivations behind the need of proactivity. This conclusion, together with our predictive data, may have relevant impact on the practical construction of proactive dialogue systems.

7 Conclusion

We propose to extend the current task-oriented framework to manage proactive behaviours of the system. We analysed four aspects: (i) the degree of proactivity provided by the system during a dialogue; (ii) the propensity of the user to be influenced by the system proactivity; (iii) the complexity of the application domain, and (iv) the failure expectation of a dialogue goal. We conducted a series of simulation experiments, which allowed to investigate our intuitions about the role of the four proactivity aspects with respect to the effectiveness of the dialogue.

As for future work, we are currently working along three directions. First we intend to further extend SimDial, to be able to simulate more complex proactive situations: particularly mixed initiative dialogues and more complex domain ontologies. Second, we will conduct a comparison of the simulated dialogues based on human judgements, in order to have a qualitative evaluation. The third research direction is about modeling proactive dialogue policies, to be used within current neural dialogue architectures.

References

James F. Allen, George Ferguson, and Amanda Stent. 2001. An architecture for more realistic conversational systems. In *Proceedings of the 6th International Conference on Intelligent User Interfaces, IUI 2001, Santa Fe, NM, USA, January 14-17, 2001*, pages 1–8. ACM.

Dan Bohus and Alexander I. Rudnicky. 2009. The ravenclaw dialog management framework: Architecture and systems. *Comput. Speech Lang.*, 23(3):332–361.

Antoine Bordes, Y-Lan Boureau, and Jason Weston. 2017. Learning end-to-end goal-oriented dialog. In *ICLR*. OpenReview.net.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. MultiWOZ - a large-scale multi-domain wizard-of-Oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026, Brussels, Belgium. Association for Computational Linguistics.

Qibin Chen, Junyang Lin, Yichang Zhang, Ming Ding, Yukuo Cen, Hongxia Yang, and Jie Tang. 2019. Towards knowledge-based recommender dialog system. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1803–1813, Hong Kong, China. Association for Computational Linguistics.

Konstantina Christakopoulou, Filip Radlinski, and Katja Hofmann. 2016. Towards conversational recommender systems. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 815–824, New York, NY, USA. Association for Computing Machinery.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, et al. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190*, pages 12–16.

BJ Fogg. 2002. Persuasive technology: using computers to change what we think and do. *Ubiquity*, 2002(December):5.

Herbert P Grice. 1975. Logic and conversation. In *Speech acts*, pages 41–58. Brill.

Curry I. Guinn. 1996. Mechanisms for mixed-initiative human-computer collaborative discourse. In *34th Annual Meeting of the Association for Computational Linguistics*, pages 278–285, Santa Cruz, California, USA. Association for Computational Linguistics.

T. Hiraoka, Y. Yamauchi, G. Neubig, S. Sakti, T. Toda, and S. Nakamura. 2013. Dialogue management for leading the conversation in persuasive dialogue systems. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 114–119.

Takuya Hiraoka, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2014. Construction and analysis of a persuasive dialogue corpus. In *5th International Workshop on Spoken Dialog Systems (IWSDS)*, Napa, California, USA.

Wenqiang Lei, Xisen Jin, Min-Yen Kan, Zhaochun Ren, Xiangnan He, and Dawei Yin. 2018. Sequicity: Simplifying task-oriented dialogue systems with single sequence-to-sequence architectures. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 1437–1447.

Xiang Li, Lili Mou, Rui Yan, and Ming Zhang. 2016. Stalematebreaker: A proactive content-introducing approach to automatic human-computer conversation. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 2845–2851. IJCAI/AAAI Press.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2017. Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1777–1788. Association for Computational Linguistics.

Elnaz Nouri and David Traum. 2014. Initiative taking in negotiation. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 186–193, Philadelphia, PA, U.S.A. Association for Computational Linguistics.

P. J. Price. 1990. Evaluation of spoken language systems: the ATIS domain. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990.*

Filip Radlinski and Nick Craswell. 2017. A theoretical framework for conversational search. In *Proceedings of the 2017 Conference on Conference Human Information Interaction and Retrieval, CHIIR 2017, Oslo, Norway, March 7-11, 2017*, pages 117–126. ACM.

Yueming Sun and Yi Zhang. 2018. Conversational recommender system. In *The 41st International ACM SIGIR Conference on Research Development in Information Retrieval*, SIGIR '18, page 235–244, New York, NY, USA. Association for Computing Machinery.

Cynthia A. Thompson, Mehmet H. Göker, and Pat Langley. 2004. A personalized system for conversational recommendations. *J. Artif. Int. Res.*, 21(1):393–428.

Xuewei Wang, Weiyan Shi, Richard Kim, Yoojung Oh, Sijia Yang, Jingwen Zhang, and Zhou Yu. 2019. Persuasion for good: Towards a personalized persuasive dialogue system for social good. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5635–5649, Florence, Italy. Association for Computational Linguistics.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gasic, Lina M. Rojas Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 438–449. Association for Computational Linguistics.

Wenquan Wu, Zhen Guo, Xiangyang Zhou, Hua Wu, Xiyuan Zhang, Rongzhong Lian, and Haifeng Wang. 2019. Proactive human-machine conversation with explicit conversation goal. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3794–3804, Florence, Italy. Association for Computational Linguistics.

Rui Yan and Dongyan Zhao. 2018. Smarter response with proactive suggestion: A new generative neural conversation paradigm. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, page 4525–4531. AAAI Press.

Fan Yang and Peter A Heeman. 2007. Exploring initiative strategies using computer simulation. In *Eighth Annual Conference of the International Speech Communication Association.*

Koichiro Yoshino and Tatsuya Kawahara. 2015. *News Navigation System Based on Proactive Dialogue Strategy*, pages 15–25. Springer International Publishing, Cham.

S. Young, M. Gašić, B. Thomson, and J. D. Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

Tiancheng Zhao and Maxine Eskenazi. 2018. Zero-shot dialog generation with cross-domain latent actions. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 1–10, Melbourne, Australia. Association for Computational Linguistics.