University of Stavanger

Faculty of Science and Technology

MASTER'S THESIS

Study program/ Specialization: Data Science	Spring semester, 20.22 Open / Restricted access
Writer: Jafar Afzali & Aleksander Mark Drzewiecki	………………………………………… (Writer's signature)
Faculty supervisor: Krisztian Balog External supervisor(s):	
Thesis title: Context Modeling for User Simulation for Conversational Information Access	
Credits (ECTS): 30	
Key words: User Simulation Conversational Information Access	Pages: ..102………………… + enclosure: ………… Stavanger, .June 15, 2022 Date/year

Faculty of Science and Technology
Department of Electrical Engineering and Computer Science

Context Modeling for User Simulation for Conversational Information Access

Master's Thesis in Computer Science
by

Jafar Afzali and

Aleksander M. Drzewiecki

Internal Supervisor

Krisztian Balog

External Supervisors

External Supervisor 1

External Supervisor 2

Reviewers

Reviewer 1

Reviewer 2

June 15, 2022

Abstract

Conversational Information Access systems are systems that help satisfy the information needs of their users via multi-turn conversations. These systems are different from traditional information access systems as they can ask clarification questions that can help narrow down the result set.

Traditionally, end-to-end evaluation of such systems has been online, i.e., hiring workers to use their system and provide feedback. However, this approach suffers from key issues: it is expensive, time-consuming, and does not scale to large amount of users. Consequently, the idea of simulating users has received considerable attention. User simulators can be used for evaluation, however, their performance is dependent on sophisticated user modeling.

In this thesis, we continue on previous work [1] and build a user simulator that incorporates more advanced user modeling by including additional contexts. The simulated users are represented via an interaction model, a preference model, time context, and group setting.

We evaluate our user simulator based on metrics in the literature. Furthermore, we capture subjective measures by computing user satisfaction for each conversation. Our work illustrates that the included contexts have an impact on the observed dialogs in terms of these metrics. Lastly, we release DialogueKit on PyPi, a toolkit for developing conversational agents.

Acknowledgements

We would like to thank the University of Stavanger for the past five years.

We would like to express our deep gratitude towards Krisztian Balog, Professor of Computer Science and head of IAI group, for his indispensable guidance and supervision throughout the thesis. Thank you for answering our questions eagerly and patiently during our weekly meetings. It is not an understatement that this thesis would not have been possible without your help.

At last, we would like to extend our gratitude to our parents, family, and friends for their encouraging words and support.

Contents

Chapter 1

Introduction

Information access systems (e.g., Google) are systems that aid users in finding information. With the vast amount of information that is available on the internet, systems that enable users to find relevant information promptly are in high demand. Google revealed in 2016 that their search engine serves trillions of searches per year.[1] In recent years, we have witnessed the emergence of systems that support conversational information access, e.g., Alexa, Google Assistant, Cortana, AliMe, etc. A Conversational Information Access (alternatively Conversational Information Seeking) (CIA/CIS) system shares the same main task as traditional information access systems, however, the system-user interactions are conversational [2].

Traditional information access systems are typically developed and fine-tuned on offline datasets, before deployment. However, this is not necessarily a reasonable approach for conversational information access systems. For example, consider a dialog between a user and a conversational recommender system. The user is not necessarily looking for a specific item (e.g., a specific restaurant), but is instead open to exploring their options. That is, the specific path a conversation takes is dependent on the interactions between the user and the system, which renders the idea of building offline datasets unfeasible.

It is more interesting to evaluate CIA systems based on users' satisfaction with the system. Generally, the approach consists of hiring crowd workers to use the system followed by a survey reflecting their experience with it. For example, the survey could include questions such as *On a scale from 1-5, how satisfied are you with the system?*, or *Please rate the system's conversational abilities with respect to fluency, understandability, and grammar on a scale of 1-5.*

[1]https://searchengineland.com/google-now-handles-2-999-trillion-searches-per-year-250247

This type of offline evaluation suffers from the same limitations as online evaluation: It is expensive, time-consuming, does not scale and is not repeatable. Consequently, researchers have looked at user simulation as a compromise. User simulation addresses the limitations of online evaluation as it is considerably quicker, the results are reproducible and there is no real cost to it. The purpose of a user simulator is to enable an automatic evaluation of a CIA system that is indicative of an online evaluation. This means that the user simulator must provide realistic simulations, i.e., it should produce situation-dependent responses that are similar to what real users would produce. The realisticity of a user simulator can be measured by conducting a *Turing test*-like evaluation on a crowdsourcing platform [1]. The idea is that if human evaluators are not able to consistently distinguish between *real user* dialogs and *simulated user* dialogs, then the user simulator has successfully passed the test, and can be regarded as realistic.

However, this is also where the challenge lies: How can user simulators become more realistic? Collecting and manually annotating a large corpus of human conversations is not a practical solution, as we have mentioned. Instead, simulators have access to little or no data. Agenda-based user simulators require a sample of dialogs between users and the system in order to operate. On the other hand, model-based user simulators depend on corpora of conversations to generate realistic responses.

In this thesis, we focus on improving the realisticity of an agenda-based user simulator for conversational recommender systems. We attempt this by extending underlying the user models with additional contexts (temporal, relational and conversational), consider personal characteristics, and their interplay (patience and cooperativeness) and also take into account the user's satisfaction with the system.

1.1 Objectives

As our thesis is an extension of the work presented in [1], the overarching objective of our thesis remains the same. Specifically, we want to create a user simulator that (1) is capable of producing situation-aware responses that resemble real user responses and (2) can compute an automatic evaluation of a conversational agent. Zhang and Balog [1] focused on interaction and preference modeling. Our aim is to find out whether more advanced modeling (of context and of user characteristics (persona)) can lead to more realistic simulations. We identify the following two Research Questions (RQs):

- RQ1: Does the additional context have any impact on the characteristics of conversation, and if so, what is the observed impact?

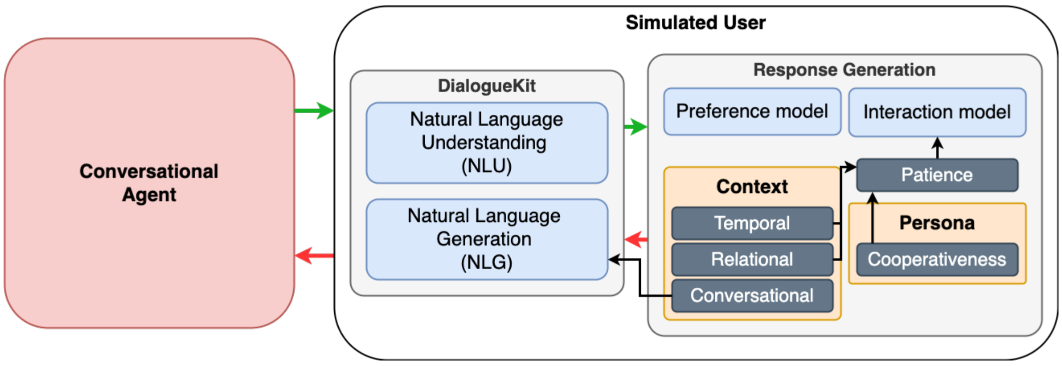


Figure 1.1: Overview architecture of the user simulator

Specifically we hypothesize that the conversations will be longer with additional context and thus lead to more successful conversations in terms of task accomplishment.

- RQ2: Does more advanced context modeling lead to more realistic user simulation? Specifically, given a persona and varying contexts, are the observed effects realistic?

To answer this, we will vary the temporal and relational context of the simulated persona and observe differences across the simulations.

1.2 Approach and Contributions

In order to answer the above-mentioned questions, we build on, and extend further two previously initiated projects: *DialogueKit* and *UserSimCRS*. Briefly explained, DialogueKit is a toolkit for developing conversational agents, and UserSimCRS is a user simulator built on top of DialogueKit, see Fig. 1.1.

Specifically, we extend DialogueKit with functionality that is required by any conversational agent: Natural Language Understanding (NLU), Natural Language Generation (NLG), and other important utilities that allow for seamless processing of conversational data. Furthermore, we modify the current implementation of preference modeling in UserSimCRS in order to get more consistent simulations, i.e., the preferences of an instantiated user should be consistent across all simulations. Next, we extend the context modeling in UserSimCRS to include temporal, relational and conversational context. We additionally develop a persona generator module that we use to instantiate the simulator. More specifically, we generate a *Persona* that includes static user properties, i.e., user cooperativeness and initial satisfaction level and sample a random context. The persona-context pair is then used to instantiate the simulator. The interplay between the persona and the context is used to define the patience of the simulated user, that is, how many times the user will retry an action before giving up on that action. The conversational context changes throughout the conversation and allows us to model the user's satisfaction with the agent.

Thus, the following contributions are made in this thesis:

- *User modeling*: We extend the current user models to include three new dimensions of context: temporal (time and day of week), relational (group setting) and conversational (satisfaction). Additionally, we modify the existing implementation of preference modeling to enable more consistent and reproducible simulations.
- *DialogueKit*: A toolkit that supports development of conversational agents. We release *DialogueKit* on PyPi.[2]
- *UserSimCRS*: An agenda-based user simulator supporting advanced user modeling, including interaction-, preference- and context modeling. Additionally, simulated users are initialized with a random persona that captures other user static variables (user id, name, cooperativeness and initial satisfaction level).
- Evaluation: DialogueKit includes simple evaluation of conversational agents based on metrics in [1] and overall user satisfaction.

The code developed in this thesis can be found on Github: `https://github.com/iai-group/UserSimCRS` and `https://github.com/iai-group/DialogueKit`.

1.3 Outline

The rest of this thesis is structured as follows: *Chapter 2* presents the necessary background information, as well as previous contributions in the field of conversational information access. *Chapter 3* describes our approach and the contributions we made. In *chapter 4* we introduce our evaluation methods and our experimental setup, followed by an analysis of our findings. Finally, *chapter 5* summarizes our work and findings, in addition to providing directions for future work.

[2]`https://pypi.org/project/dialoguekit/`

Chapter 2

Related Work

In this chapter, we introduce conversational AI and the In this chapter, we briefly explain the fundamentals of conversational artificial intelligence by studying existing work. This is followed by an introduction to conversational information seeking and the categorization of said systems. We then look at current approaches to modeling conversations in conversational information access systems. Finally, we briefly mention popular approaches to evaluating conversational systems.

2.1 Conversational AI

Conversational artificial intelligence (AI) refers to systems that are capable of conversing with users in a natural way through a range of modalities [3]. Before we expand on conversational AI, it is important to define what a conversation is. We attempt to achieve this by explaining the elements that make up a conversation.

2.1.1 Elements of Conversation

The most basic unit in a conversation is an *utterance* [2]. All contiguous utterances from a single speaker form a single *turn* [2, 4]. Finally, a *conversation* consists of multiple turns between the participants. In [2], a *conversation* is defined as an interactive communication for exchanging information between two or more *participants* (i.e., humans or machines) that involves a sequence of interactions. The types of *interaction* include natural language, click, touch, gestures, etc. This is important as interaction defines the most fundamental requirement for conversational AI systems.

In order for conversations to be as natural as possible, conversational systems try to mimic human conversations. Jurafsky and Martin [3] list subtle characteristics of human conversations such as *endpointing*, *grounding*, *mixed-initiative*, *implicature*, and various *speech acts*. *Endpointing* refers to the task of detecting *when* a speaker is finished speaking. *Grounding* is often used by the hearer to acknowledge the meaning of the speaker's utterance. A core element of human conversations is *mixed-initiative*, which refers to the shift of initiative between participants in a conversation, i.e., all participants can be active and lead the conversation. *Implicature* is used in a conversation to imply information that is understood by the hearer through a set of maxims, e.g., relevance. Each utterance in a conversation can be thought of as an action by the speaker. These actions are called *speech acts* and are often structured in pairs in a conversation, e.g., *questions* are generally followed up by an *answer*, *proposals* are either *rejected* or *accepted* and so on. However, this is not always the case, as the pairs can be separated by a *sub-dialogue* such as asking *clarification questions* instead of answering when a user asks a question. These characteristics of human conversations are complicated and are among the reasons why it is difficult to build a conversational system that can carry on natural conversations with humans. Conversational systems facilitate this by adopting a *dialogue manager* (also known as *dialogue state tracker*) which tracks the state of the dialogue and updates the necessary parameters. We discuss this in more detail in the following section.

2.1.2 Categorization of Conversational Systems

Traditionally, conversational AI systems are categorized into two classes, goal-driven (or task-oriented) or non-goal-driven (chatbots) [5]. Goal-driven systems aim to assist the user in completing a specific task, e.g., booking a flight or reserving seats in a restaurant. The tasks are constrained within some domain(s) and the systems are customized accordingly. On the contrary, non-goal-driven systems are mostly used as social "chit-chat" chatbots. Systems of this type do not have a specific intent, rather they are used to engage users in an extended conversation about various topics. As such, it is natural to regard these systems as AI companions. In the early days of conversational AI, most chatbots were built on rule-based systems, i.e., systems followed a defined set of rules [6, 7]. However, as research continues to advance the field of AI, recent chatbots are also corpus-based, meaning that they learn how to hold conversations from large datasets, e.g., human-human conversations [3]. Recently, interactive question answering (QA) has been recognized as a separate category, resulting in a 3-way categorization of conversational AI systems [5, 8, 9]. Indeed, interactive QA does not completely fit in either of the traditional categories: the dialogues are unstructured and do not follow

Table 2.1: Categorization of conversational AI systems, from [5].

	Task-oriented	**Social chat**	**Interactive QA**
Goal	Aim to assist users to solve **a specific task** (as efficiently as possible)	Aim to **carry on an extended conversation** ("chit-chat") with the goal of mimicking human-human interactions	Aim to **provide concise, direct answers** to user queries
Dialogue structure	Dialogues follow a **clearly designed structure** (flow) that is developed for a particular task in a closed domain	Developed for **unstructured, open domain** conversations	Dialogues are **unstructured**, but commonly follow a question-answer pattern; mostly **open domain** (dictated by the underlying data)
Evaluation	Well-defined measure of performance that is explicitly related to **task completion**	Objective is to be **human-like**, i.e., able to talk about different topics (breadth and depth) in an engaging and coherent manner	Evaluated with respect to the **correctness of answers** (on the turn level

any particular pattern, unlike goal-driven systems. The goal of interactive QA systems is to provide succinct and straightforward answers to questions. A comparison of the mentioned categories is provided in Table 2.1.

2.1.3 Developing Conversational Systems

In order to learn from human-human conversations, systems need to adopt and combine various techniques from different fields of computer science. These include for example natural language processing (NLP), machine learning (ML), dialogue systems (DS), recommender systems (RecSys), human-computer interaction (HCI), and information retrieval (IR). Each of these fields contributes to different parts of the system. NLP focuses on language analysis, both for understanding natural language and generating responses. Alternatively, techniques from the IR field may be used to retrieve the best responses from a corpus given the current dialogue context. ML models may then be used to rerank these responses.

When designing a dialogue system, there are several ethical concerns that need to be taken into consideration. These vary on the application domain, target user group, and the goal(s) of the system. It is therefore important to gain an understanding of the domain, in order to identify possible ethical concerns. Due to conversational AI systems requiring personal information, one of the biggest concerns that apply to most, if not all systems, is privacy. These privacy issues are related to the collection and usage of user data. Another concern is the safety of users. Systems must be careful of giving incorrect advice in safety-critical situations, e.g., medical situations. Recently, one of the most popular chatbots, Alexa, gave such a recommendation.[1] Other concerns include chatbots' reaction to toxic language (e.g., sexual harassment) and their ability to harm users through responses.[2]

Recently, Amazon released Alexa Conversations [10–12], an AI-driven dialog management model that assists agents in responding to a broad variety of phrases and unexpected conversational flows. In order to use Alexa Conversations (AC), the system developer must provide a set of annotated dialogues. These dialogues are generalized by the *dialog simulator* component of AC. By generalizing the annotated dialogues, the dialog simulator is able to cover alternative ways the user might interact with the agent. This generalization is achieved through expanding slot types, API definitions, utterance sets, and responses in the training data, resulting in a much larger set of dialogue variants. Then, these synthetic dialogue variants are used to train deep learning neural networks, such as transformer models, RNNs, etc. The trained model can then predict incoming events in the dialogue, similar to how user simulators predict the next action to perform.

2.2 Conversational Information Seeking

Conversational Information Access (CIA), also often referred as *Conversational Information Seeking* (CIS), is associated with conversations (typically between users and an information system) in which the goal is to satisfy the information need of the inquirer [2, 13]. CIS differs from traditional information seeking as it defines interactions as strictly conversational, while the latter includes other types of interaction, such as reading a book [2]. Zamani et al. [2] define a CIS system as a system that satisfies the information needs of one or more users by engaging in information seeking conversations. Furthermore, these systems are expected to output responses that are concise, fluent, stateful, context-aware, and personalized. Additionally, there should be a mixed-initiative in the conversation, meaning that the system should be able to respond to users as well as take initiative and lead the conversation [2, 5].

In the literature, CIS systems are often categorized into three goals: conversational search (ConvSearch), conversational recommendation (ConvRec), and conversational question answering (ConvQA) [2]. The former two subdomains can be regarded as task-oriented systems, where the goal is to assist users in finding information. The latter is related to interactive QA systems. However, the boundary between these subdomains is often blurred and there is no clear distinction. For example, a system that retrieves passages

[1]https://t.co/HgGgrLbdS8

[2]In 2016, Microsoft's AI chatbot Tay learned toxic language from users: https://www.theverge.com/2016/3/24/11297050/tay-microsoft-chatbot-racist

in order to answer a series of questions from the user can be seen as either ConvSearch or ConvQA. Similarly, helping the user in finding items can be seen as ConvSearch or ConvRec. This suggests that CIS systems should support multiple user goals as mentioned in [5]. Gao et al. [8] mention that one way of supporting multiple user goals is to introduce a "top-level bot" that can switch between different user goals (i.e., choose which agent should serve the user based on user goal). Conveniently, the different systems share a lot in common in terms of conversational requirements.

Apart from being capable of mixed-initiative, for a system to be truly conversational, the interactions between the user and the system should take place over multiple turns, in which the user states and clarifies their information need. The system in turn can e.g., ask clarifying questions or affirm the users' intent via grounding. Consequently, the conversation seems more realistic. Furthermore, the system should respond in a way that shows it is context-aware, and the responses should be personalized to the user. As we mentioned previously, this can be facilitated by implementing a *dialogue manager* (DM) which tracks the state of the conversation, updates user models, and decides on which action the system should take next. In the following section, we discuss the three goals (ConvRec, ConvSearch and ConvQA), with an emphasis on ConvRec as this is the main focus of this thesis.

2.2.1 Conversational Recommendation

Traditionally, recommender systems exploit user interaction history to recommend relevant items to the user [2, 14, 15] and are thus often closed within a specific domain, e.g., books, restaurants, or movies [2]. Conversational Recommender Systems (CRS) however, allow for real-time interaction between the users and the system. Two recent survey papers on CRS have proposed the following definitions for CRS:

- "A CRS is a software system that supports its users in achieving recommendation-related goals through a multi-turn dialogue." (Jannach et al. [14])
- "A recommendation system that can elicit the dynamic preferences of users and take actions based on their current needs through real-time multi-turn interactions." (Gao et al. [15])

Both definitions agree that the interactions between the users and the system should be over multiple turns. This implies that a CRS must implement some kind of dialogue management module, in order to consider all user utterances when deciding which items to recommend. Further, none of the surveys assume the modality of these interactions, i.e., the system should support multi-modality, e.g., written or spoken language, point&click and/or gestures, which is different from e.g., ConvSearch, which is typically assumed to be contained within natural language interactions (i.e., written or spoken form [14]).

The specific framework or architecture of a CRS depends on which functionalities it will support, however, any CRS should have the following modules:

- User Modeling Module: The objective of this module is to correctly represent the user by utilizing past user interactions, preferences, and other relevant user information.
- Dialogue Management Module: This module takes in the processed input (i.e., intent, entities mentioned, user preferences, etc.) and updates the user model and the dialogue state. During output time, it decides on which action to execute, e.g., recommend item(s), ask questions or explain a previous utterance by the system.
- Recommender Engine Module: The main functionality of the recommender engine is to provide relevant item(s) based on the current dialogue state and user model. In some cases, it is expanded with other capabilities, such as generating explanations.

Beyond these modules, the CRS must include some kind of user interface that can give the processed input to the dialogue management module, as well as putting together the output of the system. An overview of ConvRec architecture is shown in Fig. 2.1.

2.2.1.1 Challenges

With these modules come many difficult challenges. In order for the system to model a user as accurately as possible, the system must know which questions it should ask, and how to interpret the user responses. This is labeled as *question-based user preference elicitation* in [15].

Deciding on which action to take. One of the key differences between a traditional recommender system and a CRS is the added conversational aspect. The system must deduce when to ask questions, and when to give recommendations. Asking more questions will help make a better user profile and thus better recommendations, however, the user might run out of patience and end the conversation. Thus, knowing when to ask and when to recommend is a critical challenge [15] related to the dialogue management module.

NLU and NLG. Another challenge with conversational recommender systems is related to the systems capability of understanding, i.e., natural language understanding, and

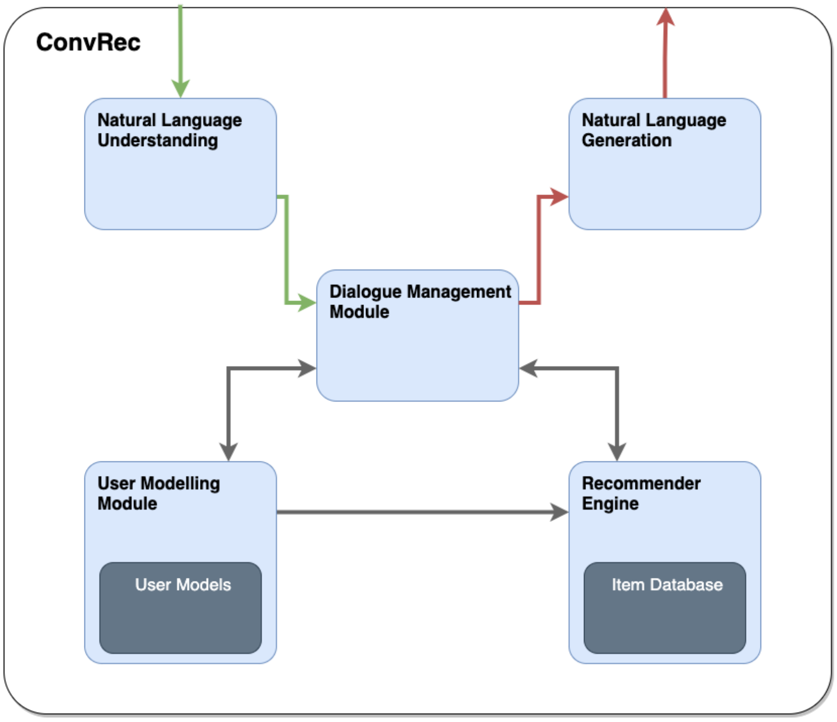

ConvRec
Natural Language Understanding
Natural Language Generation
Dialogue Management Module
User Modelling Module
User Models
Recommender Engine
Item Database

Figure 2.1: Illustration of a typical ConvRec architecture, based on [14]. The green arrows indicate an input utterance, while red arrows indicate output utterances. The grey arrows show the interaction between the modules.

its ability to generate responses that are concise and fluent, i.e., natural language generation [15]. Various techniques are implemented in the literature, from pre-defined tags that capture semantic information to slot filling techniques, where the goal is to fill semantic slots with values given in the user utterances [14, 15]. For responses, some researchers deploy end-to-end frameworks for both understanding and generating utterances [16, 17], while others use natural language templates [18].

Evaluation. Finally, evaluating static recommender systems is significantly easier than conversational recommender systems. Offline evaluation is a standard evaluation methodology for the former, however, it is not straightforward to use offline evaluation for the latter. Due to its interactive nature, evaluating the conversational aspect of the system becomes important. Unfortunately, this process generally requires human judges for evaluating the flow of the conversation, and whether the user-system interaction was successful or not. To this end, many researchers propose using simulation [14, 15].

Finding a useful dataset for conversational systems is difficult, often due to how the dialogues are collected or which domain the dataset focuses on. For conversational recommender systems, the dataset must be centered around the recommendation task, while emphasizing natural conversations. Furthermore, not all systems implement mixed-initiative. Some systems follow the "System Ask, User Respond" [19, 20] (SAUR) paradigm. In this scenario, the user is passive, and system takes the initiative. Others implement the opposite, i.e., “User Ask, System Respond.” This is also reflected in the datasets that are available.

2.2.1.2 Datasets

Sun and Zhang [21] introduce a movie dataset containing 999k dialogues with 6k users, 3.4k items and 15 facets. They use the MovieLens1M[3] dataset for rating information which is used to build user models. Furthermore, it is worth noting that their dialogues are randomly simulated based on templates collected from crowdsourcing tasks, where crowdworkers are asked to generate natural language text from a given dialogue schema. Assuming that they used the same procedure as for their restaurant dataset, they collected 385 dialogues from crowdsourcing and simulated the rest of the dialogues based on these. Furthermore, the dialogues there are based on the SAUP paradigm, i.e., “System Active, User Passive,” thus there is a lack of mixed-initiative [19].

Hayati et al. [22] present INSPIRED, a dataset containing 1k human-human dialogues for movie recommendation with measures for successful recommendations. Their dialogues are collected through crowdsourcing, where they pair up crowd-workers in a natural setting and assign each with a role. The seekers are asked to talk about movie recommendations without any strategy support. The recommenders are given tips on sociable recommendation strategies before the chat, and are tasked with gathering information about the seeker before the recommendation phase. Recommenders are also encouraged to continue the conversation until the seeker accepts a movie recommendation. Their analyses show that sociable recommendation strategies are correlated with successful recommendations in dialogues. Finally, each utterance in their dataset is manually annotated with sociable strategies.

Similarly, REDIAL consists of over 10k conversations between crowd-workers on Amazon Mechanical Turk (AMT) platform. To construct the dataset, Li et al. [23] paired crowd-workers and assigned each with a role; the movie seeker has to explain what kind of movie they like, and asks for movie suggestions. Then, the recommender suggests movies based on their understanding of the seeker’s movie preferences. Critically, the workers were required to recommend at least 4 movies, keep the conversation for roughly 10 turns and to use formal language. Additionally, they were asked to not mention the AMT/task explicitly. Although similar to INSPIRED, there are some differences according to the authors of [22]: the recommendations in REDIAL are conditioned on the movies mentioned in the dialogue, and not directly on the language usage. Furthermore, in

[3]MovieLens 1M dataset: `https://grouplens.org/datasets/movielens/1m/`

REDIAL they tend to only mention movie names rather than an in-depth discussion on the movie preferences.

Radlinski et al. [24] focused on movie preference elicitation through natural interactions between a pair of crowd-workers following the Wizard-of-Oz methodology. They created CCPE-M (Coached Conversational Preference Elicitation dataset for Movies), a dataset focusing on preference elicitation rather than recommendation. Furthermore, these conversations were coached, i.e., the assistant (or Wizard) asks pre-designed questions in order to minimize the bias in the terminology the "user" (or requester) employs to convey their preferences and to obtain these in as natural language as possible. Each dialog is annotated with entity mentions, preferences expressed about entities, descriptions of entities provided, and more.

Finally, Dodge et al. [25] presents a ConvQA + ConvRec dialogue dataset consisting of 1M conversations, where each participant (user and system) takes three turns, resulting in a total of six turns. The historical context of the user is given in the form of five previously liked movies. Then, the first exchange of the conversation is a recommendation task, where the user informs the system of a genre or topic that they like. This is followed by a QA exchange, where the user asks a factoid question about the suggestion. The third and final exchange consists of the user asking for an alternative recommendation while revealing additional preferences. However, as each conversation is restricted to 6 turns, this dataset is not fit for systems that aim for an extended conversation with the user. Further, their dialogues are generated using a fixed set of natural language templates. More specifically, they filter the MovieLens dataset and select a user at random. Then, they sample 1-8 movies which the user has rated as 5 (highest score) and create a statement expressing the users' sentiment about these movies. A recommendation is deemed successful if the user has seen the recommended movie and assigned it a high score.

2.2.2 Conversational Search

Conversational search is the process of interacting with a conversational system through natural conversations to search for information [2]. Improvements in the aforementioned fields of AI and growing computational power allow for highly interactive systems that go beyond the traditional "query box" search model [26]. Users can express their information need and receive help from the system in sifting through the information that is available, through natural conversation. Conversational search has received significant attention in recent years accompanied by the development of benchmark datasets such as MSDialog [27] and TREC CAsT. [4]

The TREC Conversational Assistance Track (CAsT) is an initiative that aims to facilitate research within CIS and to create a reusable test collection for conversational search systems [28–30]. In both 2019 and 2020, the CAsT dataset included MS MARCO and CAR, resulting in $38,426,252$ passages. The CAsT-19 dataset includes 80 information-seeking dialogs, split into train (30) and test (50). On the other hand, CAsT-20 has only 25 information-seeking dialogs. Different from the previous two years, the CAsT-21 dataset consists of MS MARCO and Wikipedia – the KILT dump [31]. Additionally, CAsT-21 included 26 information-seeking dialogs. Over the past three years of the TREC CAsT, a canonical approach including query reformulation, multi-stage retrieval and reranking has emerged [2]. Yang et al. [32] achieved the highest score in terms of NDCG@3 (the main metric) in 2019. They expanded the MS MARCO documents with queries, and used BM25 for initial retrieval, followed by a BERT reranker model. CAsT 2020 saw two teams being tied for the best score. Pradeep et al. [33] employed a T5 query reformulation model fine-tuned on the CANARD dataset [34] before passing the queries into a hybrid dense-sparse model for retrieval. Finally, they used another T5 model fine-tuned on MS MARCO for the reranking stage. Chang et al. [35] had a similar approach. In addition, they expanded the queries using keyword distillation, RM3 from Anserini and sentence-level query extraction. However, they used BM25 for initial retrieval, followed by a T5 reranker model. Overall, h2oloo's [33] approach scored highest, although both approaches had the same NDCG@3 score. H2oloo group continued their success in CAsT 2021, where they also achieved the best score.

2.2.3 Conversational QA

Conversational QA (ConvQA) differs from traditional one-shot QA systems in that questions can build on previous questions. Generally, the initial question will be well-formed and complete, however, follow-up questions will often have missing entities and predicates [36]. Additionally, they may implicitly and/or explicitly refer to previous system responses. Thus, the conversational aspect of ConvQA brings new challenges and requirements to such systems. More specifically, systems should retain the previous question-answer pairs as they can provide context for future questions. Another major challenge is interpreting utterances, as ellipsis phenomena appear frequently [37].

ConvQA systems are often based on either a set of knowledge bases (KB), e.g., DBPedia, YAGO, etc., or on text collections [8, 36]. The procedures are different for the two;

[4] https://www.treccast.ai/

ConvQA systems that utilize knowledge bases (KG-ConvQA) often use semantic parsing in order to map the question into its formal meaning e.g., logical form, while text collection-based methods (OR-ConvQA) operate on a large corpus of passage text [2]. OR-ConvQA typically consists of a retriever and a reader. The retriever is similar to the systems discussed in the previous subsection, Sect. 2.2.2. The reader is responsible for extracting or generating an answer from the retrieved passages of the retriever. Similar to the case for the retriever, the reader is often a BERT based model. Systems that generate answers instead of extracting also have to consider the dialogue context and ground the answers accordingly.

Two of the popular datasets for ConvQA systems are the CoQA and QuAC dataset. Reddy et al. [38] present CoQA which contains over 8k conversations consisting of an average 15 turns, where a turn is noted as a (question,answer) pair. The text passages are retrieved from multiple domains which allow for both in-domain and out-of-domain evaluation. On the other hand, the QuAC dataset contains 14k information-seeking QA dialogs [38]. The dialogs are constructed as follows: The student, which does not have access to the underlying Wikipedia text, asks questions to learn more about it. The teacher, who has access to the Wikipedia text, answers the question to the best of his ability. More specifically, if the teacher is not able to find the answer within the given text, the question is considered unanswerable. These datasets were originally designed for ConvQA systems that are grounded to a single paragraph.

However, as we have mentioned, OR-ConvQA systems operate on a large corpus, thus rendering these benchmarks as non-applicable. This led to the extension of the original QuAC dataset, named OR-QuAC, to integrate passage retrieval over Wikipedia. More specifically, synthetic queries were created including the Wikipedia title, initial paragraph and a rewritten question to remove ambiguities. For more details, we refer to [15, 39, 40].

Examples of datasets for KG-ConvQA systems include Wikidata [41], DBPedia [42], FreebaseQA [43] etc. Both Wikidata and DBPedia are large knowledge graphs, with Wikidata consisting of over 98 million items.[5] FreeBaseQA is a QA dataset over the Freebase [44] knowledge graph. It contains over 28000 questions divided into *train*, *dev* and *eval* sets.

2.3 Modeling Conversations

The multi-turn interactive nature of CIS systems requires the system to keep track the conversation history and what the system understands regarding the user's information

[5]Items represent anything in human knowledge, see `https://www.wikidata.org/wiki/Help:Items`

need. This section provides a brief introduction to conversation modeling, which entails the task of modeling conversation history, interaction modeling and user modeling.

Most importantly, conversation modeling entails the task of modeling previous interactions as the conversation progresses. This section provides a brief introduction to modeling conversations (i.e., conversational history) and previous approaches. Modeling conversations refers Conversational history and state, conversational query rewriting, intent and entity recognition, dialogue actions and user modeling.

2.3.1 Conversational History and State

Conversational history modeling refers to the past utterances that were uttered and how these are related. It is important due to the language usage in conversations; users rightfully expect the system to understand their incomplete utterances by using the context of the conversation (i.e., history). Indeed, in CIS systems, utterances are more prone to ellipses, coreferences (anaphoras and zero-anaphoras) [2], see Table 2.3. Consequently, the NLU module of any CIS system is of paramount importance. Using the context can be of huge advantage for understanding indirect answers. For example, given the question "Do you want to go to the gym together?" the prompted person could answer with "I'd like to do some calisthenic exercises outside." In this example, the user implicitly denies the offer. Simple approaches to modeling history attempt to provide context by including the previous k turns. These approaches were later expanded to also include the positional relation of the turns [45]. Furthermore, their model learns the significance of previous answers' tokens to the current answer using a History Attention Module. More recent approaches utilize transformer models, e.g., BERT, which are adept at understanding semanticity. Transformer models use attention to encode long-term conversations with separation tokens to indicate turn structure. Similarly, recurrent networks, such as LSTMs, use latent states to encode conversations. Recent approaches attempt to model the flow of the conversation, i.e., an explicit latent representation of the previous context. In the information retrieval community, approaches using dense retrieval have emerged recently. Yu et al. [46] present a Conversational Dense Retrieval (ConvDR) system that learns contextualized embeddings for queries in a conversational setting. More specifically, they use a teacher-student framework to train their model at matching query and document embeddings.

2.3.2 Conversational Query Rewriting

Conversational Query Rewriting (CQR) entails the task of leveraging conversation history to rewrite the current query. This is helpful for incorporating the conversational context

Table 2.3: Overview of frequent linguistic concepts in conversations

Concept	Description	Example
Ellipsis	omitting words or topics implied by the context	"**That** sounds logical"
Anaphora	words that explicitly refer to previous conversational turns	"**We** told you so"
Zero-anaphora	omitted anaphora	"Told you so"

into the query and allows for more successful retrieval, i.e., retrieving most relevant documents. Additionally, it has proven to be useful for reranking first-pass retrieval [29] results.

A popular approach that we have mentioned earlier is to employ sequence-to-sequence models such as T5. These models are often pre-trained on CQR datasets such as CANARD [34].

2.3.3 Intent and Entity Recognition

Intent and entity recognition is a critical part of any conversational system and makes up the NLU component. It is responsible for detecting intent(s) and entities within a user utterance, such that the system is able to act in a correct manner. Traditionally, intent detection methods include rule-based and template-based recognition methods, and statistical classifiers [47, 48]. While it is possible to build accurate classifiers with rule-based templates, they are costly in that different domains require different rules and templates. Furthermore, statistical classifiers require feature extraction which, if done manually, is ineffective and there is no guarantee on the accuracy of features [47]. More recent methods however, focus on deep learning architectures, such as Convolution Neural Network (CNN), Recurrent NN (RNN), Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), NNs with Attention mechanism (e.g., Transformer architectures). RASA [49] introduces DIET (Dual Intent and Entity Transformer), a multi-task transformer architecture which performs intent classification and entity recognition jointly. It allows for flexible choice pre-trained embeddings, such as BERT, GloVe, ConveRT, etc. Its performance [49] is comparable to large-scale pre-trainted language models, while being computationally effective [50].

2.3.4 Dialogue Actions

We have briefly mentioned speech acts (also known as dialogue acts) in Sect. 2.1, and how each utterance in a conversation can be seen as an action. These are the same as intents, but following recent work [51], will be referred to as actions. Thus, it is natural to build schemes that summarize the set of actions that the user and/or the system can take. While acknowledging previous work, Azzopardi et al. [51] attempt to conceptualize a framework for the different actions and intents that can take place in a conversation, and elaborate what they regard as key decision points within the conversation. Moreover, they develop this under the assumption of a conversational search agent. Thus, their framework, as they mention several times, is not complete and would need some adjustments to include all capabilities of a CIS system (i.e., ConvQA and ConvRec intents/actions). More specifically, Azzopardi et al. [51] identify the followings intents and actions for the user:

- **Reveal**: Reveal actions refer to utterances where the user reveals more information regarding their need, e.g., budget for a travel. This is also known as disclosing (*disclose*) information. However, there are cases where the user is either not sure, e.g., the user has not decided on a budget, or they refuse to share such information. These are known as *disclose-unsure* and *disclose-not* respectively. Further, a user can *revise* their information need, e.g., changing their budget in the middle of the session, or *refine* it. Finally, the user can also *expand* their information need, which is especially useful for cases where they have over-constrained their information need and there are none or too few options available. In general, reveal actions are used to formulate the users' query.

- **Inquire and Navigate**: In order to process the query results, the user must be able to both inquire results and navigate through these in a seamless manner. For example, the user can inquire a *list* of options that satisfy their *current information need* (referred to as CIN in [51]), or ask the agent to *summarize* the options available. Further, they might also be interested in comparing (*compare*) or selecting a *subset* of the results, or ask for similar options. The user should also be able to navigate through the options that have been revealed to them, e.g., by repeating (*repeat*) them, or saving (*note*) some particular options that have peaked their interest. Additionally, the user could ask for more information regarding one or more options (*more*), etc. Thus, while *inquire* considers the different means of presenting relevant options, *navigate* refers to further actions that the user can perform regarding these options.

- **Interrupt and Interrogate**: Sometimes, the agent could lead the conversation in a direction that is unwanted by the user. In such cases, the user can interrupt the agent and change the direction, e.g., by having the agent explain why it suggested a particular item (*explain*). The user might also be interested in knowing what the agent has understood so far regarding their information need (*understand*). Thus, interrupt and interrogate may be used to facilitate mixed-initiative.

The agent must be able to respond to the above intents and actions from the user. In this regard, [51] define these actions for the system to perform as answer to the actions performed by the user:

- **Inquire**: When a user performs a reveal action, the system must be able to *extract* relevant parts of the user utterance, e.g., constraints and criteria. Furthermore, the agent must be able to *elicit* information that is most important in filtering the result set. Systems that are deployed in already constrained domains implement this via slot-filling techniques, where the agent attempts to fill pre-defined slots with their values, e.g., "Price: $ 300." Sometimes, the agent might not be confident in its understanding of the user's utterance, which creates the need for clarification questions by the system (*clarify*). Clarification questions are often used in conversations between humans and thus make the agent-human conversations more human like. However, it comes at the price of user patience. Indeed, if asked too many times, the user might lose their patience and end the conversation, resulting in an unsuccessful conversation. Note that inquire actions by the system is often related to query formulation, similar to the set of reveal actions of the user.
- **Reveal**: At some point in the conversation, the agent will have to disclose what it has found given the CIN. In [51], they also mention past and alternative information needs (PINs and AINs respectively). Further, the system has to decide how to present the results, i.e., should it present a list of the options or a summary. Additionally, as the user could ask for a comparison, subset or similar options, the agent must be able to respond correspondingly, i.e., it should support the *compare*, *subset* and *similar* actions.
- **Traverse**: Traverse corresponds to the navigation action of the user. More specifically, the agent must remember where the user is in the list of options, be able to elaborate at the request of the user, save particular items that the user showed interest in, etc.
- **Suggest**: Although similar to the reveal action, the agent can decide to take initiative and *recommend* items despite the user not asking. This implements mixed-initiative on the system side and can lead to a natural flow in the conversation. Additionally, the system can suggest (*hypothesize*) other AINs by generating what-if questions such as "what if you (the user) travelled to a different country/place?" This leads to more options being available and may lead to a better experience. For example, if the user is willing to change location for a significantly better option, and the system fails to mention this, it could lead to regret and the user might think less of the system's capabilities.

As we mentioned earlier, this conceptual framework is incomplete, however, it provides a good starting point as it mentions several actions that are relevant across all sub-domains of CIS systems. Furthermore, we did not discuss the key decision points in the conversation, where the agent must decide on which action to perform. This is generally referred to as the agent policy and is usually implemented by the dialogue manager module as we previously mentioned.

Finally, there are other intent/action schemes, such as the QRFA [52] model. Here, the user-system interactions are summarised into four general categories; Query (user), Request (system), Feedback (user) and Answer (system). Vakulenko et al. [52] show that their QRFA model better reflects conversational flows in real information-seeking conversations than previous models. However, it should be noted that the QRFA model is an extremely coarse-grained model, thus facilitating the modeling of conversational flows to a limited extent.

2.3.5 Context Modeling

The importance of context information for recommender systems has been presented by many researchers, and it is evident that context is influential on the quality of recommendations [53–59]. Context is any information useful to characterizing the situation of an entity (e.g., a user) [60] and can come from many sources including user history, preferences, and profile [61]. Villegas and Müller [62] identify five overarching categories of context. These are: individual (i.e., information from independent users or items), location (user's location), time (during the day, week, and year), activity (e.g., shopping), and relational (i.e., relationships between the user and the environment). Although these concepts are initially presented in the field of recommender systems, they are also applicable for CRS.

2.4 Evaluation

Goal-driven and non-goal driven systems have different goals, and thus evaluated differently.

Chatbots are evaluated on their ability to hold a natural conversation, which is inherently difficult to evaluate automatically. Therefore, evaluation of non-goal driven systems usually requires human assistance/judges. There are two ways for human judges to evaluate a human-chatbot conversation, participant evaluation and observer evaluation [3]. In the former, judges converse with the chatbot and evaluate it, while in the latter, judges are given transcripts and evaluate these. Commonly used metrics revolve around a chatbot's ability to seem human.

As for goal-driven systems, one way of evaluating their performance is to measure how successful the system was at solving the given task, e.g., by asking the users [63]. Jannach et al. [14] identify four main categories of quality dimensions related to CRS:

- *Task effectiveness*: Related to CRS's ability to support its main task. Can be quantified objectively, e.g., accuracy measures and acceptance/rejection rates. Subjective evaluation can be obtained by asking users directly (e.g., satisfaction level [64], quality of recommendation [65]).
- *Task efficiency*: The efficiency of a CRS is often quantified by considering the number of turns in the dialogue until a successful recommendation [66]. Systems that require less interaction are preferred.
- *Quality of the conversation and usability aspects*: These aspects are concerned with quantifying the conversational skills of the CRS, and its ease-of-use [65, 67]. Generally achieved through subjective assessment, with regards to fluency, understandability or response quality [16, 17, 68].
- *Subtask effectiveness*: Associated with the performance of individual components, e.g., NLU [69].

Evaluation can be broadly categorized into two categories: offline evaluation and online evaluation.

2.4.1 Offline evaluation

One of the advantages of offline evaluation is the reproducibility of results. For this category of evaluation, there exists many benchmarks, some of which we have already mentioned in Sect. 2.2. One important aspect of offline benchmarks for CIS systems is that benchmarks can have either *synchronous* or *asynchronous* dialogues, or a combination of both [2]. Synchronous dialogues are live conversations, whereas asynchronous dialogues allow participants to reply at their convenience, e.g., conversations on a Reddit forum. By exploiting such free-form dialogue datasets, agents can be trained to produce a plausible response to any given utterance [2]. However, it is important to carefully consider which forums to use to train and evaluate the CIS system, as it can introduce bias to the evaluation. Furthermore, the language used in these dialogues may be inappropriate, which is undesirable for the CIS to learn. Constructing conversational datasets for offline evaluation presents several challenges. Dialogues may be heavily impacted by the motivation of the participants. For example, in a task-oriented setting, users that are hired to complete a certain task might not approach it in the same way as someone who *wants* to complete that task [2]. Furthermore, systems that are capable of solving the task, however in a different way, could be judged unjustly, especially in dialogues where a "correct" response exists.

In terms of automatic assessment, popular metric choices for end-to-end systems are BLEU [2, 14, 15, 70, 71], ROUGE [72] and METEOR [73] which are based on word-overlap, with respect to the actual user response given at the particular turn, and typically used for machine translation tasks. However, it has been shown that word-overlap based scores have a low correlation to human evaluation [70]. Other metrics such as per-turn, and per-dialogue accuracy can also be implemented [3], in addition to task completion rate, the average number of turns for completion, trust, and fairness [2]. The appropriateness of responses has also been evaluated using trained metrics [74, 75]. These have a significantly higher correlation to human judgments than word-overlap measurements such as BLEU, ROUGE or METEOR. However, it has been shown that simple manipulation (e.g., reversing word order) of the responses can lead to substantial changes in the score of trained metrics, indicating that these metrics are not robust [76].

Systems can also be evaluated on component level, e.g., evaluating the NLG component of a CIS system. Datasets that aim to evaluate single components are dubbed as "single-step datasets" in [2]. The End-to-End NLG challenge [77] aimed at evaluating end-to-end NLG systems capability of generating complex output by introducing a new dataset with challenges such as open vocabulary, complex syntactic structures and various discourse phenomena. The task showed that sequence-to-sequence models were able to perform well according to word-overlap measures and human rankings of naturalness. However, manual systems outperformed these models in the overall quality, diversity and complexity of outputs. Another fine-grained metric (for chatbots) is topic-based evaluation, in which the agents ability to talk about multiple topics is evaluated [71]. Two aspects of topic-based evaluation include topic breadth and topic depth [78]. These refer to the variety of topics the agent is knowledgeable about and the depth of knowledge (i.e., how many turns can it talk about the same topic) respectively. In [78], the correlation between human judgments and topic-based metrics was higher than the trained metrics we mentioned above, where topic depth was measured to have a correlation of 0.707 and topic breadth a correlation of 0.512.

2.4.2 Online evaluation

In online evaluation, the system is deployed and real users have the chance to interact with it. Therefore, online evaluations are naturally more indicative of how the system performs. Users' responses to system utterances can identify limitations, making this type of evaluation more robust. Zamani et al. [2] categorize online evaluation into two groups: lab or crowdsourced studies and real-world studies.

In lab or crowdsourced studies, paid users or volunteers are hired to assess the system either as a whole or a sub-module of it (e.g., preference elicitation). Workers can be tasked with e.g., providing clarification questions/answers [79], eliciting preferences, and evaluating recommendation explanations based on these preferences [80] and so on. Note also that many of the metrics mentioned under offline evaluation can be evaluated online as well, such as topic-based metrics and the agent's ability to seem human.

However, online evaluation is expensive and time-consuming to do for multiple iterations of the system - hence the research on automatic evaluation that aims to facilitate this. Experiments are also not repeatable, which makes it difficult to compare the system performance across experiments.

The evaluation of a system's overall quality, i.e., natural responses, conversation flow (such as mixed-initiative, discourse management, etc.), vocabulary, being human-like, etc., can be obtained through real-world studies. The Alexa Prize Challenge challenges academics to create conversational agents to converse with users on a variety of topics [81]. At the end of each conversation, the user is requested to rate their experience, thus providing competing teams with real user data and enabling them to continuously improve their conversational systems during the competition.

2.4.3 User simulation

Employing human judges and/or real users for evaluation of dialogue systems can be both costly and time-consuming. Therefore, user simulation has become popular. However, it remains a difficult task to create simulators that can imitate real users to a high degree. To achieve this goal, a user simulator has to accurately model users with respect to interaction space, preferences and other context such as preferences, cooperativeness, etc.

Generally, user simulators are broadly categorized into agenda-based or model-based.

2.4.3.1 Agenda-based user simulation

Agenda-based simulators build on the idea that dialogues are essentially an exchange of actions between the participants. These actions are similar to the ones we defined in Sect. 2.3.4. Through these actions, the initiating participant (often the user) wishes to achieve a certain goal. This goal is the driving factor behind the user agenda and is often described through a set of constraints. For example, consider the following utterance from a user: "I'm looking for a nice bar serving beer in the town center." [82] In this example, the user goal is finding venues with the constraints that venue type must be a *bar*, it should serve *beer* and the area should be *central*. Typically, the user is interested in metadata regarding the recommended venues, such as the phone number, address, and the name of the venue. This type of information is obtained through a set of request actions [82].

Throughout the dialogue, the agenda and goal may be updated, depending on the agent's actions. For example, the agent may *inquire* more information about a previous user action, thus pushing a new action with higher priority onto the agenda. Similarly, the goal may change (i.e., the goal is too restricted, thus prompting a goal change). In each turn, a new action is sampled from the top n actions on the current agenda. Schatzmann et al. [82] note that n can correspond to the level of initiative taken by the simulated user, which allows for incorporating user cooperativeness into the simulation. Thus, a statistical model trained on prior dialogue data may be used to sample n for each turn [82]. Naturally, such a model would be dependent on the agenda.

Zhang and Balog [1] develop an agenda-based user simulator and evaluate it by comparing real users vs. simulated users objectively and subjectively through crowdsourcing. The workers are presented with two dialogues side by side: one is a simulated dialogue while the other is a real dialogue, both with the same agent. Then, the workers have to guess which one of the dialogues was performed by a real human and provide a short explanation of their choice. They find that by introducing a more advanced interaction model and preference model, they are able to simulate more realistic conversations. Furthermore, by studying the comments they collected from the workers, they conclude that future simulators can gain substantially by focusing on these three properties: realisticity (being human-like), response (increased transparency and less repetitive), and engagement (involving the user in the conversation).

2.4.3.2 Model-based user simulation

User simulators that are model-based typically employ neural network architectures to create end-to-end, (or sequence-to-sequence) user simulators [83, 84]. These models take as input the dialogue context, which can be a combination of previous dialogue acts and states to output a set of user intents for the current turn [83], or it can be the last utterance used to generate a natural language response (NL-to-NL) [84].

Asri et al. [83] introduce a sequence-to-sequence model using an encoder-decoder RNN architecture. The encoder receives the entire dialogue history, i.e., a sequence of dialogue contexts, where each dialogue context consists of four components: a vector indicating the most recent machine acts, two vectors that indicate the inconsistency between the most recent machine information and user goal, a vector to track the status of each goal constraint and similarly for requests. Both the encoder and the decoder are based on Long-Short-Term-Memory (LSTM), followed by a fully connected layer. The output of the encoder's fully connected layer is used as input to the decoder. Finally, the decoder outputs probabilities for the action space. In order to generate a set of actions, they first sample an action from the first step of the LSTM and use this as input to the second step, repetitively. Importantly, their model is limited to only generating actions, thus necessitating a separate NLG module. Crook and Marin [84] acknowledges this limitation and introduces a sequence-to-sequence, NL-to-NL user simulator. Excluding the specific details, their model consists of a GRU-based encoder and an LSTM-based decoder. The model takes as input the system utterance and generates a user response. They develop three models with different levels of context and show that their models are close to human level in terms of both naturalness of responses and dialogue discourse cohesion. For further details, we refer to [84].

Chapter 3

Approach

In this chapter, we present our approach to building the user simulator. We start by explaining the mathematical concept upon which our simulator framework is built in Sects. 3.2 and 3.3. Then, we introduce the existing user modeling and the modifications made to it. Furthermore, we detail our additions to the user model module, i.e., context and persona. Next, we describe the process to instantiate the simulator. At the end of the chapter, we detail the implementation of our approach.

3.1 Overview

Our user simulator is based on [1], thus it is agenda-based. It consists of four main parts: interaction modeling, preference modeling, context modeling, and persona modeling. The interaction model is based on the concept of Markov Decision Processes (MDP) and built with a modular approach with respect to the underlying interaction model, allowing it to be replaced with more sophisticated models and schemes. Preference modeling facilitates the simulator to elicit preferences and is consistent for a given user. The new concepts compared to [1], are the context model and persona model. Context modeling encapsulates the temporal, relational, and conversational context of a user. While the persona encapsulates the static user properties.

All these concepts and the mathematical foundation will be described in the next sections.

3.2 Background - Mathematical foundations

The underlying mathematical concept behind the user simulator is known as Markov Decision Process (MDP). A Markov decision process is a discrete-time stochastic control

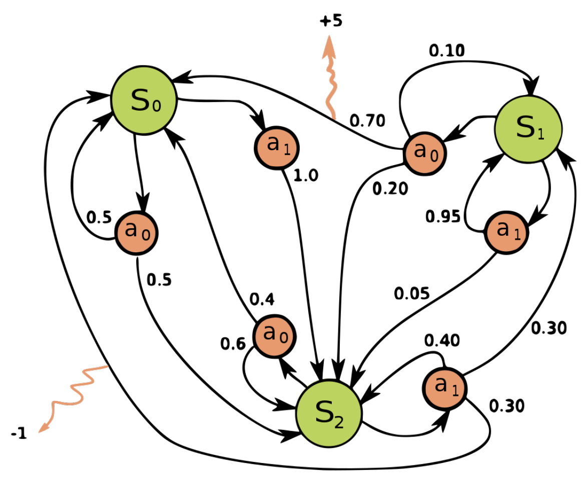


Figure 3.1: An example MDP with three states, two actions and two rewards.

process [85], in which a process with decision-making situations is modeled in discrete time steps and outcomes are partly random and partly under the control of the decision-maker. Formally, an MDP is defined as a tuple consisting of four variables:

- S: The state space, i.e., the set of states in the process.
- A: The action space, i.e., the set of actions in the process.
- $P_a(s_t, s_{t+1})$: The probability of transitioning to state s_{t+1} from s_t by taking action a.
- $R_a(s_t, s_{t+1})$: The reward by transitioning to s_{t+1} from s_t by taking action a.

As a result, we can say that at each time step the process is in state s_t and a set of actions, $A_s \subseteq A$, are available to the decision-maker. By taking some action $a \in A_s$, the decision-maker transitions to the state s_{t+1} with probability $P_a(s_t, s_{t+1})$ and receives reward $R_a(s_t, s_{t+1})$.

Figure 3.1 illustrates a MDP. In this illustration, there are three states: S_0, S_1 and S_2. From these states, the decision-maker can take either action a_0 or a_1. The squiggly arrows that point outside the process are rewards that the decision-maker receives for taking the corresponding action. The probabilities that are shown in the figure represent the likelihood of continuing the specific path. For example, from S_0, there is a 50% chance that the decision-maker will take action a_0. From there, the decision-maker either stays at S_0 with probability 0.5 or continues to S_2 with probability 0.5. Thus, the *state transition probability* from S_0 to S_0 can be computed as $0.5 \cdot 0.5 = 0.25$. The full mapping between states can be formulated in a *state transition probability matrix*:

$$P = \begin{bmatrix} 0.25 & 0 & 0.75 \\ 0.35 & 0.525 & 0.125 \\ 0.35 & 0.15 & 0.5 \end{bmatrix} \tag{3.1}$$

Each s_{ij} in the state transition probability matrix defines the probability of the decision-maker transitioning from state i to state j.

Markov decision processes assume that the underlying state is known to the decision-maker. However, in most real-world applications, this is not the case. A *partially observable* MDP (POMDP) [63] is different from standard MDPs as it does not assume that the underlying state is known to the decision-maker. Instead, it relies on past observations to compute the transitional probabilities. POMDPs are therefore often used to model situations where the complete underlying state is unknown, e.g., simulations.

MDPs/POMDPs for simulation are especially popular in the field of Reinforcement Learning. Most reinforcement learning algorithms take advantage of the episodic simulator, in which a decision-maker is simulated from an initial state. Each time the decision-maker receives an input action, it yields the resulting state and reward, thus useful for training the decision-maker to achieve an optimal policy.

The simulations in this thesis can be seen as POMDPs without rewards: the simulated user is the decision-maker, and the past observations are given in the form of sample dialogs between users and the agent.

3.3 Agenda-Based Simulation for Conversational Recommendation

Our approach to building a user simulator is based on the work presented in [1]. In this section, we briefly explain the core of this simulator.

Categorically, it is an Agenda-based User Simulator [82], in which the users follow an agenda consisting of dialogue acts in order to achieve their goal, see Sect. 2.4.3.1. Once again, consider the example where a user is looking for central bars serving beer. Figure 3.2 illustrates the example in an agenda-based system. There are a couple of observations to make. First, notice that the constraints and requests are defined as a list of slot-value pairs, similar to how slot-filling techniques function. These are defined in variables C_0 and R_0 respectively. Secondly, the user agenda A_1 is a stack-like structure, initialized to contain the actions necessary in order to accomplish the user goal. The

$$C_0 = \begin{bmatrix} type = bar \\ drinks = beer \\ area = central \end{bmatrix}$$

$$R_0 = \begin{bmatrix} name = \\ addr = \\ phone = \end{bmatrix}$$

Sys 0 Hello, how may I help you?

$$A_1 = \begin{bmatrix} inform(type = bar) \\ inform(drinks = beer) \\ inform(area = central) \\ request(name) \\ request(addr) \\ request(phone) \\ bye() \end{bmatrix}$$

Figure 3.2: An example of how a user agenda could look like. Taken from [82]. The subscripts indicate turn number.

constraints are expressed through multiple *inform* actions, which is similar to the *reveal* action defined in Sect. 2.3.4. Similarly, the desired information is represented as a set of *request* actions that correspond to the *more* action described earlier.

As we mentioned in Sect. 2.4.3.1, the agenda is continuously updated throughout the dialogue. Thus, by considering a dialogue as a sequence of state transitions enabled by dialogue acts, and each state depends only on the previous state, dialogues can be modeled as an MDP, see Sect. 3.2. In other words, the user transits from an initial state s_t to the next state s_{t+1} by performing action a_t from the agenda A_t [1]. Furthermore, the following state transition probabilities are defined by Zhang and Balog [1]:

- $P(s_{t+1}|A_t, s_t) = P(A_{t+1}|A_t, g_{t+1}) \cdot P(g_{t+1}|A_t, g_t)$: The probability of going from one state to the next.
- $P(g_{t+1}|A_t, g_t)$: An indicator function returning 1 if g_t was accomplished, otherwise 0.
- $P(A_{t+1}|A_t, g_{t+1})$: Agenda update signifying a **pull** operation, in cases where the goal g_t was accomplished.
- $P(\tilde{a}_t|A_t, g_{t+1})$: Agenda update signifying a **push** operation. This probability is used when the goal was not accomplished, and a replacement action $\tilde{a}_t$ needs to be pushed onto the agenda, with the same original goal.

Thus, the state transition probability $P(s_{t+1}|A_t, s_t)$ reduces down to $P(A_{t+1}|A_t, g_{t+1})$ when the goal was accomplished, otherwise $P(\tilde{a}_t|A_t, g_{t+1})$. The agenda update probabilities are determined by the interaction model.

3.4 User modeling

We model users with respect to four distinct models: interaction, preference, context and persona. Zhang and Balog [1] model users via an interaction model and a preference model. The interaction model is somewhat similar to what we discussed in Sect. 2.3.4 and defines how the user agenda can update throughout the conversation. More interesting however, is their preference model, as it attempts to capture personal tastes represented as a set of attribute-value pairs. Out of the two preference models they develop, the advanced one allows for consistent preferences across several simulations.

Salle et al. [86] model other contextual variables, i.e., cooperativeness and patience for conversational search refinement. The cooperativeness is measured as a real value between 0 to 1. For example, users with higher cooperativeness are more likely to provide additional information in their responses, in contrast to users with lower cooperativeness. The patience parameter indicates how many turns a user is willing to converse with an agent before it runs out patience and ends the conversation. These values may also change dynamically, i.e., throughout the session.

In the following sub-sections, we explain our approach to modeling these four dimensions (interaction, preference, context and persona). An overview of our approach is illustrated in Fig. 3.3.

3.4.1 Interaction Modeling

An interaction model can be seen as a communication protocol between the users and the agent. It defines which actions the user can take based on the most recent agent action. Zhang and Balog [1] experiment with two interaction models: QRFA [52] and CIR6, a model they developed specifically for Conversational Item Recommendation (CIR). They define a list of possible user actions $\mathcal{A}$ based on the framework presented in [51], which we discussed in Sect. 2.4.3.1. Since they primarily consider the user side, they treat the agent as a black box, with the assumption that the simulator can understand a set of agent actions, which are also based on [51]. Specifically, $\mathcal{A}$ includes the following actions (the following is a high level overview; there are other sub-categories within each action category see Sect. 2.4.3.1.): **Inquire**, **Reveal**, **Disclose**, **Navigate**, **Note** and

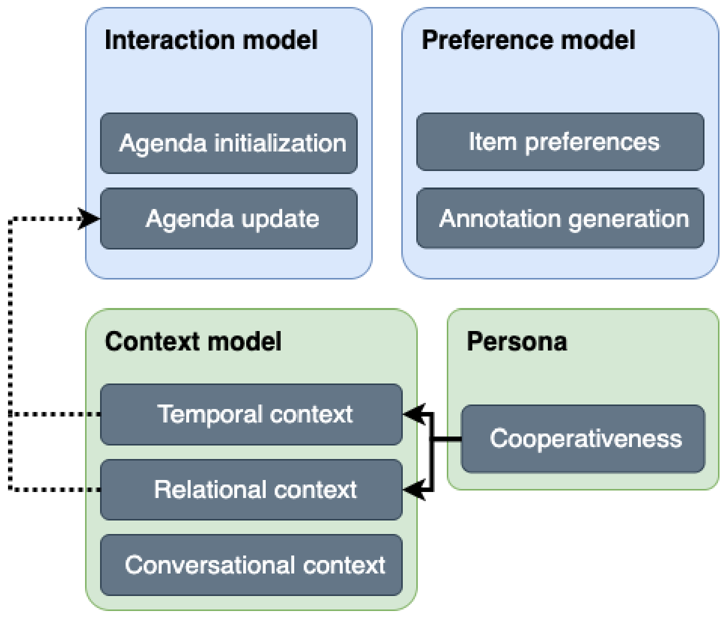


Figure 3.3: User modeling concepts. The green boxes indicate our main contributions. There is an indirect link between the context model and interaction model. The context model influences the user's patience which in turn impacts agenda update.

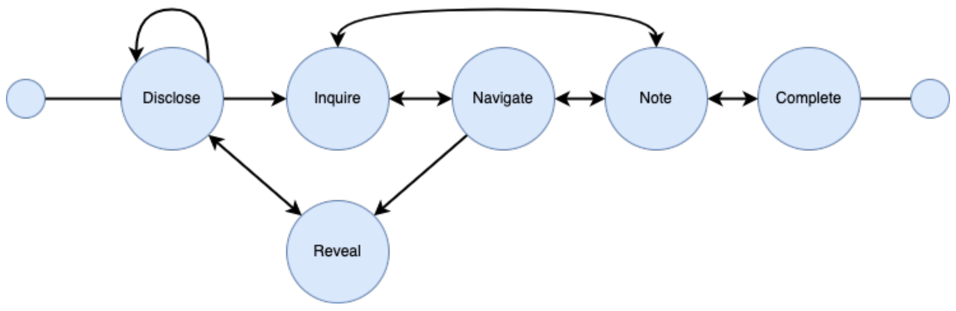


Figure 3.4: User state diagram. Arrows indicate action connection.

Complete. The agent actions that the user is assumed to understand includes **Reveal**, **Inquire**, **Navigate**, **Interrupt** and **Interrogate**.

The state diagram of the CIR6 model is shown in Fig. 3.4 with connections between the different actions. In this model, the next user action is solely based on the current user action. Using a probability distribution based on the training corpus, the most probable action is selected. Thus, the agenda update in Sect. 3.3 can be written as:

$$P(A_{t+1}|A_t, g_{t+1}) = \begin{cases} P(A_{t+1}|A_t) \cdot \mathbb{1}(a_{t+1}, A_t), & \delta(g_{t+1}, g_t) = 1 \\ P(\tilde{a}_t|b_t) & \delta(g_{t+1}, g_t) = 0 \end{cases} \tag{3.2}$$

where $\delta(g_{t+1}, g_t)$ is an indicator function returning 1 if the goal g_t was met, otherwise 0. $\tilde{a}_t$ is the replacement action sampled based on previous agent action, b_t.

The interaction model in our simulator is an extension of the CIR6 model. The difference is that we allow for intents that can depend on user preferences, e.g., the *note* intent. This allows for more reproducible simulations, as user preferences are consistent across simulations. Furthermore, we let agenda updates be dependent on the new contexts. Thus, the agenda update becomes slightly different as we will see in Sect. 3.4.3.

3.4.2 Preference Modeling

Preference modeling refers to modeling users' individual tastes and allows for a personalized experience. Zhang and Balog [1] experiment with two preference models. The simplest of the two is the *Single Item Preference* (SIP) model, in which user preferences for seen movies are generated randomly (i.e., by flipping a coin). In the more advanced model, *Personal Knowledge Graph* (PKG), preferences are based on the MovieLens dataset: a user's preference for a movie (or item) is based on the rating they gave it. The user's preference for an attribute is computed by averaging the user's ratings for movies that have that attribute. The set of sampled movies is divided into liked and disliked sets, as is the attribute. The PKG includes nodes for both items and attributes based on eight rated[1] movies. Thus it is used to elicit the user's preferences when asked. Preferences that are not explicitly specified by the user are inferred based on the user's historical preferences. For example, if an agent asks about the user's preference for a movie that the user has not seen yet, the preference model will infer it by aggregating the user's ratings for the genres in that movie.

In our approach, we use the PKG preference model, since preference modeling is not our focus. Our attention is dedicated to incorporating other contexts that we believe contribute to more realistic simulation. We did however implement some additional logic to the preference model, specifically the logic for choosing the next set of annotations for the user.

3.4.3 Context Modeling

Context modeling, as we have mentioned previously, is any information useful to characterize the situation of a user. Interaction modeling and preference modeling are just two of many ways to incorporate context. We extend the contexts used in [1] by considering the temporal and relational contexts. For the temporal context, we distinguish between *weekdays* and *weekends*, and the *time of the day*. The relational context is based on the group setting.

[1]One of these eight movies must be liked, i.e., rated at least 4/5. Movies that are rated 2 or less are considered disliked, while movies with ratings between 2 and 4 are considered neutral.

We believe that the *temporal* context influences the cooperativeness of a user, and by extension the patience. As we do not have access to real data that quantify the exact effect the temporal and relational contexts have on the two variables, we proceed by intuition. Specifically, we consider that users tend to be more cooperative during the evening as opposed to earlier in the day, as users are generally busy (e.g., with work) during the day and more relaxed in the evening. Similarly, users are less likely to be cooperative during hectic hours but otherwise cooperative. Furthermore, we differentiate between weekdays and weekends, to model the fact that users have fewer obligations during weekends. Intuitively, fewer obligations imply a larger cooperativeness score

The relational context incorporates the group setting. In our project, we take a simple approach and model it as a boolean variable, indicating whether the user is watching alone ("false") or with a group ("true"). We assume that groups tend to be more cooperative than single-users.

Similar to [86], we sample cooperativeness as a number between 0.0 and 1.0. Then, we scale this according to the user's temporal and relational contexts, according to Eq. (3.3) in order to compute the max number of times the user is willing to re-try when the agent breaks down. In this equation, c is the summed quantified contexts and h is the user cooperativeness. In our work, cooperativeness is closely connected to patience. Since the relational and temporal contexts are dynamic variables that we model as cooperative bonuses, we generate personas that are objects that contain static variables. For example, the initially sampled cooperativeness score is a static variable that we store in the user *Persona*. The patience of the user is changes depending on the dialog flow, i.e., it indicates the number of consecutive retries the user has made at a particular action/intent. We say that the user has run out of patience if the patience reaches $max\_retries$, thus it is dependent on cooperativeness as well. The distinction between static and dynamic user variables allows us to observe the effect of the new contexts on the conversation, for the same persona.

$$max\_retries = c \cdot h \tag{3.3}$$

Further, we model the user's satisfaction both overall and on the turn-level. We use satisfaction as a feature to influence the user response, as seen in Fig. 3.5. The figure depicts a situation where the conversational agent (green utterances) does not provide relevant recommendations to the user (blue utterances.) Thus the user's satisfaction with the system decreases, resulting in responses that reflect a dissatisfied user. The turn-level user satisfaction is classified using the models developed in [87]. Sun et al. [87] introduces the following satisfaction scores: 1 - very dissatisfied, 2 - dissatisfied, 3 -

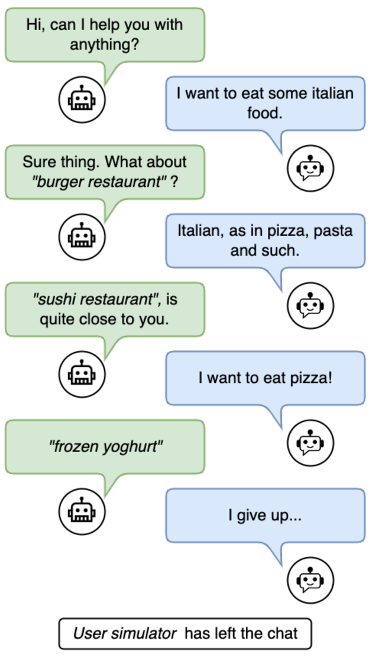

Hi, can I help you with anything?
I want to eat some italian food.
Sure thing. What about "burger restaurant" ?
Italian, as in pizza, pasta and such.
"sushi restaurant", is quite close to you.
I want to eat pizza!
"frozen yoghurt"
I give up...
User simulator has left the chat

Figure 3.5: An example dialog between an agent and a user. Although the user is reasonably cooperative, their patience runs out and the result is a failed dialog. The user responses also show that the user becomes less and less satisfied with the system.

Table 3.1: An overview of the variables used in the state transition probabilities with corresponding explanation.

Variable	Explanation
C_t	Overall user satisfaction at time t
c_t	Turn-level user satisfaction at time t
e_t	Patience level corresponding to action a_t
h	User cooperativeness

neutral, 4 - satisfied, and 5 - very satisfied. Throughout the conversation, the current satisfaction level will be classified using the previous turns utterances.

The state transition probabilities introduced in Sect. 3.3 are then modified to incorporate the new contexts:

- $P(s_{t+1}|A_t, s_t, e_{t+1}) = P(A_{t+1}|A_t, g_{t+1}, e_{t+1}) \cdot P(g_{t+1}|A_t, g_t)$: The probability of going from one state to the next. Compared to [1], we incorporate turn-level patience e_t into the agenda update.
- Following [1], we let $P(g_{t+1}|A_t, g_t) = \delta(g_{t+1}, g_t)$, i.e., an indicator function returning 1 if the goal g_t was accomplished and 0 otherwise.
- We differentiate how agenda updates are done based on goal accomplishment. For turns where the goal was accomplished, the agenda update corresponds to a pull action from the agenda, as in [1]. If the goal was not met, we consider the context.

Thus, a pull action results in the following agenda update:

$$P(A_{t+1}|A_t, g_{t+1}, e_{t+1}) = P(A_{t+1}|A_t) \cdot \mathbb{1}(a_{t+1}, a_t) \tag{3.4}$$

where $\mathbb{1}(a_{t+1}, a_t)$ indicates if the two consecutive actions are connected in the state diagram (Fig. 5) or not.

- If the goal was not met, the agenda is updated according to:

$$P(A_{t+1}|A_t, g_{t+1}, e_{t+1}) = \begin{cases} P(a_t^{i+1}|g_t, a_t^i) & \delta(e_{t+1}, e_t) = 1 \\ P(\tilde{a}_t|b_t) & \delta(e_{t+1}, e_t) = 0 \end{cases} \tag{3.5}$$

where $\delta(e_{t+1}, e_t)$ is an indicator function returning 1 if patience > 0, otherwise 0. If the patience of the simulated user has **not** run out, the user will repeat their previous action. On the other hand, once the patience reaches 0 (i.e., runs out), a new action based on the last agent action b_t is pushed onto the agenda.

- The overall user satisfaction is updated according to:

$$C_{t+1} = \phi(c_{t+1}, c_t, ..., c_1) \tag{3.6}$$

In other words, the overall satisfaction is an aggregation of the past turn-level satisfaction scores, for example, the average of the last three turn-level scores:

$$C_{t+1} = \frac{1}{3}(c_{t+1}, c_t, c_{t-1}) \tag{3.7}$$

- Similarly, we update the patience as a function of the previous patience and cooperativeness:

$$e_{t+1} = \sigma(e_t, h) \tag{3.8}$$

In this thesis, increase the patience by 1 if the goal was not met, and otherwise reset it to 0, thus indicating the number of retry attempts made by the user.

Figure 3.6 illustrates the agenda update process. Comparing this illustration to Fig. 3.7, the main difference is consideration of patience in order to decide between a new action or re-trying the previous action.

3.5 Instantiating the Simulator

To instantiate the simulator a few components have to be provided. Specifically, the simulator requires the interaction, preference and context models to be instantiated and a

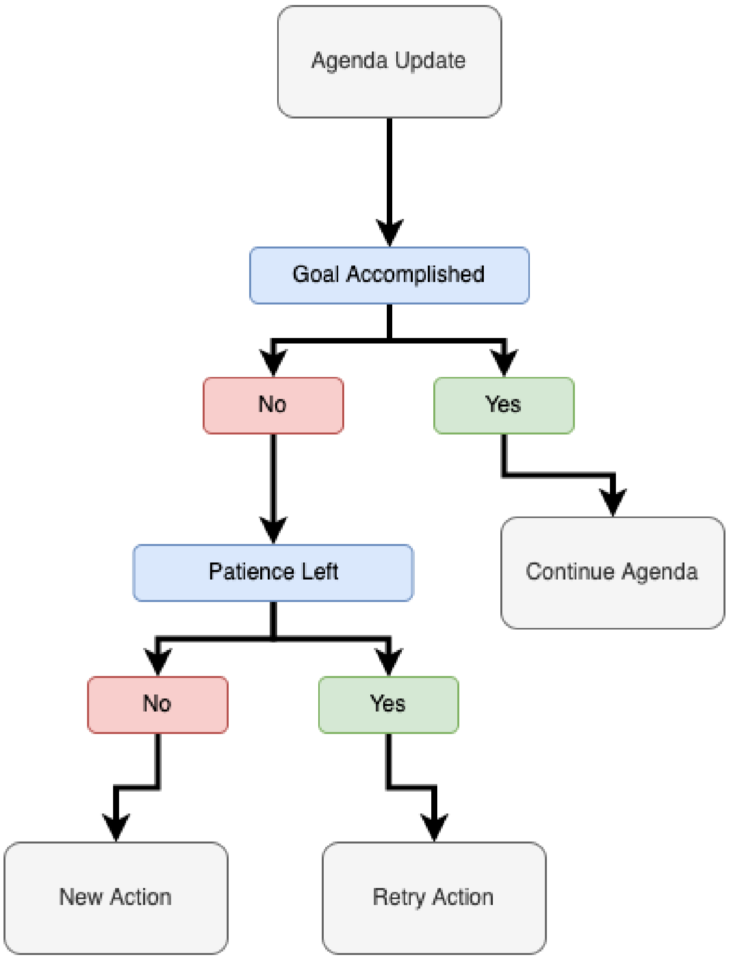


Figure 3.6: Illustration of the paths an agenda update can make.

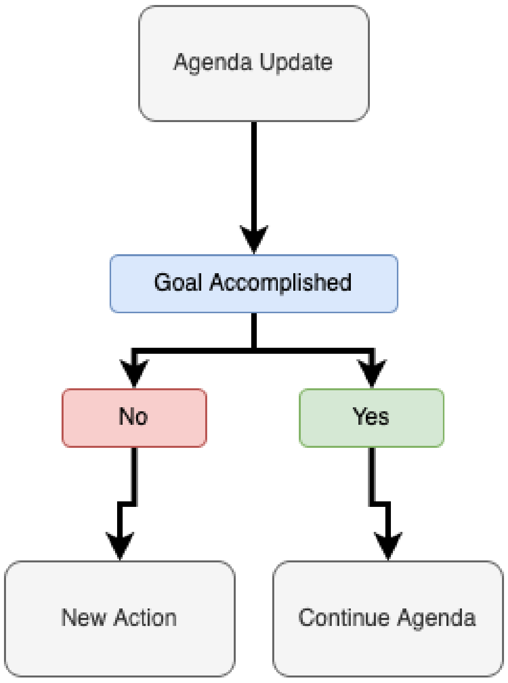


Figure 3.7: Illustration of the paths an agenda update can make with the simulator in [1].

persona. Note that the context model contains a sampled temporal and relational context that will be incorporated in the persona before the simulation starts. Alternatively, it is possible generate a persona with a specific context. Additionally, the NLU and NLG modules need to be trained and provided to the simulator. These components are initialized as follows:

- Interaction Model: Instantiated with a file defining the action space and another file with annotated dialogs.
- Preference Model: Requires an item collection and historical user ratings corresponding to these items. Additionally takes in a user id that defines which user's preferences the model should retrieve. The necessary data to initialize the preference model is also specified in two files: one containing items, and the other containing ratings. The preference model samples n consumed items for the provided user id and uses the corresponding user ratings to build the user's preferences. Whenever it is consulted for preferences, it returns the $top - k$ ranked preferences.
- Context Model: The context model requires values for temporal and relational contexts.
    1. Temporal context: *Time of the day* is sampled according to specified probabilities, which depend on *weekend/weekday* status. Similarly, the cooperativeness bonus this context outputs, is dependent on *weekend/weekday* status. The probability of a weekend situation is also specified by us.
    2. Relational context: We sample the relational context based on a uniform probability distribution.
- Persona: The persona is cooperativeness score is sampled using a normal distribution.
- Satisfaction classifier: This classifier is required in order to update the conversational context on a turn basis.
- NLU and NLG: The NLU module is instantiated with an intent classifier and a list of entity recognition models. Similar to the interaction model, the NLG module requires a file with annotated dialogs and which participant's utterances it should extract from this file. Furthermore, the NLG can compute satisfaction scores for these utterances if it is provided a satisfaction model.

The necessary files can be specified in a config file, which is then processed in order to instantiate the necessary components, i.e., the interaction and the preference models.

Chapter 4

Implementation

In this chapter we describe the technical details of our approach. First, an overview of the complete architecture is given, followed by the implementation details of *DialogueKit* and *UserSimCRS*.

4.1 Overview

The architecture of the user simulator is shown in Fig 4.1. To explain this architecture we will start with the red *Annotated Utterance* from the Conversational Agent. This utterance is an utterance with annotations from a CIA. The *Dialogue Manager* (DM) is part of *DialogueKit* which we will describe in more detail in the following sub-sections. The DM sends the same utterance but without annotations, to the simulated user. The utterance will be sent down two branches. The left branch is for understanding the utterance and responding appropriately, while the right branch is for satisfaction classification. Going down the left branch, the utterance will have its intent classified and entities will be extracted, using the *DialogueKit* Natural language module. The intent will be sent to the *interaction model*, which returns the next user intent. The annotations from the utterance are sent to the *preference model*. In this model, the user's preferences for the utterance annotations are retrieved. Stepping back to the right branch. The DM stores the dialogue history, the last user utterance will be retrieved from this. Then the agent utterance and last user utterance will be sent to the *DialogueKit satisfaction classifier*. The results from the *interaction model*, *preference model*, and *satisfaction classifier* are used to generate natural language responses using the *DialogueKit* NLG. This user response will be sent to the DM with annotations and sent back to the agent.

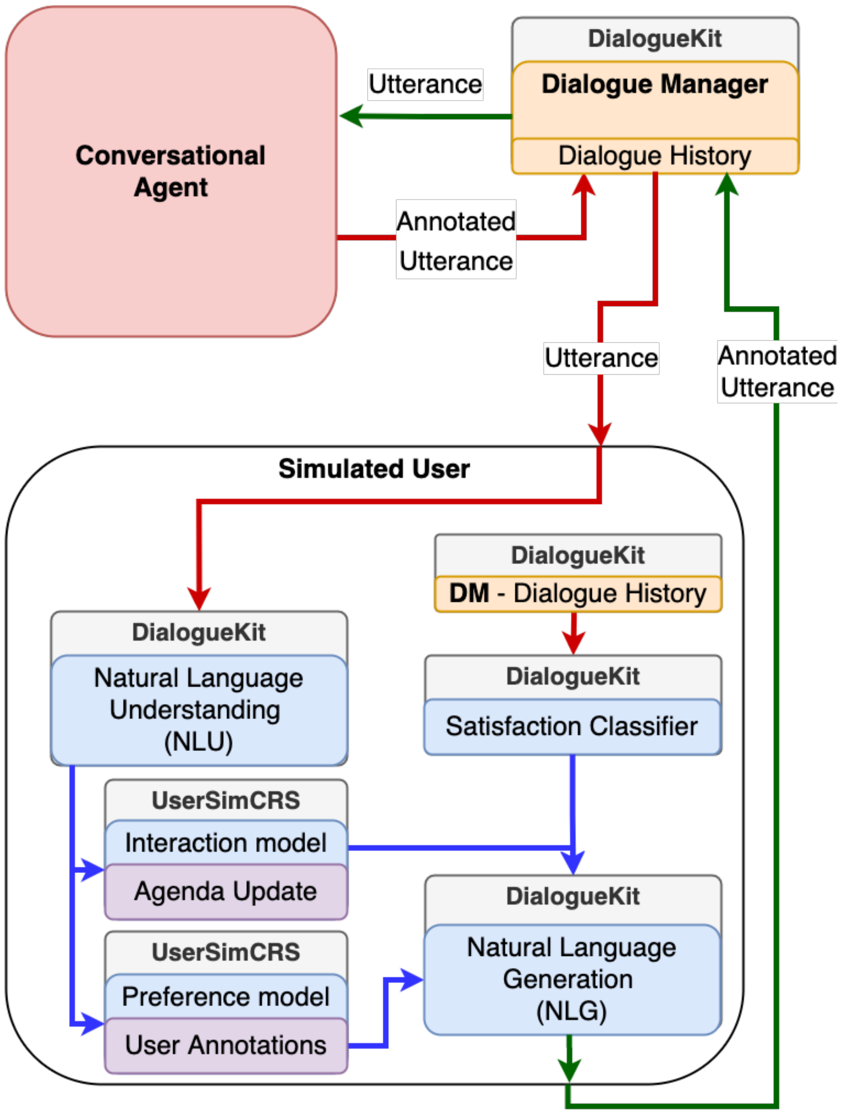


Figure 4.1: User simulator architecture.

4.2 DialogueKit

DialogueKit is a conversational framework that was initially part of another project. In this thesis, we have *extended* it with functionality that facilitates both the development of conversational assistants and the communication between two conversational assistants. In the following, we formally introduce *DialogueKit* and the functionalities we contribute.

4.2.1 Core Concepts

The *core* contains the central classes for the DM. These are: **AnnotatedUtterance**, **Annotation**, **Dialogue**, **Intent**, **Ontology**, **SlotValueAnnotation** and **Utterance**. These classes have different responsibilities:

- **Utterance**: Holds the user or agent utterance. This is a simple string, without any additional information.
- **Intent**: The intent of a participant utterance, i.e., the action which the participant wishes to take.
- **Ontology**: The domain ontology. Contains which slot keys are allowed and expected in the conversation.
- **Annotation**: Stores a key value pair of text annotation, e.g., key:*GENRE*, value:*comedy*.
- **AnnotatedUtterance**: Is a subclass to **Utterance**, that additionally stores *annotations*, **Intent** and **satisfaction** score.
- **SlotValueAnnotation**: Is a subclass of **Annotation**, that additionally stores where in the **Utterance** the annotation is.
- **Dialogue**: Stores the **User** and **Agent** *utterances*.

4.2.2 Architecture

There are seven central parts that make up the *DialogueKit* package these are illustrated in Figure. 4.2. The indicated interactions show how a system may look when it is built using *DialogueKit*.

4.2.3 Dialogue Manager

The dialogue manager is the most important part of *DialogueKit*. The DM's responsibility is to facilitate the conversation between the participants, store the conversation in a **Dialogue** and display the conversation on a **Platform**.

After a conversation ends, the DM can also export the conversation history to a JSON file. This document can then be used for other parts of *DialogueKit*, e.g., as training data for Natural Language Generation (NLG) and Natural Language Understanding (NLU).

4.2.4 Platform

The **Platform**'s responsibility is to display the conversation. DialogueKit includes a simple terminal-based platform. However, it can support other platforms by facilitating communication over POST requests. To avoid unnecessary complexity, the **Platform** is

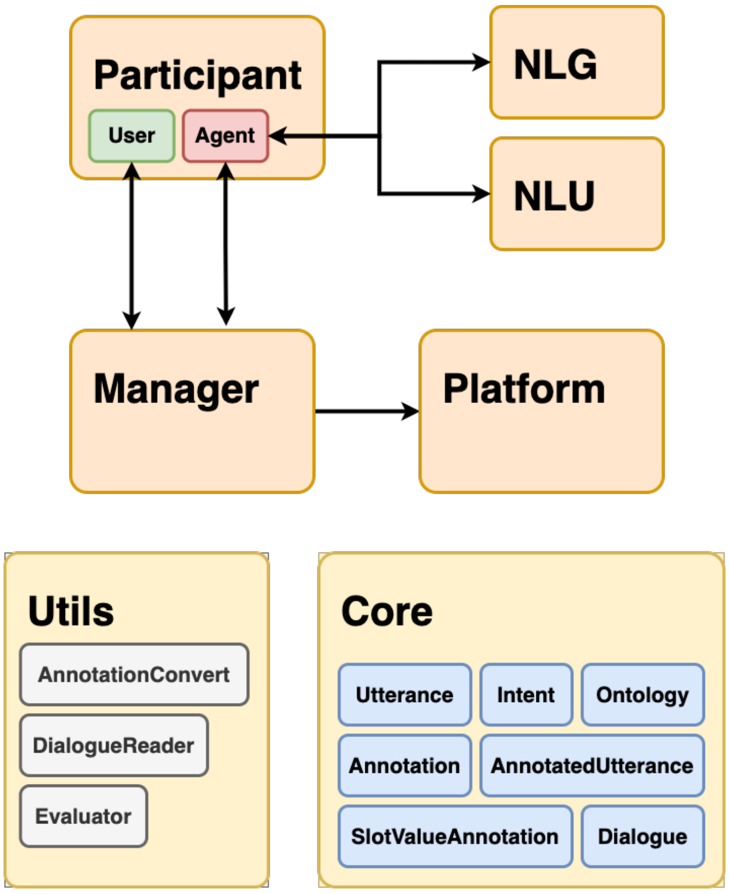


Figure 4.2: Illustration of the concepts *DialogueKit* incorporates, and a general idea of the interaction between the modules.

limited to only display utterances from the participants, i.e., user and agent utterances. This approach allows the DM and the **Platform** to be as independent of each other as possible and simplifies the integration of other platforms.

4.2.5 Intent Classification and Entity Extraction

The NLU module's responsibility is twofold: **intent classification** (identifying which action the participant is taking) and **entity extraction** (extracting *annotations* in the participant utterance).

DialogueKit includes two intent classifiers: cosine intent classifier and Rasa DIET classifier. The cosine intent classifier computes the cosine similarity between the input utterance and the training set to identify the utterance intent. Note that this classifier was implemented by the previous contributors of *DialogueKit*.

The Rasa DIET classifier, as the name suggests, uses the Dual Intent and Entity Transformer (DIET) classifier developed by Rasa. It can perform both intent classification

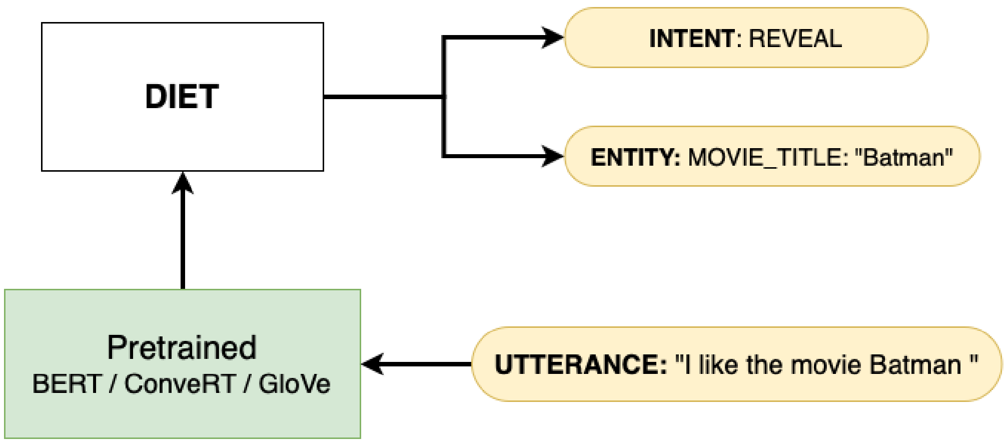

DIET
INTENT: REVEAL
ENTITY: MOVIE_TITLE: "Batman"
Pretrained
BERT / ConveRT / GloVe
UTTERANCE: "I like the movie Batman "

Figure 4.3: High level DIET classifier overview.

and entity extraction and is our contribution to the NLU module of *DialogueKit*. Figure 4.3 includes an overview of the Rasa DIET classifier.

To implement the Rasa DIET classifier, we use Rasa as a package and re-create the minimal implementation of DIET.[1] Furthermore, since the core classes in *DialogueKit* are not the same as the classes used in Rasa, we translate from *DialogueKit* to Rasa during inference and back to *DialogueKit* afterward. In essence, we have created a wrapper around the Rasa package. The implementation is quite minimal, using only the `WhitespaceTokenizer` and `CountVectorsFeaturizer` for the pipeline. This is much smaller than the default pipeline Rasa uses. The default Rasa pipeline can be seen in listing 4.1. It is possible to add more steps in the pipeline and add configuration parameters to tune the model, however, we omit this as it is task-dependent.

```

pipeline:
  - name: WhitespaceTokenizer
  - name: RegexFeaturizer
  - name: LexicalSyntacticFeaturizer
  - name: CountVectorsFeaturizer
  - name: CountVectorsFeaturizer
    analyzer: char_wb
    min_ngram: 1
    max_ngram: 4
  - name: DIETClassifier
    epochs: 100
    constrain_similarities: true
  - name: EntitySynonymMapper
  - name: ResponseSelector
    epochs: 100
    constrain_similarities: true
```

[1]This is possible as Rasa has an Apache-2.0 license: https://github.com/RasaHQ/rasa/blob/main/LICENSE.txt

```
  - name: FallbackClassifier
    threshold: 0.3
    ambiguity_threshold: 0.1
```

Listing 4.1: Rasa default pipeline.

```
[
    {
        "conversation ID": "\"39GHHAVOMGCMCWD43ZNZ2ARD9RF4JH\"",
        "conversation": [
            {
                "participant": "USER",
                "utterance": "hello\n",
                "intent": "DISCLOSE.NON-DISCLOSE"
            },
            {
                "participant": "AGENT",
                "utterance": "Which genres do you prefer?\n",
                "intent": "INQUIRE.ELICIT"
            },
            ...
    },
    ...
]
```

Listing 4.2: Sample from a dialogue export JSON.

```
version: '3.0'
nlu:
  intent: DISCLOSE.NON-DISCLOSE
- examples: |
    - Let's start over.
    - hello
    - /restart
    - /start
    - Hello.
  intent: DISCLOSE
- examples: |
    - Sports
    - Action
    - How about sci-fi?
    - Conspiracy thriller
    - action, fantasy
    - I want to restart for a new movie.
    - Cyberpunk
    - a new movie
    - Soccer
    - conspiracy
```

Listing 4.3: Sample from a nlu.yml document.

In order to train the DIET model, we need a `nlu.yml`[2] file containing intent to annotated utterances mappings, as shown in listing 4.3. The **AnnotationConverter** module in *DialogueKit* includes functionality that generates the `nlu.yml` file provided a JSON file with annotated dialogues, or **Utterance** and **Intent** pairs. A dialogue export / annotated dialogues excerpt can be seen in Listing 4.2.

If the input to the AnnotationConverter is a JSON dialogue export file, the converter will output two `nlu.yml` files, one for agent utterances, and another for user utterances. Additionally, the converter generates two other YAML files: a simple format conversion from JSON to YAML for the annotated dialogues and lastly a YAML document mapping annotation keys to corresponding values. The latter document can be useful for verification, i.e., whether the conversion has captured all the *annotations*.

As the Rasa DIET classifier can handle both NLU tasks, no further entity extraction models were created.

4.2.6 Natural Language Generation

```
{
    "DISCLOSE.NON-DISCLOSE": [
        "Hi, okay.",
        "Hi",
        "hello",
        "Sure, sounds fun.",
        "Let's start over.",
        "Okay, sounds good.",
        "/start",
        "Hello."
    ],
    "DISCLOSE": [
        "something like the {TITLE}",
        "{GENRE} {GENRE}",
        "{GENRE}, {GENRE}",
        "a new movie",
        "yes, {GENRE}",
        "{TITLE}",
        "I like {GENRE} movies",
        "I would like to restart",
        "{KEYWORD}",
```

[2]Rasa nlu.yml: https://rasa.com/docs/rasa/training-data-format/

```
        "something with {KEYWORD}",
        "How about {GENRE}?",
        "/restart",
        "{GENRE}",
        "I want to restart for a new movie.",
        "something {KEYWORD} and {KEYWORD}"
    ],
    ...
}
```

Listing 4.4: Sample from a NLG template. Notice the placeholder annotations marked with curly braces.

The Natural Language Generation (NLG) module is responsible for generating natural language responses. *DialogueKit's* NLG is based on [1], meaning it is template based. Based on the training corpus, utterance (either agent or user) templates are retrieved for each intent, including annotation placeholders. These placeholders are replaced with actual annotations provided by the participant. A sample from the NLG template can be seen in Listing. 4.4.

Selecting a response can be done in two ways: Randomly based on a desired **Intent**, or based on a desired **Intent** and **satisfaction score**. The latter approach selects the utterance with the closest satisfaction score to the desired one. The templates can be generated either with a previous dialogue export or by using the *core* classes. We follow the approach in [87] for satisfaction scoring. That is, we use the models introduced therein. The current implementation uses the SVM model trained on the English datasets in [87]. This model is also used to score the utterances in the manually annotated dialogues since this was previously missing. The satisfaction score of an utterance there is based on the previous two utterances in the dialog.

4.2.7 Satisfaction Classifier

As mentioned in Sect. 4.2.6, *DialogueKit* implements a satisfaction classifier. This classifier can be used to classify the satisfaction of a user based on the *utterances* of the user and agent.

4.2.8 Agents and Users

Agents and users are the participants in a dialog. *DialogueKit* assumes the agent to be a conversational system and the user to be a human. However, users of *DialogueKit* can

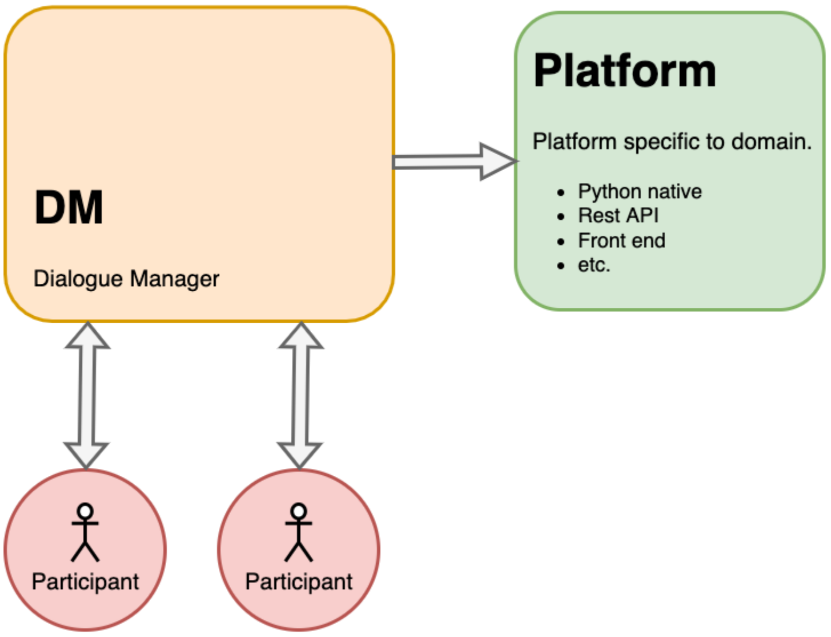


Figure 4.4: Illustration of the interaction between the DM, participants and Platform

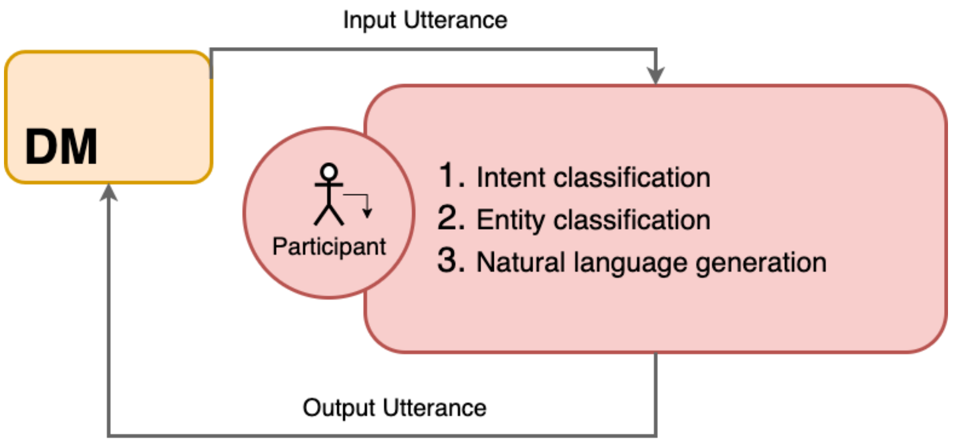


Figure 4.5: Illustration of how a Agent could be implemented

change this by implementing a subclass of User. For example, *UserSimCRS* implements a simulator as the user.

In a conversation, *DialogueKit* assumes the **Agent** to always start the conversation and also end it. A **User** can initialize the end of a conversation, but the responsibility lies on the **Agent** to stop it.

Out of the box *DialogueKit* contains some sample *agents*. These are described below:

- **ParrotAgent**: This agent will welcome the **User**, but will always parrot (echo) the **User**.

- **RasaParrotAgent**: This agent looks like the **ParrotAgent** to the user, but is actually just a connector to a Rasa conversational agent. This conversational agent is also part of *DialogueKit*.
- **MathAgent**: This **Agent** will ask the user simple arithmetic (addition, subtraction, multiplication and division) questions.
- **MovieBotAgent**: A Connector agent for IAI MovieBot[3]
- **TerminalAgent**: Allows a real human to interact with a **User**. This can be useful for testing user simulators.

These agents are subclasses of the **Agent** class. Similarly, users of *DialogueKit* are expected to extend the Agent class when creating their own implementation of an agent.

The **User** implementation in *DialogueKit* is kept simple so that it is easily extendable. It consists of only one function, *receive_utterance* which the DM will call to send agent utterances. It is up to the users of *DialogueKit* to decide what this function will do in their case. For example, in the case of *UserSimCRS*, the simulator will perform NLU tasks and decide how to respond based on the detected intent and entities. It is also possible for a human to act as the user since the **User** class' default behavior asks for input via the terminal. However, these inputs (utterances) will lack the annotations in the dialogue export. *DialogueKit* includes one sample user, **MathUser** that additionally asks for annotations.[4] Figure 4.4 illustrates the connection between the DM, **Platform** and the participants.

4.3 UserSimCRS

Our simulator framework, UserSimCRS, is an extended version of the simulator in [1]. In the previous sections, we have introduced the main properties of the simulator, i.e., it is an agenda-based user simulator with interaction-, preference- and context modeling.

UserSimCRS is built with adaptability in mind, that is, it can be used with any conversational agent. This is possible since it is data-driven, i.e., the simulator requires a file with annotated user-system dialogues, see Listing 4.2 for an example. Furthermore, unless a specific file identifying the user-system interactions is given, we assume that the agent can understand the set of actions defined by the CIR6 model, mentioned in Sect. 3.4.1. The framework also allows for other intent schemes to be used, however, these should follow the same format as shown in listing 4.5.

[3]IAI MovieBot: https://github.com/iai-group/moviebot

[4]MathUser was specifically created to be used with MathAgent.

```
user_intents:
    - Intent

user_preference_intents:
    Intent:
        - Sub-intent

user_add_preference_intents:
    - Intent

user_remove_preference_intents:
    - Intent

agent_intents:
    - Intent

agent_elicit_intents:
    - Intent

agent_set_retrieval:
    - Intent

expected_responses:
    Intent:
        - Intent
```

Listing 4.5: The modified CIR6 intent scheme format. Modifications include the addition of "user_preference_intents", "user_add_preference_intents" and "user_remove_preference_intents". These allow for consistent preferences in simulated dialogs.

4.3.1 Preference Model

Throughout a conversation, a user may specify various preferences and also refine or revise these preferences. In order to enable this functionality, we extend the preference model to store which preferences have been specified at any time during the conversation. This way, whenever the user wants to perform a revise or refine action, the preference model randomly samples one of these preferences.

Additionally, we implement the logic for choosing user slots in the preference model. We implement this as a function of the agent's annotations. The entity recognition module is responsible for detecting which slots the agent is looking for, e.g., a certain item property preference. The function then returns the $top - k$ relevant values for the detected slot.

4.3.2 Interaction Model

As we mentioned in the previous chapter, we modify the CIR6 intent scheme. Specifically, we make the following modifications:

- reduction: We remove a number of intents from the CIR6 scheme, as they are not used in our case. This includes:
    1. User *inquire* sub-intents: *inquire.list*, *inquire.compare* and *inquire.subset.*
    2. User *navigate* sub-intents: *navigate.back* and *navigate.repeat.*
    3. User *disclose* sub-intents: *disclose.review* and *disclose.non-disclose-review.*
    4. Agent *inquire* sub-intents: *inquire.clarify* and *inquire.elicit-review.*
    5. Agent *reveal* sub-intents: *reveal.list* and *reveal.subset*
    6. Agent *traverse* intents.
- **addition**: We add a few intents to the CIR6 scheme:
    1. Agent *disclose* sub-intent: *disclose.more.* This intent signifies that the agent is disclosing more information regarding a recommendation, e.g., the duration/plot/actors of a movie.
    2. Agent *inquire* sub-intents: *inquire.more* and *inquire.next.* The former denotes utterances where the agent asks the user "What would you like to know about movie?", while the latter is used to inquire about the user's next step in the dialog.
    3. Agent *reveal* sub-intent: *reveal.none.* This intent is detected when the agent conveys that there are no recommendations that satisfy the user's requirements.
- *note* intent: We split the previous *note* intent to four new sub-intents: *note.yes*, *note.like*, *note.dislike* and *note.accept.* This is because the *note* intent is *generally* used to express the user's opinion on a given recommendation. The exception is *note.accept*, which is used to accept a recommendation. Since the interaction model has no knowledge of the preference model, it still outputs a *note* intent, which we change during the response generation. The preference model is consulted to check for two things: consumption status and preference. That is, whether the user has seen (or consumed) the recommendation or not, and the user's normalized rating for the recommendation. Note that if the user has seen the suggested movie, the user will respond with *note.yes.* On the other hand, if the user has not previously seen the recommended movie, the output intent will be either *note.like* or *note.dislike* based on the preference (which is inferred by the preference model).

- We add two new categories:
    1. *user_preference_intents*: This category of intents was added to implement the logic required for the *note* intent. The annotated dialogues contain the specific sub-intents of the *note* intent, which we map back to the general *note* intent during agenda initialization, as we do not know which movie the agent will recommend, in addition to what the user's preference for that movie will be. Listing 4.6 displays this category.
    2. *user_remove_preference_intents*: This category includes intents that are used when the user removes a previously specified preference. In our case, we only have one intent that does this, which is *reveal.revise*.

It should be noted that the changes we made to the intent scheme are catered to IAI MovieBot, as it simplifies testing and usage of the simulator. In general, any intent scheme can be used, as long as it captures the user-agent interactions. Furthermore, these changes were propagated to the annotated dialogues file, since the dialogues need to be annotated according to the intent scheme being used. We annotate each turn in the dialogues according to what we believe is the most correct intent for that turn. Annotation disagreements were resolved through discussion.

```
user_preference_intents:
  NOTE:
    POSITIVE:
      - NOTE.LIKE
    NEGATIVE:
      - NOTE.DISLIKE
    CONSUMED:
      - NOTE.YES
```

Listing 4.6: user_preference_intents

4.4 IAI MovieBot

IAI MovieBot is a conversational movie recommender developed by the IAI group. It was originally created to recommend movies using multi-modal interfaces, such as Telegram [5] and Facebook Messenger [6]. This conversational agent is one of the agents used in previous research for evaluating user simulation approaches. However, in order to use the bot to evaluate our user simulator, some modifications were necessary. In particular, we required single-modal responses (text responses) and a means of communication that did

[5]Telegram: https://telegram.org/
[6]Facebook Messenger: https://www.messenger.com/

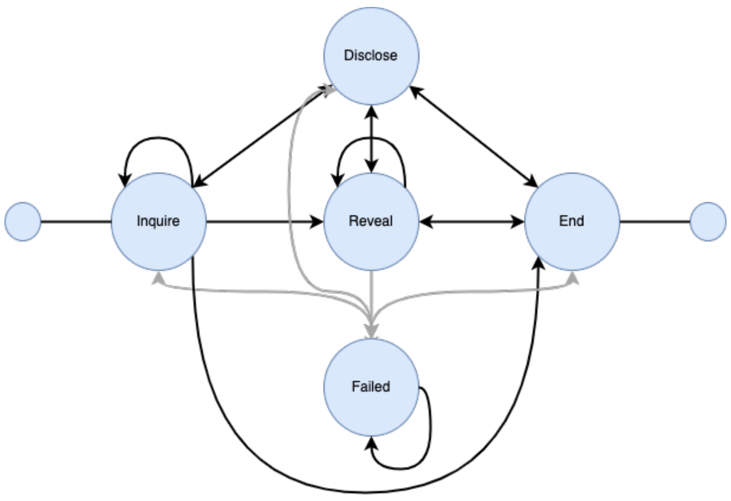


Figure 4.6: Agent state diagram. Arrows indicate action connection.

not include third-party platforms. To this end, we implemented a new controller that launches a Flask server[7] and all responses are text.

Figure 4.6 illustrates the state diagram of the IAI MovieBot, which we use to test UserSimCRS. Notice that any action can be followed by any other action, except *end* which always terminates the conversation. This is mostly due to the fact IAI MovieBot reacts to the user's action, in addition to mistakes made by the NLU.

[7]Flask: `https://flask.palletsprojects.com/en/2.1.x/`

Chapter 5

Experimental Evaluation

This chapter presents our experiments and the corresponding results. We start by presenting our experimental setup, with the used evaluation measures, and the item and preference data in Sect. 5.1. Further, we describe how the simulator gets instantiated with context, preference, and interaction model, alongside the different simulator configurations in Sect. 5.2. Finally, we present our results in Sect. 5.3, with the corresponding discussion in Sect. 5.4.

5.1 Experimental Setup

We follow the evaluation method in [1]. Zhang and Balog [1] evaluate their user simulator by having it converse with three conversational recommender systems in the movie domain. In our case, we evaluate our user simulator using IAI MovieBot as the conversational agent. Additionally, we conduct a Wizard of Oz experiment.

5.1.1 Experimental Measures

Zhang and Balog [1] compare the conversational agents against each other by considering the following metrics:

1. **AvgTurns**: As suggested by its name, this metric indicates the average number of turns in the conversation.
2. **Reward**: This metric reflects the performance of a simulator by assigning and deducting points throughout the dialogue. Specifically, the authors assign 4 points for each of the following action: *Disclose*, *Refinement*, *Inquire*, *Navigation* and

mixed-initiative. Thus, *Full* is set to 20 for agents that support all of the actions above, otherwise they deduct 4 points (from Full) for each action that is not supported. Furthermore, the authors consider two consecutive *Repeat* actions as one turn and deduct 1 point for each turn. Finally, the metric is defined as: $Reward = max0, Full - Cost * T$, where T indicates the number of user turns in the dialogue.

In our case, the actions *Refinement* and *Navigate* are not supported, thus *Full* is set to $20 - 8 = 12$. Additionally, since each participant may only send one utterance at a time, our results are not influenced by *Repeat* actions.

3. **Success Rate**: The authors consider a turn successful if the agent returns an appropriate action, otherwise failure. Thus, this metric is on the turn level, in contrast to Reward which was end-to-end.

Additionally, Zhang and Balog [1] consider the metrics *UserActRatio* and *DS-KL*. However, these are not relevant in our case, and will thus be omitted from our results. Furthermore, we expand on the above metrics by adding the conversational measure *satisfaction* from Sect. 3.4.3, as a measure of users' overall satisfaction with the conversation.

We believe that the contexts influence the metrics, as the contexts are used to compute the maximum number of times a user is willing to retry their previous action whenever the agent responds with an unexpected action. Therefore, if the agent keeps misunderstanding the user, the *AvgTurns* will increase, resulting in a lower *AvgReward*. Additionally, *AvgSuccess* will decrease as there are fewer turns where the turn goal, g_t, is met. On the other hand, if the agent understands the action being performed by the user, these metrics should receive the opposite effect, i.e., *AvgTurns* should decrease, *AvgReward* increase as there are fewer turns in the dialog and *AvgSuccess* increase as turn level goals are being met more often.

5.1.2 Item and Preference Data

As part of our setup, we will use the MovieLens 25M dataset[1]. This dataset provides a collection of 62,000 movies, 25 million user ratings, and 1 million tag applications applied to the movies by 162,000 users. Additionally, the tag genome data comes with 15 million relevance scores across 1,129 tags.

The dataset is separated across multiple files:

- **movies.csv** Stores movie data as a three-tuple: *movieId*, *title*, *genres*.

[1]MovieLens 25M Dataset`https://grouplens.org/datasets/movielens/25m/`

- **ratings.csv** Includes user movie ratings in the following format: *userID*, *movieId*, *rating*, *timestamp*. We omit the *timestamp* field.
- **tags.csv** User given tags to a movie are structured as: *userId*, *movieId*, *tag*, *timestamp*. We do not use this file.
- **genome-tags.csv** Every tag gets associated with an identifier. The structure of the file is: *tagId*, *tag*.
- **genome-scores.csv** Tags are assigned a relevance score to every movie. This file is divided into: *movieId*, *tagId*, *relevance*.

5.2 Simulator Instantiation

We use a combination of distributions, hard mapped probabilities, and cooperativeness bonuses to generate a persona's context, as mentioned in Sect. 3.5. Every persona should have the context features identified in Sect. 3.4.3. Every distribution is chosen by intuition, and may not reflect real users. To augment the influence of a context feature, cooperativeness bonuses were given. In the following subsections, we explain how we quantify the contexts (i.e., cooperativeness bonuses).

5.2.1 Temporal Context

In this section, we describe the sampling process for temporal context.

5.2.1.1 Weekend and weekday

The distribution of weekend vs. weekday feature can be seen in Fig. 5.1. We set the probability of there being a weekend to $\frac{2}{7}$ to reflect the real world. The weekend feature does not result in a direct cooperativeness bonus, however, it influences the probabilities for *time of the day* feature.

5.2.1.2 Time of the Day

To sample data for the *time of the day* feature, we first discretize days into 3-hour segments, e.g., $0 - 3$, $3 - 6$, $6 - 9$ and so on. The result is a discrete probability distribution, which depends on the weekend feature for the specific probabilities, as shown in Fig. 5.2. The probabilities in both distributions are handcrafted to reflect

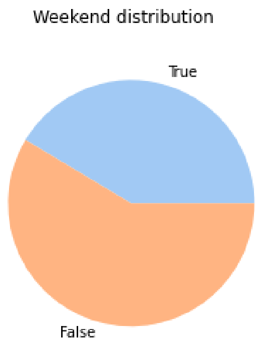


Figure 5.1: The distribution of the generated personas weekend feature. True denotes weekend, and False means it is a weekday.

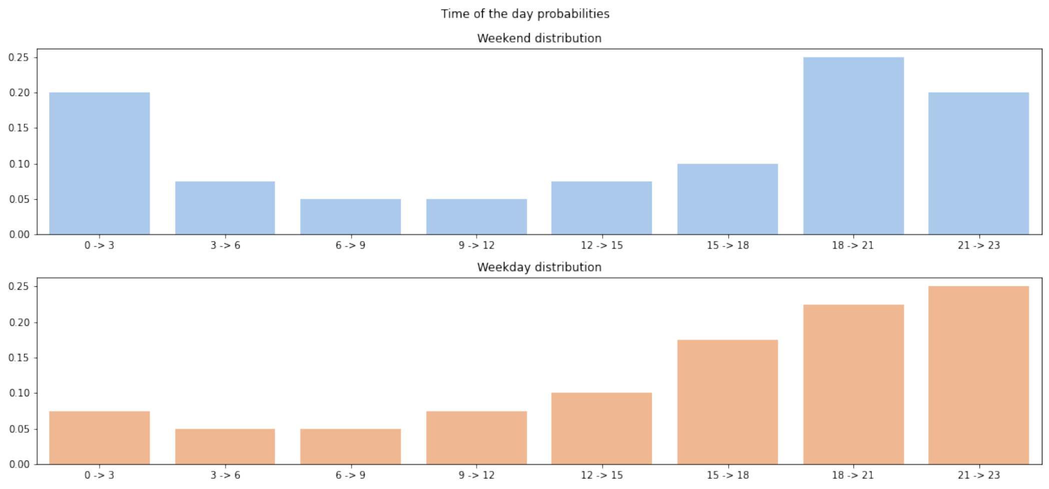


Figure 5.2: The probabilities of different time slots.

that users tend to be up longer during weekends. Each time slot (segment) results in a cooperativeness bonus $b \in [-2, 2]$. The distribution of bonuses can be seen in Fig. 5.4. The specific values are based on our intuition that users are less cooperative during busy hours (i.e., between 08-16), see Sect. 3.4.3.

5.2.2 Relational Context

Following our simple approach of modeling relational context as a boolean variable, we set the probability of relational context being *True* to $\frac{1}{4}$. In our thesis, this probability is not that important, as we want to investigate both situations. We assign a bonus of 1 for users with relational context, otherwise 0.

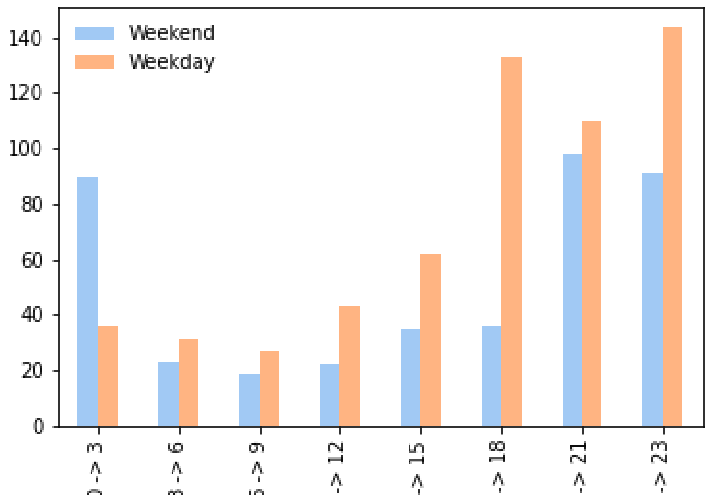


Figure 5.3: Distribution of time of the day context feature for both weekends and weekdays. Time slots are assigned along the x-axis, while the y-axis denotes the amount of users.

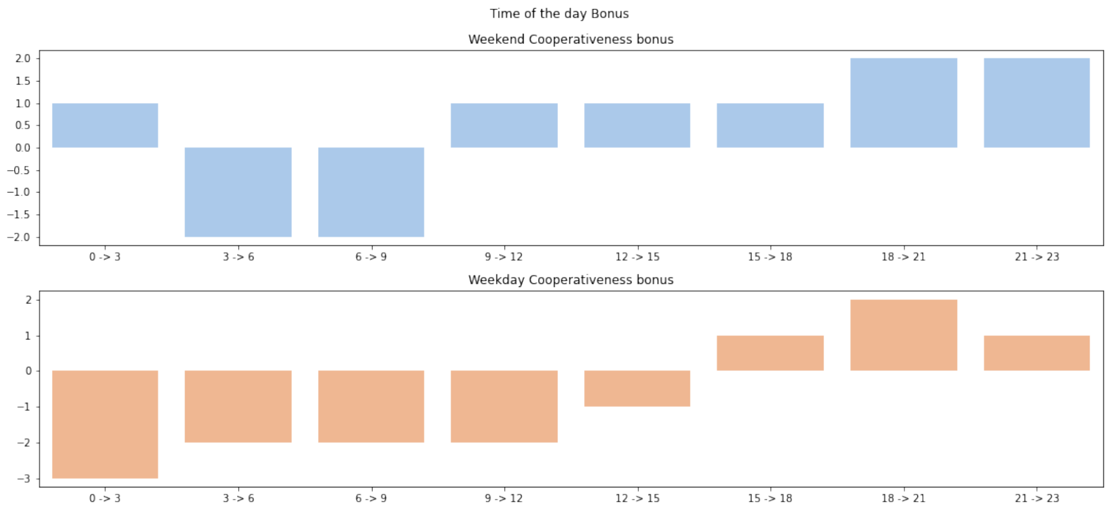


Figure 5.4: Time of day cooperativeness bonus. Notice that during weekdays, we assume highest cooperative bonus during 18-21 in the evening, while during weekends this bonus continues until 24, i.e., 18-24. This is due to users staying up later and having less obligations.

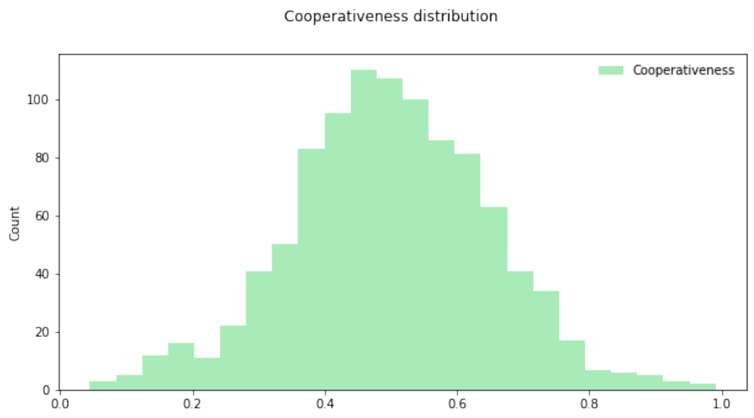


Figure 5.5: Cooperativeness distribution.

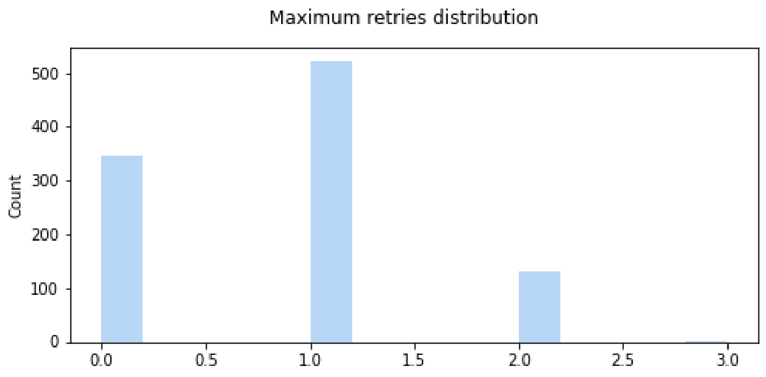


Figure 5.6: Distribution of the maximum retires property.

5.2.3 Persona

The cooperativeness score is sampled from a normal distribution, normalized between 0 and 1. As we have mentioned earlier, we multiply the user cooperativeness and the cooperativeness bonus to obtain the persona *max_retries* value. The distribution of cooperativeness can be seen in Fig. 5.5, while the distribution of *max_retries* can be seen in Fig 5.6.

5.2.4 Preference Model

We instantiate the preference model with the MovieLens dataset, which we detailed earlier in Sect. 5.1.2. More specifically, we will use the `ratings.csv` to build a user's preference. If a user rates a movie highly, then the preference for that type of movie will be increased and vice versa for low ratings. We set n, the number of items to sample, such that all items consumed by the user are used to build the user's preferences. Furthermore, we set k to be 3 in the function used to retrieve user slots. This further allows us to

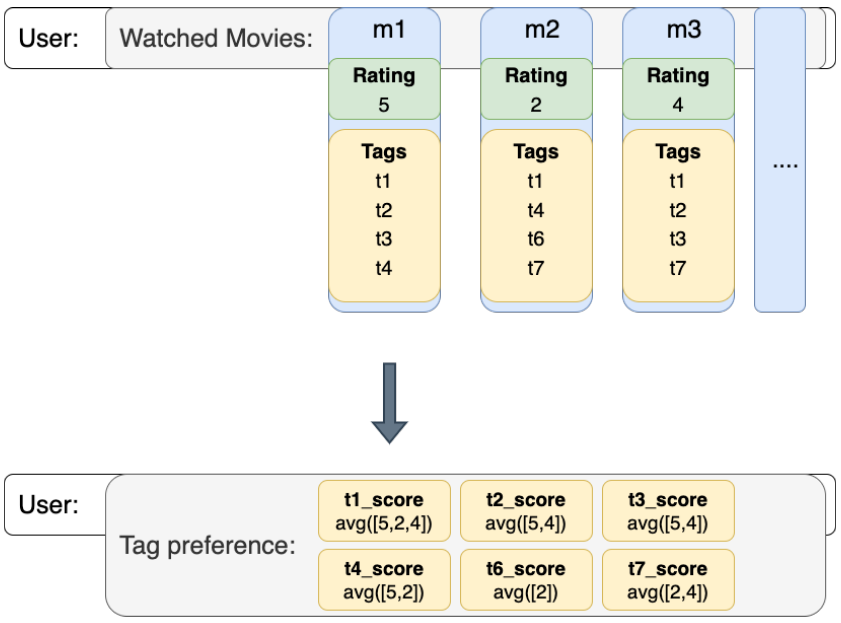


Figure 5.7: Illustration of the user tag ranking.

consistently model consumed items for simulated users. Since the IAI MovieBot can ask the user for keywords, we incorporate the genome data file. In particular, we take the top-5 relevant tags for each movie to represent keywords for the movie. To infer user tag ratings, we consider the movie ratings. That is, given a movie m with assigned user rating r_u, we add r_u to the score list of each tag representing m. At last, the score list of each tag is aggregated (by averaging) to obtain a single tag rating. The process is illustrated in figure 5.7.

To simplify the file usage, the top five highest-rated tags per movie are added to the *movies.csv* file as a new field named *keywords*.

Thus, the preference model is instantiated:

- **item collection**: This is an ItemCollection object from *DialogueKit* and is responsible for storing the domain items. The `movies.csv` file from MovieLens is used to instantiate this object.
- **historical ratings**: This is a Ratings object from *DialogueKit* and is responsible for storing users historical ratings. Note that these ratings are normalized between -1 and 1. The `ratings.csv` file from MovieLens is used in our setup to instantiate this object.
- **user id**: This id comes from the persona and is used to determine which user's ratings should be used.

Table 5.1: Context abbreviations

Temporal context	
W1/0	Weekend True/False
T1	Timeslot between time 0 and time 3
T2	Timeslot between time 15 and time 18
T3	Timeslot between time 21 and time 24
Relational context	
W1/0	Relational context True/False

Table 5.2: Personas and context combinations, group 1.

Persona	**P1**	**P2**	**P3**
Cooperativeness	0.2	0.6	1.0
Initial satisfaction	3	3	3
R0_W0	**Max retries**		
T1	0	0	0
T2	0	1	1
T3	0	1	1
R1_W1	**Max retries**		
T1	0	1	2
T2	0	1	2
T3	1	2	3
no_context	0	0	0

5.2.5 Simulating Conversations

Setting up the simulator was done as described in Sect. 4.3. We generate 3 personas, each with a different cooperativeness score, but the same initial satisfaction score. Furthermore, each persona is tested with 7 contexts, arranged into 3 arrangements: one with no relational context (i.e., group setting is set to *False*) and weekend status set to *False*, another with relational setting (i.e., group setting is set to *True*) and weekend status set to *True*, and finally the last arrangement includes no context. Table 5.2 displays the personas and the contexts, along with the resulting *max_retries* for each persona-context combination. The blue color gradient signifies patience, i.e., a darker color indicates higher patience.[2] A persona-context pair represents a user. The contexts are named according to their properties, see Table 5.1 for an explanation of the naming convention. For example, **R0_W0** stands for relational *False* and weekend *False* (i.e., the first arrangement mentioned above), and **T1** denotes the time slot from 00:00-03:00.

Additionally, we run the simulator in a Wizard-of-Oz (WoZ) setting, that is, we "pretend" to be the agent and converse with the simulator. The persona-context combination used in the WoZ setting can be seen in Table 5.3. Note that the personas have the same values for both cooperativeness and initial satisfaction level. The difference lies in the contexts,

[2]The difference between each gradient is 1 *max_retries* value, with blank color indicating $max\_retries = 0$

Table 5.3: Personas and context combinations, group 2.

Persona	**P_A**	**P_B**
Cooperativeness	1	1
Initial satisfaction	3	3
Max retries		
Time	T3	T3
Group setting	True	False
Weekend	True	False
Max retries	3	1

where we differentiate between **R0_W0** and **R1_W1**, but keep the same time slot. This results in different *max_retries* property for the two personas.[3]

The intent scheme used to instantiate the interaction model for all simulations can be seen in Appendix B.

5.3 Experimental Results

In this section we describe our results in terms of the metrics mentioned in Sect. 5.1. We start by verifying the integrity of the interaction model, as this is one of the most important aspects of the simulator. Then, we present the simulation results achieved using IAI MovieBot as the conversational agent. At last, we describe the WoZ setting results and point out the differences we observe when using IAI MovieBot vs. WoZ setting.

5.3.1 Interaction model verification

To verify that the interaction model's agenda initialization works as intended, i.e., the agenda is initialized based on observations, we first sample 3 simple "toy" agendas as training data and then simulate 10 000 agendas. Figure 5.8 shows the resulting intent distributions of both the training data and the simulated agendas. The x-axis indicates current intent and the bars the following intent. For example, in the upper row of Fig. 5.8, we observe that the *DISCLOSE* intent is followed by either *NOTE* (~ 0.4 probability) or *DISCLOSE* (~ 0.6 probability). Notice that the distribution of the simulated agendas follows the training data distribution very closely. This verifies that the interaction model initializes the agendas correctly. Figure 5.9 shows the intent distribution of the annotated dialogues (5 dialogues). Again, when we simulate 10 000 agendas based on the observed intent distribution, we find that the distributions are extremely similar.

[3]Note that we compute a persona's *max_retries* property according to Eq. 3.3, which is dependent on both context and persona cooperativeness.

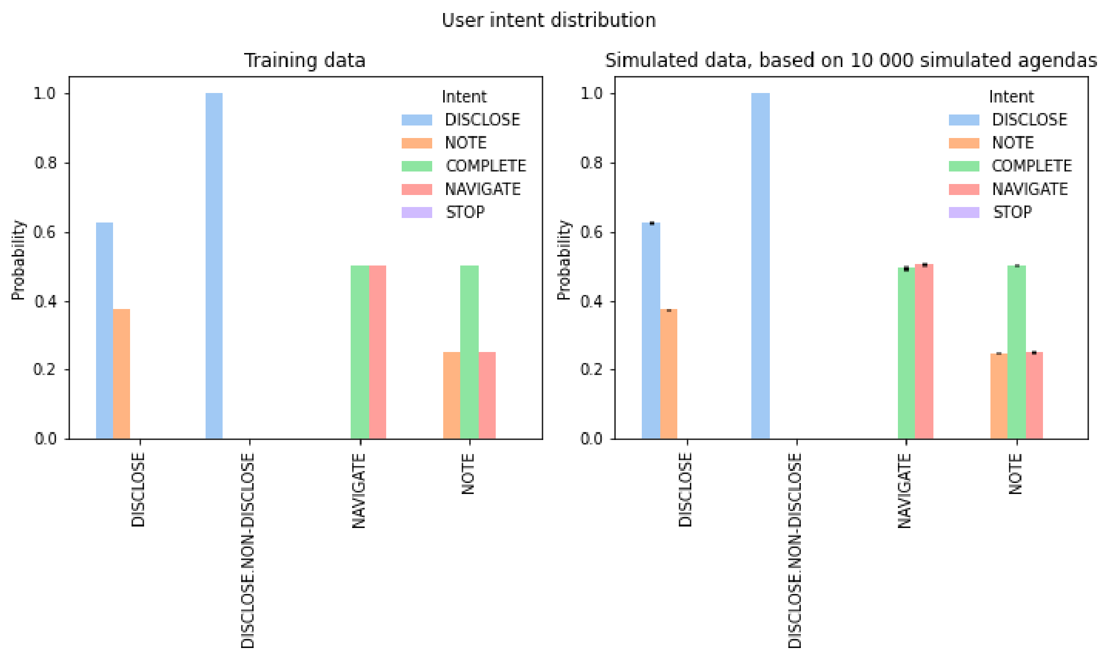


Figure 5.8: Intent distribution of toy data in the upper row and simulated data in the lower row. The intent distribution of simulated agendas closely resemble the training data distribution. Note the small error bars. These indicate standard deviation.

Similarly, we check if the lengths of initialized agendas are similar to the ones in the training data. Figure 5.10 shows the agenda lengths for our annotated dialogues (5 agendas) in the upper row and the agenda lengths for the simulated agendas in the lower row. We point out that most of the simulated agendas have similar lengths to the historical dialogues agendas, however, there are some that are longer (due to probability).

5.3.2 IAI MovieBot

We evaluate IAI MovieBot by simulating a total of 465 conversations, divided into two categories: dynamic agenda and static agenda, with 105 and 360 conversations respectively. The difference between the dynamic and the static agenda is that the agenda changes between each conversation in a dynamic agenda setting. For a more fair evaluation, we experiment with static agenda, i.e., all conversations have the same agenda. In the following, we present the measured metrics for the two categories and some of the issues related to IAI MovieBot.

5.3.2.1 Dynamic Agenda

Tables 5.4 and 5.5 show the resulting metrics for 105 conversations using the persona-context pairs from Table 5.2. We simulate 5 conversations per persona-context pair. Due

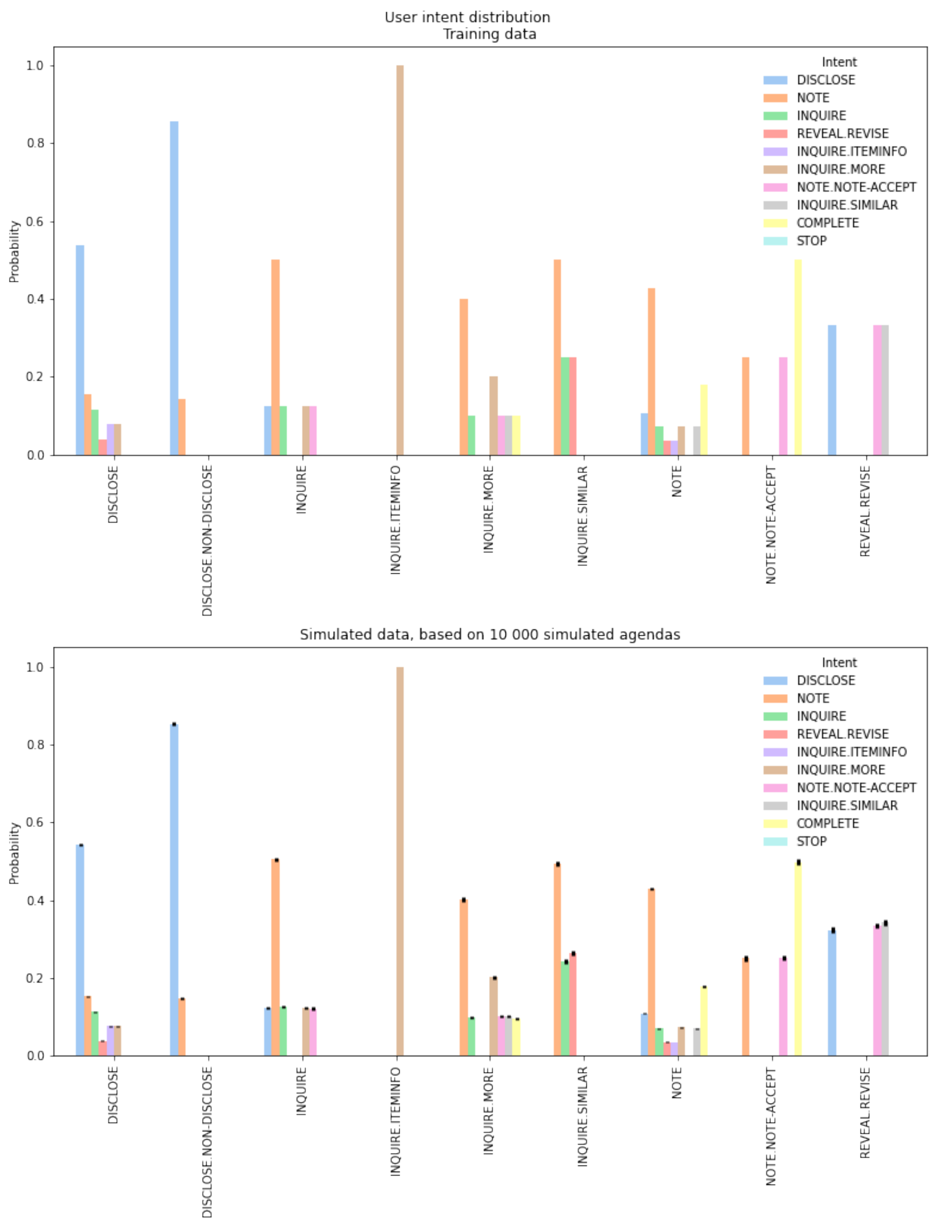


Figure 5.9: Intent distribution of the full set of data. The simulated agendas closely resemble the training data. Note the small error bars. These indicate standard deviation.

to IAI MovieBot limitation in conjunction with the manual annotation process, we did not simulate more conversations in this experiment and focused our efforts on the experiment with static agenda. Importantly, in this experiment, the initialized user agenda is different between the conversations. Furthermore, note that some of these persona-context pairs result in the same *max_retries* for the persona. For example, personas **P1**, **P2** and **P3** paired with either context (**R0_W0**,**T2**) or (**R0_W0**,**T3**) result in three users with *max_retries* of 0, 1 and 1 respectively. Therefore, the metrics for these persona-context pairs with the same value for *max_retries* should be similar. However, considering the resulting metrics for each persona in contexts (**R0_W0**,**T1**) and **no_context** in

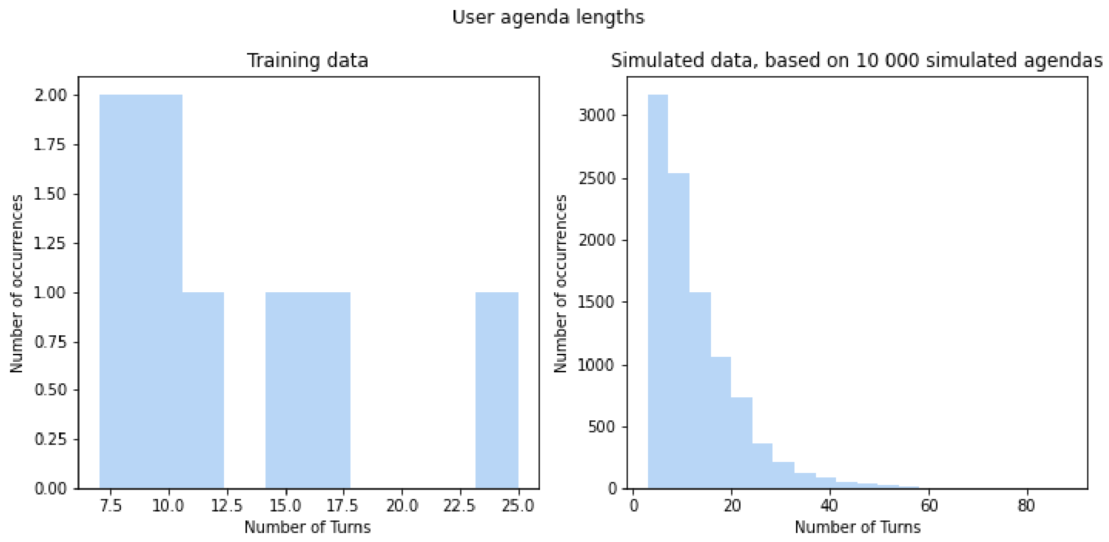


Figure 5.10: Agenda lengths from the full set of data. The simulated agendas resemble the training data. Because of the probability aspect, longer conversations do occur.

Table 5.4: Comparison of characteristics of dialogues with different contexts, and dynamic agenda. The standard deviation is presented alongside the values. Color shading is used to indicate user patience. A darker shade indicates a more patient user. Part 1

R0_W0	**AvgTurns**			**AvgSatisfaction**		
	P1	**P2**	**P3**	**P1**	**P2**	**P3**
T1	14.5 ± 9.94	14.2 ± 8.24	13.6 ± 7.56	1.47 ± 0.62	1.93 ± 0.93	2.00 ± 0.89
T2	13.2 ± 7.31	13.4 ± 7.94	13.6 ± 7.71	1.20 ± 0.40	2.00 ± 0.89	1.80 ± 0.98
T3	13.5 ± 7.78	13.3 ± 7.65	13.2 ± 7.52	1.80 ± 0.75	1.40 ± 0.80	1.20 ± 0.40
R1_W1	**P1**	**P2**	**P3**	**P1**	**P2**	**P3**
T1	12.9 ± 7.14	12.6 ± 7.43	12.6 ± 8.14	1.87 ± 0.88	1.53 ± 0.88	1.33 ± 0.60
T2	12.7 ± 8.08	12.6 ± 8.03	12.7 ± 8.38	1.20 ± 0.40	1.20 ± 0.40	2.00 ± 0.89
T3	12.7 ± 8.29	12.9 ± 8.94	13.2 ± 8.92	1.00 ± 0.00	2.20 ± 0.98	1.00 ± 0.00
no_context	13.1 ± 8.84	13.1 ± 8.73	13.1 ± 8.64	1.20 ± 0.40	2.20 ± 0.98	1.60 ± 0.49

Tables 5.4 and 5.5, it is not obvious that these metrics are fluctuations of essentially the same configuration. In fact, persona **P1** should have similar metrics across all contexts except (**R1_W1**,**T3**), however again, this is not clearly shown by the metrics.

AvgTurns. According to our results for this metric, users that are more patient tend to have shorter conversations. This does not align with our hypothesis, in which we stated that the opposite effect should be observed. However, notice that the standard deviations are large, which could be an indicator of little data and/or agendas with notably different lengths.

AvgSatisfaction. This metric helps us understand users' satisfaction with IAI MovieBot. Our understanding of the obtained results is that users' *max_retries* property does not impact these satisfaction scores. However, categorizing these measurements by persona, we find that users with persona **P1** is generally least satisfied with the conversations,

Table 5.5: Comparison of characteristics of dialogues with different contexts, and dynamic agenda. The standard deviation is presented alongside the values. Color shading is used to indicate user patience. A darker shade indicates a more patient user. Part 2.

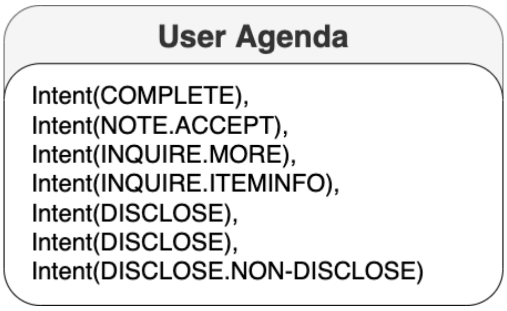
R0_W0	**AvgReward**			**AvgSuccess**		
	P1	**P2**	**P3**	**P1**	**P2**	**P3**
T1	2.20 ± 2.86	1.93 ± 2.67	2.33 ± 2.62	0.67 ± 0.13	0.72 ± 0.16	0.72 ± 0.15
T2	2.20 ± 2.71	0.00 ± 0.00	1.60 ± 3.20	0.67 ± 0.11	0.55 ± 0.08	0.57 ± 0.15
T3	3.60 ± 3.50	2.60 ± 2.87	3.00 ± 2.76	0.62 ± 0.18	0.44 ± 0.17	0.52 ± 0.26
R1_W1	**P1**	**P2**	**P3**	**P1**	**P2**	**P3**
T1	2.60 ± 2.44	3.00 ± 2.92	2.67 ± 2.65	0.71 ± 0.11	0.64 ± 0.13	0.53 ± 0.19
T2	3.20 ± 3.92	4.40 ± 3.01	1.80 ± 2.71	0.59 ± 0.12	0.42 ± 0.12	0.54 ± 0.19
T3	3.20 ± 3.54	0.80 ± 1.60	3.40 ± 2.80	0.45 ± 0.10	0.49 ± 0.17	0.40 ± 0.12
no_context	3.80 ± 2.71	0.40 ± 0.80	1.80 ± 1.94	0.65 ± 0.12	0.80 ± 0.07	0.76 ± 0.16

while users with persona **P2** generally receives highest satisfaction scores (although these users are still dissatisfied with the agent). This leads us to believe that the preference model somehow impacts a user's satisfaction with the agent.

AvgReward. The reward metric is dependent on how many turns there are in a conversation. As we mentioned in Sect. 5.1.1, this metric should move in the opposite direction of **AvgTurns**. However, the results in Table 5.5 indicate that this is not necessarily the case, especially if we consider users with persona **P1** across the different contexts. Notice that as **AvgTurns** decreases in contexts **R1_W1**, the reward increases as expected, however, the same observation cannot be made for contexts **R0_W0**.

AvgSuccess. Finally, we take a look at the average success of our simulations. Recall that we deem a turn successful if the turn goal was accomplished and otherwise unsuccessful. In our case, this metric will reveal whether the agent benefits from the user repeating their action or not. If the agent recovers, this metric will not decrease significantly, but on the other hand, if the agent does not recover, it will be clear based on this metric. Our results from the simulations are in accordance with our personal experience with IAI MovieBot, i.e., it does not benefit from the user repeating their action. This conclusion can be drawn by the fact that users with increased value of *max_retries* overall achieve a lower **AvgSuccess**, as shown in Table 5.5.

Overall, considering all four metrics, we find that our results are not helpful in identifying the impact of our context modeling. This could be due to the fact that we only sample 5 conversations per persona-context pair, and thus the amount of data is not enough to draw any conclusion. Additionally, the agendas in these conversations are dynamic and could thus vary greatly in length. In our next experiment, we address both of these issues.


User Agenda

Intent(COMPLETE),
Intent(NOTE.ACCEPT),
Intent(INQUIRE.MORE),
Intent(INQUIRE.ITEMINFO),
Intent(DISCLOSE),
Intent(DISCLOSE),
Intent(DISCLOSE.NON-DISCLOSE)

Figure 5.11: The static agenda used in the second experiment. This agenda was initialized by the interaction model based on our training data. Also note that this agenda is rather small, with only 7 actions.

Table 5.6: Comparison of characteristics of dialogues with different contexts, and static agenda. Color shading is used to indicate user patience. A darker shade indicates a more patient user. Part 1

R0_W0	**AvgTurns**			**AvgSatisfaction**		
	P1	**P2**	**P3**	**P1**	**P2**	**P3**
T1	7.50 ± 1.80	7.67 ± 2.87	7.78 ± 2.59	1.10 ± 0.30	1.45 ± 0.80	2.05 ± 0.92
T2	7.90 ± 2.50	8.10 ± 2.99	8.27 ± 2.86	1.15 ± 0.48	2.15 ± 0.91	2.25 ± 0.94
T3	8.37 ± 2.68	8.46 ± 2.93	8.61 ± 3.07	1.60 ± 0.86	1.75 ± 0.94	2.00 ± 1.00
R1_W1	**P1**	**P2**	**P3**	**P1**	**P2**	**P3**
T1	8.67 ± 2.99	8.63 ± 3.04	8.76 ± 3.35	1.00 ± 0.00	1.65 ± 0.91	2.25 ± 0.99
T2	8.84 ± 3.27	8.83 ± 3.39	8.95 ± 3.57	1.00 ± 0.00	1.95 ± 0.92	2.00 ± 1.14
T3	9.07 ± 3.50	9.15 ± 3.68	9.35 ± 3.94	1.50 ± 0.74	2.0 ± 0.95	1.90 ± 1.18

5.3.2.2 Static Agenda

In order to gain a better understanding of the impact of our context modeling, we simulate another 360 conversations with the same persona-context pairs as the experiment above (i.e., 20 dialogs per persona - context pair), excluding **no_context**. However, since context (**R0_W0**,**T1**) results in each persona having *max_retries* value of 0, the results obtained with this context should be similar to **no context**. Furthermore, these conversations are all simulated with the *same agenda*, meaning that the results should be less prone to randomness with regard to agenda items and lengths. The results are presented in Tables 5.6 and 5.7. The static agenda used in this experiment is shown in Fig 5.11.

AvgTurns. Considering the results we obtain with a static agenda, we observe that conversations become longer for users that are patient (i.e., higher *max_retries* value). Notice that dialogs with non-patient users are on average around 7-8 turns, which is coherent with the agenda length. Then, by increasing the patience of users, we find that the average conversation length also increases, as indicated by e.g., the user with persona P3 and context **R1_W1_T3** (9.35 ± 3.94 **AvgTurns**).

AvgSatisfaction. In terms of this metric, our results suggest that users are on average more satisfied with the agent in this experiment. The reason behind this is not immediately

Table 5.7: Comparison of characteristics of dialogues with different contexts, and static agenda. Color shading is used to indicate user patience. A darker shade indicates a more patient user. Part 2.

R0_W0	**AvgReward**			**AvgSuccess**		
	P1	**P2**	**P3**	**P1**	**P2**	**P3**
T1	4.50 ± 1.80	4.40 ± 2.56	4.10 ± 1.92	0.75 ± 0.14	0.66 ± 0.15	0.78 ± 0.09
T2	3.55 ± 2.01	3.90 ± 2.55	3.10 ± 1.76	0.75 ± 0.12	0.55 ± 0.16	0.67 ± 0.09
T3	3.15 ± 1.01	3.80 ± 2.62	2.50 ± 2.13	0.76 ± 0.12	0.57 ± 0.12	0.64 ± 0.12
R1_W1	**P1**	**P2**	**P3**	**P1**	**P2**	**P3**
T1	4.10 ± 1.84	3.90 ± 2.47	2.30 ± 2.03	0.75 ± 0.12	0.59 ± 0.10	0.58 ± 0.10
T2	3.95 ± 1.63	3.90 ± 2.41	1.75 ± 1.89	0.76 ± 0.11	0.58 ± 0.13	0.56 ± 0.13
T3	2.75 ± 1.67	2.85 ± 2.41	1.30 ± 1.93	0.66 ± 0.11	0.50 ± 0.10	0.47 ± 0.12

obvious, and requires a deeper analysis of the agenda and the general behavior of IAI MovieBot. Generally, IAI MovieBot tends to start the conversations with an introduction, followed by questions about the user's movie genre and tag preferences. This is generally followed up by a recommendation - the rest of the conversation depends more on the user actions. Bearing this in mind and taking a deeper look at the static agenda, we find that the first three actions on this agenda allows for this scenario to occur. Considering that the agenda is initialized based on previous dialogs with the system, this scenario is not unrealistic, and in fact, our simulated agendas in Sect. 5.3.1 proved that it is more probable than not.

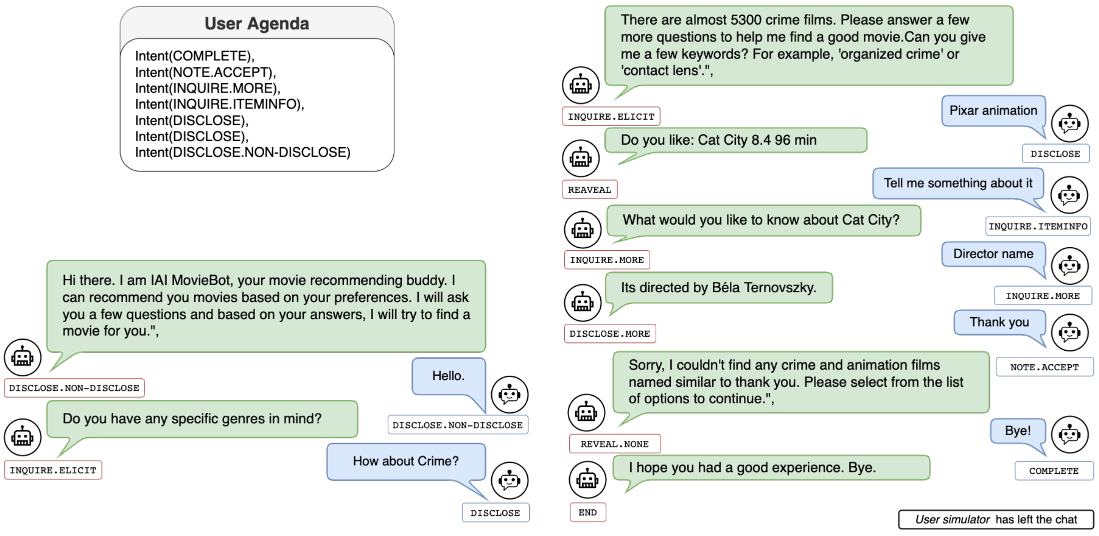
AvgReward. Recall that we mentioned that this metric should theoretically move in the opposite of **AvgTurns**. However, this was not the case in the previous experiment, with dynamic agendas. In this experiment however, we find that the metric does in fact decrease with increasing **AvgTurns**. From Tables 5.6 and 5.7, we find that when we add corresponding **AvgTurns** and **AvgReward** values, the result is a number close to 12, i.e., what we initially set *Full* as (see Sect- 5.1.1).

AvgSuccess. Overall, we note that this metric stay consistent with its counterpart in the experiment with dynamic agendas, although they fluctuate less in this experiment. Our intuition is that this is due to the fact that we simulate 15 additional conversations per user, compared to the previous experiment.

The experiment with a static agenda shows that our context modeling has a measurable impact on the simulated conversations.

5.3.2.3 Problems with IAI MovieBot

Before we move on to the WoZ experiment, we present some of the issues concerning IAI MovieBot. In general, there have been many inconsistencies with it. Some of the issues/bugs were fixed by the authors of this thesis, however, as this was not part of



User Agenda

Intent(COMPLETE),
Intent(NOTE.ACCEPT),
Intent(INQUIRE.MORE),
Intent(INQUIRE.ITEMINFO),
Intent(DISCLOSE),
Intent(DISCLOSE),
Intent(DISCLOSE.NON-DISCLOSE)

Hi there. I am IAI MovieBot, your movie recommending buddy. I can recommend you movies based on your preferences. I will ask you a few questions and based on your answers, I will try to find a movie for you.",

DISCLOSE.NON-DISCLOSE

Hello.

DISCLOSE.NON-DISCLOSE

Do you have any specific genres in mind?

INQUIRE.ELICIT

How about Crime?

DISCLOSE

There are almost 5300 crime films. Please answer a few more questions to help me find a good movie.Can you give me a few keywords? For example, 'organized crime' or 'contact lens'.",

INQUIRE.ELICIT

Pixar animation

DISCLOSE

Do you like: Cat City 8.4 96 min

REAVEAL

Tell me something about it

INQUIRE.ITEMINFO

What would you like to know about Cat City?

INQUIRE.MORE

Director name

INQUIRE.MORE

Its directed by Béla Ternovszky.

DISCLOSE.MORE

Thank you

NOTE.ACCEPT

Sorry, I couldn't find any crime and animation films named similar to thank you. Please select from the list of options to continue.",

REVEAL.NONE

Bye!

COMPLETE

I hope you had a good experience. Bye.

END

User simulator has left the chat

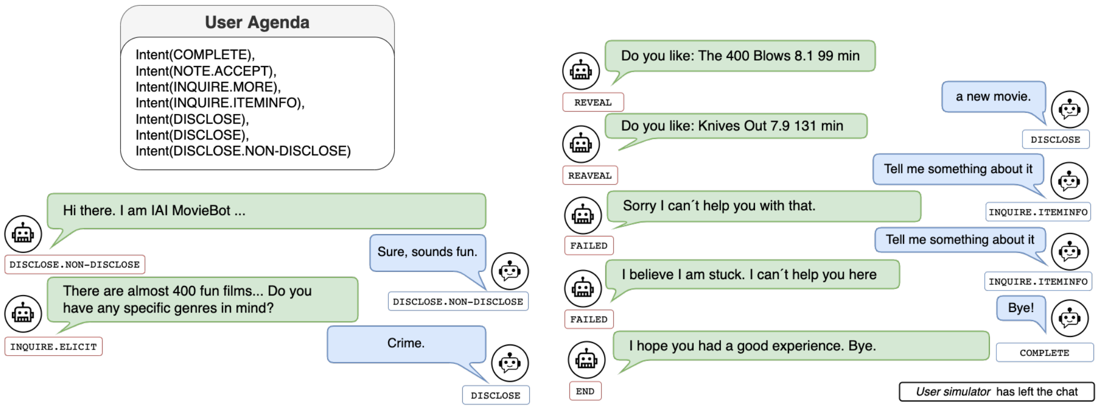
Figure 5.12: One of the successful conversations with IAI MovieBot.

the project scope, we did not attend to the bigger issues. The main issue concerns IAI MovieBot's intent classification and query understanding. User utterances such as "I like this recommendation" does not always work, although this exact utterance was part of the multi-modal interface of IAI MovieBot previously, as a button. We suspected that the underlying logic was causing this, however, our investigation showed that the logic is the same. This proved to be challenging, as IAI MovieBot would not always act as anticipated. These issues are illustrated in Figs. 5.12 and 5.13. In the former figure, we show an example of a successful dialog taken from the experiment with static agenda. In the latter, an unsuccessful conversation is shown, from the same experiment. Notice that in the successful dialog, IAI MovieBot understands the utterance "Tell me something about it", but fails to understand the same utterance in the unsuccessful dialog. The conversation ends up with the agent becoming stuck and the simulated user runs out of patience, thus sampling a new action based on the agent action *FAILED*. In this case, the sampled action was *COMPLETE* which indicates that the user wants to end the conversation. IAI MovieBot understands this utterance and terminates the conversation.

These issues are somewhat expected since IAI MovieBot is a stale project.

5.3.3 Wizard of Oz

In the WoZ setting, we "pretend" to be the conversational agent. Specifically, in each of these WoZ conversations, one of the thesis authors acts as the conversational agent. Using a terminal-based interface, a response is formulated manually and sent to the simulated user, mimicking the behavior of an ideal conversational agent. We instantiate two users, one with persona **P__A** and context **R1__W1__T3** and another with persona **P__B** and



User Agenda

Intent(COMPLETE),
Intent(NOTE.ACCEPT),
Intent(INQUIRE.MORE),
Intent(INQUIRE.ITEMINFO),
Intent(DISCLOSE),
Intent(DISCLOSE),
Intent(DISCLOSE.NON-DISCLOSE)

Hi there. I am IAI MovieBot ... — DISCLOSE.NON-DISCLOSE

Sure, sounds fun. — DISCLOSE.NON-DISCLOSE

There are almost 400 fun films... Do you have any specific genres in mind? — INQUIRE.ELICIT

Crime. — DISCLOSE

Do you like: The 400 Blows 8.1 99 min — REVEAL

a new movie. — DISCLOSE

Do you like: Knives Out 7.9 131 min — REAVEAL

Tell me something about it — INQUIRE.ITEMINFO

Sorry I can´t help you with that. — FAILED

Tell me something about it — INQUIRE.ITEMINFO

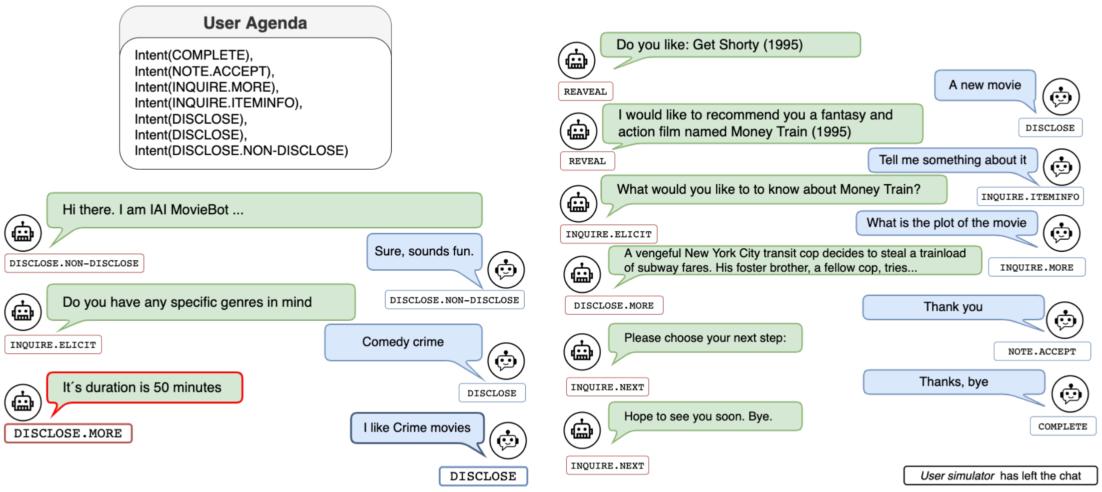
I believe I am stuck. I can´t help you here — FAILED

Bye! — COMPLETE

I hope you had a good experience. Bye. — END

User simulator has left the chat

Figure 5.13: One of the failed conversations with IAI MovieBot.



User Agenda

Intent(COMPLETE),
Intent(NOTE.ACCEPT),
Intent(INQUIRE.MORE),
Intent(INQUIRE.ITEMINFO),
Intent(DISCLOSE),
Intent(DISCLOSE),
Intent(DISCLOSE.NON-DISCLOSE)

Hi there. I am IAI MovieBot ... — DISCLOSE.NON-DISCLOSE

Sure, sounds fun. — DISCLOSE.NON-DISCLOSE

Do you have any specific genres in mind — INQUIRE.ELICIT

Comedy crime — DISCLOSE

It´s duration is 50 minutes — DISCLOSE.MORE

I like Crime movies — DISCLOSE

Do you like: Get Shorty (1995) — REAVEAL

A new movie — DISCLOSE

I would like to recommend you a fantasy and action film named Money Train (1995) — REVEAL

Tell me something about it — INQUIRE.ITEMINFO

What would you like to to know about Money Train? — INQUIRE.ELICIT

What is the plot of the movie — INQUIRE.MORE

A vengeful New York City transit cop decides to steal a trainload of subway fares. His foster brother, a fellow cop, tries... — DISCLOSE.MORE

Thank you — NOTE.ACCEPT

Please choose your next step: — INQUIRE.NEXT

Thanks, bye — COMPLETE

Hope to see you soon. Bye. — INQUIRE.NEXT

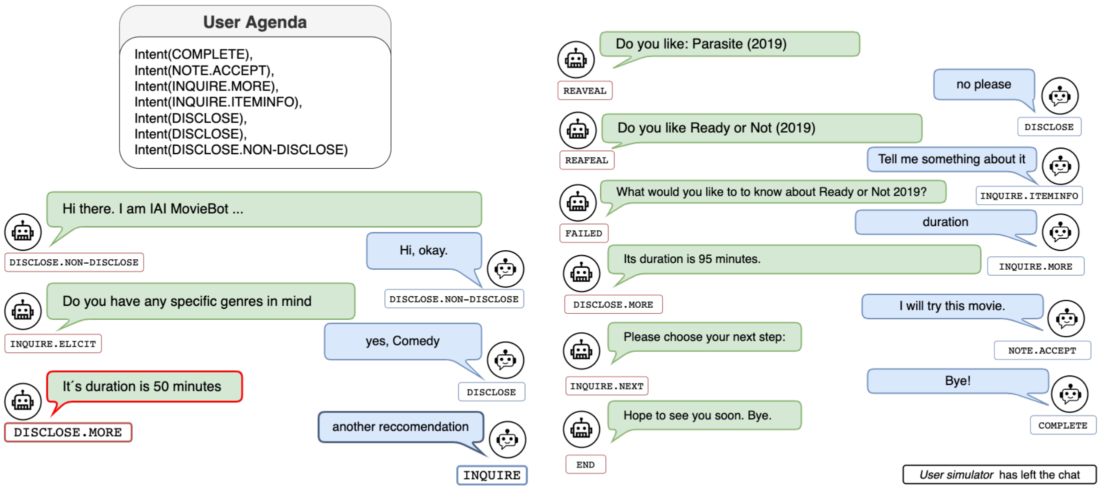
User simulator has left the chat

Figure 5.14: Example dialogue in the WoZ setting. The user has patience above 0. Thus it will retry the same intent when the WoZ agent fails to answer with an expected intent. The unexpected intent is highlighted with a red border.

context **R0__W0__T3**. We collect 10 dialogs per user, i.e., 20 in total. Furthermore, agendas in this experiment are dynamic.

Considering the persona *max_retries* property, we note that the conversations with persona A should last longer since the simulator will be more patient and retry the same action 3 times before giving up and trying a different action.

However, we do note that a consequence of better NLU is that the user will not repeat itself as frequently, since the WoZ agent understands what the user is trying to achieve. This would lead to both users having similar values for the **AvgTurns** and **AvgSuccess** metrics. In fact, this is what we observe in Table 5.8. This premise is further backed by the **AvgReward** metric, which is 1.00 ± 1.79 for persona **P__A** and 0.80 ± 1.54. Considering that *Full* is set to 12, this metric should result in -1, however, since we place a lower bound on this metric, it will always be above 0. Furthermore, its value


User Agenda

Intent(COMPLETE),
Intent(NOTE.ACCEPT),
Intent(INQUIRE.MORE),
Intent(INQUIRE.ITEMINFO),
Intent(DISCLOSE),
Intent(DISCLOSE),
Intent(DISCLOSE.NON-DISCLOSE)

Hi there. I am IAI MovieBot ... — DISCLOSE.NON-DISCLOSE

Hi, okay. — DISCLOSE.NON-DISCLOSE

Do you have any specific genres in mind — INQUIRE.ELICIT

yes, Comedy — DISCLOSE

It´s duration is 50 minutes — DISCLOSE.MORE

another reccomendation — INQUIRE

Do you like: Parasite (2019) — REAVEAL

no please — DISCLOSE

Do you like Ready or Not (2019) — REAFEAL

Tell me something about it — INQUIRE.ITEMINFO

What would you like to to know about Ready or Not 2019?
FAILED

duration — INQUIRE.MORE

Its duration is 95 minutes. — DISCLOSE.MORE

I will try this movie. — NOTE.ACCEPT

Please choose your next step: — INQUIRE.NEXT

Bye! — COMPLETE

Hope to see you soon. Bye. — END

User simulator has left the chat

Figure 5.15: Example dialogue in the WoZ setting. The user has patience equal to 0. Thus it will not retry the same intent when the WoZ agent fails to answer with an excepted intent. The unexpected intent is highlighted with a red border.

Table 5.8: Comparison of characteristics of dialogues with different contexts and static agenda, with WoZ evaluation. Color shading is used to indicate user patience. A darker shade indicates a more patient user.

AvgReward		AvgSuccess	
P_A	**P_B**	**P_A**	**P_B**
1.00 ± 1.79	0.80 ± 1.54	0.74 ± 0.20	0.78 ± 0.11

AvgTurns		AvgSatisfaction	
A	**B**	**A**	**B**
13.1 ± 8.47	13.2 ± 8.38	1.90 ± 0.83	1.20 ± 0.60

can be explained by the fact that **AvgTurns** has a large standard deviation for both personas, resulting in some conversations with a high reward which increases the overall **AvgReward**. In terms of **AvgSatisfaction**, we do not observe a significant deviation from the other two experiments, which is expected, as the WoZ agent utterances are similar to IAI MovieBot.

Figures 5.14 and 5.15 illustrates two conversations in the WoZ setting. In order to observe the effect of the contexts, we as the WoZ agent intentionally misinterpret the user utterance. The utterance boxes with red borders indicate the turns where this scenario occurs. The difference between these dialogs is essentially how the user reacts to the agent misinterpreting. In Fig. 5.14, the user with high patience retries its previous action, while the non-patient user in Fig. 5.15 immediately samples a new action, thus moving away from its initial agenda. Although this happens for only one turn, in reality, a general conversational agent could misinterpret the user's utterance several times in a row, which ends up in a less realistic dialog.

5.4 Discussion

In this section, we analyze our results with regards to the research questions we formulated in Chap. 1.

5.4.1 Impact of Contexts

In Sect. 5.1.1, we hypothesized that **AvgTurns**, **AvgReward**, and **AvgSuccess** will be influenced depending on the performance of the conversational agent. For example, IAI MovieBot does not perform well when a user restates the same intent after the conversational agent fails to understand the user. This is reflected in the metrics we considered. The increase in **AvgTurns** is evident especially for users that are patient, i.e., have higher *max_retries* value. Furthermore, **AvgSuccess** and **AvgReward** decrease as a consequence. This hypothesis was backed by our results, especially the ones obtained by considering more data in a static agenda setting. This observations enables us to answer our first research question:

- **RQ1**: Do the additional context have any impact on the characteristics of conversation, and if so, what is the observed impact?

The simple answer is that it depends on the conversational agent in use. In our case, we experimented with an agent that had limited NLU capabilities which was reflected in the characteristics of conversation, i.e., the metrics mentioned above. However, we theorize that context will not affect these metrics if the conversational agent is capable at understanding which actions the user is performing. This scenario would not trigger the repetitive behavior of the user, thus not impacting the metrics. For this research question, we note that impact of the contexts on characteristics of conversation is inversely proportional to the agent's NLU capability.

- **RQ2**: Does more advanced context modeling lead to more realistic user simulation? Specifically, given a persona and varying contexts, are the observed effects realistic?

In general, to find out whether simulations are realistic or not, one would have to consider human evaluation. An example of this would be conducting a *Turing test*-like evaluation, as we mentioned in Chapter 1. However, we will attempt to give an answer based on our results. Consider Table 5.9, which shows the results obtained for persona **P3** across different contexts. We find that the observed effects by varying the context are realistic, as they reflect the patience of the user. For example, the difference between a non-patient

Table 5.9: Comparison of characteristics of dialogues with different contexts, and static agenda. Persona P3. Color shading is used to indicate user patience. A darker shade indicates a more patient user.

R0_W0	**AvgTurns P3**	**AvgSatisfaction P3**	**AvgReward P3**	**AvgSuccess P3**
T1	7.78 ± 2.59	2.05 ± 0.92	4.10 ± 1.92	0.78 ± 0.09
T2	8.27 ± 2.86	2.25 ± 0.94	3.10 ± 1.76	0.67 ± 0.09
T3	8.61 ± 3.07	2.00 ± 1.00	2.50 ± 2.13	0.64 ± 0.12
R1_W1				
T1	8.76 ± 3.35	2.25 ± 0.99	2.30 ± 2.03	0.58 ± 0.10
T2	8.95 ± 3.57	2.00 ± 1.14	1.75 ± 1.89	0.56 ± 0.13
T3	9.35 ± 3.94	1.90 ± 1.18	1.30 ± 1.93	0.47 ± 0.12

user (context **R0_W0_T1**) vs. a patient (context **R1_W1_T3**) is that the patient user tends to have longer conversations, which is reasonable. The overall satisfaction does not decrease by a lot, which indicates that longer conversations do not necessarily translate to less satisfaction with the system. This also indicates that our approach to computing **AvgReward** is not necessarily realistic, as it is biased towards shorter conversations. The **AvgSuccess** metric shows in this case that the agent has issues understanding the user, but we note that this could also indicate that the interaction model should include more advanced conversational flows. Therefore, our answer to this question is mixed. The effects on the objective measures are realistic, however, we cannot conclude the same for subjective measures, i.e., human evaluation. Considering Figs. 5.12 and 5.13, we believe that it is more realistic for a user to retry their action at least once before trying another action. However, this is subjective and these figures show only two conversations out of 400+. Furthermore, considering the WoZ dialogs in Figs. 5.14 and 5.15, we personally find that the conversation with the user with context modeling is more realistic than without context modeling. This is due to the fact that the user with context modeling attempts to steer the conversation in the right direction, while the other user samples a random action to perform next. We present these dialogs as anecdotal evidence that more advanced context modeling leads to more realistic simulation, leaving a proper evaluation with humans for future work.

Chapter 6

Conclusions

This chapter provides an overview of the thesis. We start by presenting our conclusion in Sect. 6.1. Section 6.2 presents pointers for future work. Finally, in Sect. 6.3 we reflect on our work.

6.1 Conclusion

In this thesis, our goal was to identify new contexts that could be incorporated into user simulators to increase their realisticity. We formulated this goal by presenting two **RQs**. In order to conduct our research, we continued development on two previous projects, *DialogueKit* and *UserSimCRS*. Both projects required the implementation of key functionality that was previously missing. Existing user modeling in *UserSimCRS* had to be revisited and extended with the new contexts, i.e., temporal, relational, and conversational contexts. These contexts were used to affect the simulated users' cooperativeness score, which we incorporated into different personas.

We extended *DialogueKit* with core conversational concepts and improved the overall usability aspect of the package, enabling it to be used by others to develop conversational assistants. The package is available on PyPi.[1]

Our results showed that the characteristics of conversation are indeed affected by the temporal and relational contexts (**RQ1**). However, our results were not conclusive enough to answer our second **RQ**, i.e., whether more advanced context modeling leads to more realistic user simulation. While the metrics we considered are affected in a realistic way, a conclusion to this question cannot be reached without human evaluation.

[1] `https://pypi.org/project/dialoguekit/`

We believe this is due to the following: the agenda initialization is performed by utilizing prior annotated dialogues. Since the agenda is probabilistic, it could be initialized to e.g., ask for a directors name after having *disclosed* genre preference. That is, we as the agent ask the user for genres. The user provides genres, and next, we ask for keywords in order to narrow down the search space. Since the turn-level goal was accomplished, the user pulls the next action off the agenda, and asks for director name. As IAI MovieBot has no way of telling the user that it has not recommended a movie yet, and can thus not retrieve any directors, we imitate this behaviour and simply ask for keyword preferences again. This time, the turn-level goal is not accomplished and thus the user will repeat itself.

6.2 Future Directions

In this section, we identify key aspects for future work. These are:

1. **Better interaction modeling**: Improvements to the current interaction modeling can be made by considering other intent schemes that are based on large user studies. Lyu et al. [88] extend the taxonomies in [89] by studying dialogs between users and CRSs. Although these datasets are partially synthetic, the resulting taxonomies are more fine-grained and may be interesting to evaluate in user simulators. As we mentioned in the introduction, the interaction scheme in UserSimCRS can easily be replaced. Simultaneously, we note that the interaction taxonomies in [88, 89] contain intents that include preference, e.g., "SEEN". Agenda-based user simulators suffer from this.

2. **Agenda initialization**: In agenda-based user simulators, the agenda is initialized before the dialog starts. The initialization is based on past user-system interactions, e.g., the annotated dialogs that UserSimCRS requires. However, this can cause issues that disrupt the dialog flow and the realisticity of the simulated dialogs. Consider the following situation. A user is talking to a conversational recommender system, e.g., IAI MovieBot. The user has revealed their preferences (intent: *DISCLOSE*), and the next action on the agenda is *INQUIRE.MORE*, i.e., the user is expecting a recommendation from the agent and will ask for more information regarding this recommendation. However, if the agent instead asks for more user preferences (i.e., *INQUIRE.ELICIT*), the conversation quickly becomes unrealistic. This is due to the fact that the intent scheme includes this agent intent as a possible agent response, but the agenda is already initialized with a different dialog “path” in mind. This issue becomes more evident with user

intents that are dependent on the preference model (e.g., *NOTE* intent in our case). Currently, the next intent in the agenda would be based on *NOTE* instead of e.g., *NOTE.LIKE/NOTE.DISLIKE/NOTE.YES/NOTE.NO*, which can often lead to unexpected dialog flow. Therefore, investigating other structures for storing the agenda that allows modeling alternative dialogue paths could be of interest. Alternatively, one can investigate whether agenda initialization is necessary or not.

3. **NLG**: The NLG module of DialogueKit is currently template-based. In the future, DialogueKit should be extended to support large language models that are capable of *generating* real user responses based on intent, satisfaction, and annotations.

4. **NLU**: The DIET classifier implementation in DialogueKit is limited as the pipeline includes only *WhitespaceTokenizer* and *CountVectorsFeaturizer*. A more advanced DIET pipeline would lead to better intent and entity classification.

5. **Persona generation / context modeling**: Our approach to persona generation is based on intuition and not grounded in real data. The cooperative bonuses and the distributions that were used to sample context variables should reflect users realistically. This could be facilitated by studies on user patience in different contexts.

6. **Evaluation**: Although we compute the same metrics as in [1], we were not able to compare these metrics against metrics obtained with real users. In the future, this could be accomplished by using crowdsourcing to collect dialogs with real users and compute the same metrics for these dialogs. Furthermore, a *Turing test* could be advantageous for evaluating the user simulator's realisticity.

6.3 Reflection

To conclude our thesis, we would like to reflect upon our approach and takeaways.

Briefly explained, we started off by conducting a comprehensive literature review and investigated what other researchers in the field had experimented with. In hindsight, this was very helpful in order to gain a better understanding of our task, and what worked vs. not. This also helped us understand the importance of user simulators.

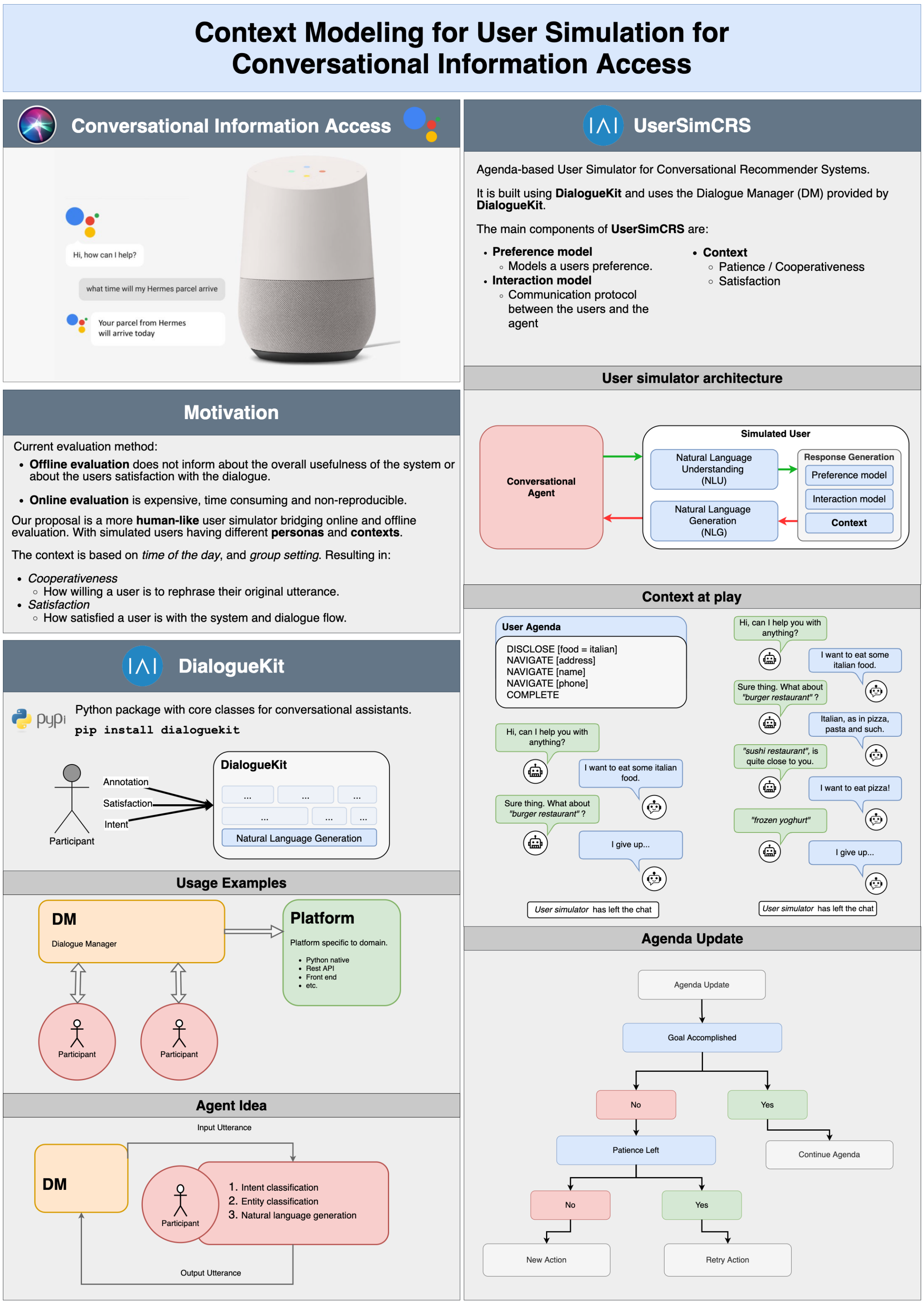
We decided early on to follow the approach in [1]. We adapted the interaction and preference models from [1], including the intent scheme which was based on [51]. After a long development process in both *DialogueKit* and *UserSimCRS*, we were able to run simulations with additional context.

In hindsight, we should have spent more time on interaction modeling and investigating other interaction schemes, e.g., the user study in [88]. The same applies to agenda initialization techniques, although most of the literature here supported the current technique. Furthermore, we wish we had spent more time at exploring replacements for IAI MovieBot instead of addressing its issues, as we faced a lot of difficulties due to its limited capability.

Lastly, to our knowledge, there are no previous works that experiment with the same contexts as we do in a conversational setting. In general, there is a lack of information available on how these contexts (temporal and relational) can be quantified in user modeling. In retrospect, it would have been beneficial to conduct user studies to learn which contexts impact user cooperativeness and patience, and how to quantify these.

Appendix A

Poster



Context Modeling for User Simulation for Conversational Information Access

Conversational Information Access

Hi, how can I help?
what time will my Hermes parcel arrive
Your parcel from Hermes will arrive today

Motivation

Current evaluation method:

- **Offline evaluation** does not inform about the overall usefulness of the system or about the users satisfaction with the dialogue.
- **Online evaluation** is expensive, time consuming and non-reproducible.

Our proposal is a more **human-like** user simulator bridging online and offline evaluation. With simulated users having different **personas** and **contexts**.

The context is based on *time of the day*, and *group setting*. Resulting in:

- *Cooperativeness*
  - How willing a user is to rephrase their original utterance.
- *Satisfaction*
  - How satisfied a user is with the system and dialogue flow.

DialogueKit

Python package with core classes for conversational assistants.

`pip install dialoguekit`

Usage Examples

Agent Idea

UserSimCRS

Agenda-based User Simulator for Conversational Recommender Systems.

It is built using **DialogueKit** and uses the Dialogue Manager (DM) provided by **DialogueKit**.

The main components of **UserSimCRS** are:

- **Preference model**
  - Models a users preference.
- **Interaction model**
  - Communication protocol between the users and the agent
- **Context**
  - Patience / Cooperativeness
  - Satisfaction

User simulator architecture

Context at play

User Agenda
DISCLOSE [food = italian]
NAVIGATE [address]
NAVIGATE [name]
NAVIGATE [phone]
COMPLETE
Hi, can I help you with anything?
I want to eat some italian food.
Sure thing. What about "burger restaurant"?
I give up...
User simulator has left the chat
Hi, can I help you with anything?
I want to eat some italian food.
Sure thing. What about "burger restaurant"?
Italian, as in pizza, pasta and such.
"sushi restaurant", is quite close to you.
I want to eat pizza!
"frozen yoghurt"
I give up...
User simulator has left the chat

Agenda Update

Figure A.1: Poster used for the poster presentation.

Appendix B

Instantiated intent schema

```
# Possible user intents with optional description.
user_intents:
  COMPLETE:
  DISCLOSE:
    - DISCLOSE.NON-DISCLOSE
  REVEAL:
    - REVEAL.EXPAND
    - REVEAL.REFINE
    - REVEAL.REVISE
  INQUIRE:
    - INQUIRE.SIMILAR
    - INQUIRE.ITEMINFO
    - INQUIRE.MORE
  NAVIGATE:
    - NAVIGATE.MORE
  NOTE:
    - NOTE.LIKE
    - NOTE.DISLIKE
    - NOTE.YES
    - NOTE.NO
    - NOTE.ACCEPT
  INTERROGATE:
  COMPLETE:

user_preference_intents:
  NOTE:
    POSITIVE:
      - NOTE.LIKE
    NEGATIVE:
      - NOTE.DISLIKE
    CONSUMED:
      - NOTE.YES
```

```

user_add_preference_intents:
    - REVEAL.EXPAND
    - REVEAL.REFINE

user_remove_preference_intents:
  - REVEAL.REVISE

# Possible agent intents with optional description.
agent_intents:
  END:
  DISCLOSE:
    - DISCLOSE.NON-DISCLOSE
    - DISCLOSE.MORE
  INQUIRE:
    - INQUIRE.ELICIT
    - INQUIRE.MORE
    - INQUIRE.NEXT
  REVEAL:
    - REVEAL.NONE
    - REVEAL.SIMILAR
  RECORD:
  END:

# List of agent intents (including sub-intents) that elicit preferences.
agent_elicit_intents:
  - INQUIRE
  - INQUIRE.ELICIT

# List of agent intents (including sub-intents) that are for set retrieval.
agent_set_retrieval:
  - REVEAL
  - REVEAL.NONE
  - REVEAL.SIMILAR
  - DISCLOSE.MORE

# Expected agent intents in response to a (simulated) user intent.
expected_responses:
  DISCLOSE.NON-DISCLOSE:
    - INQUIRE
    - INQUIRE.ELICIT
    - DISCLOSE.NON-DISCLOSE
  DISCLOSE:
    - INQUIRE.ELICIT
    - REVEAL
    - REVEAL.NONE
  REVEAL.REVISE:
    - INQUIRE.ELICIT
    - REVEAL
    - REVEAL.NONE
  REVEAL.REFINE:
    - INQUIRE.ELICIT
    - REVEAL
    - REVEAL.NONE
  REVEAL.EXPAND:
    - INQUIRE.ELICIT
    - REVEAL
    - REVEAL.NONE
  NOTE:
    - INQUIRE.NEXT
    - INQUIRE.MORE
    - END
    - REVEAL
    - REVEAL.SIMILAR
  NOTE.YES:
    - INQUIRE.ELICIT
    - REVEAL
    - REVEAL.SIMILAR
  NOTE.NO:
    - INQUIRE.ELICIT
    - REVEAL
    - REVEAL.SIMILAR
  NOTE.LIKE:
    - INQUIRE.NEXT
    - REVEAL
    - REVEAL.SIMILAR
  NOTE.DISLIKE:
    - REVEAL
    - REVEAL.SIMILAR
    - REVEAL.NONE
  NOTE.ACCEPT:
    - INQUIRE.NEXT
    - INQUIRE.MORE
    - END
  INQUIRE:
    - INQUIRE.ELICIT
    - REVEAL
    - REVEAL.SIMILAR
    - REVEAL.NONE
  INQUIRE.SIMILAR:
    - REVEAL
    - REVEAL.SIMILAR
    - REVEAL.NONE
  INQUIRE.ITEMINFO:
    - INQUIRE.MORE
```

```
INQUIRE.MORE:
  - DISCLOSE.MORE
COMPLETE:
  - END
```

Listing B.1: The instantiated intent schema.

Bibliography

[1] Shuo Zhang and Krisztian Balog. Evaluating Conversational Recommender Systems via User Simulation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, pages 1512–1520, 2020.

[2] Hamed Zamani, Johanne R. Trippas, Jeff Dalton, and Filip Radlinski. *Conversational Information Seeking.* cs.IR. 2022.

[3] Dan Jurafsky and James H. Martin. *Speech and Language Processing.* 2021. URL https://web.stanford.edu/~jurafsky/slp3/.

[4] David R. Traum and Peter A. Heeman. *Utterance units in spoken dialogue*, volume 1236 of *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics.* Springer, 1997.

[5] Krisztian Balog. Conversational AI from an Information Retrieval Perspective: Remaining Challenges and a Case for User Simulation. In *Proceedings of the 2nd International Conference on Design of Experimental Search & Information REtrieval Systems*, DESIRES '21, pages 1–11, 2021.

[6] Joseph Weizenbaum. ELIZA—A Computer Program for the Study of Natural Language Communication between Man and Machine. *Communications of the ACM*, 9:36–45, 1966.

[7] Kenneth Mark Colby, Sylvia Weber, and Franklin Dennis Hilf. Artificial Paranoia. *Artif. Intell.*, 2:1–25, 1971.

[8] Jianfeng Gao, Michel Galley, and Lihong Li. Neural Approaches to Conversational AI. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, page 1371–1374, 2018.

[9] Chen Qu, Liu Yang, Minghui Qiu, W. Bruce Croft, Yongfeng Zhang, and Mohit Iyyer. BERT with History Answer Embedding for Conversational Question Answering. *CoRR*, abs/1905.05412, 2019.

[10] Drew Meyer. Introducing Alexa Conversations (beta), a New AI-Driven Approach to Providing Conversational Experiences That Feel More Natural, 2020. URL https://developer.amazon.com/en-US/blogs/alexa/alexa-skills-kit/2020/07/introducing-alexa-conversations-beta-a-new-ai-driven-approach-to-providing-conversatio Accessed: 2022-04-15.

[11] Amazon Inc. About Alexa Conversations, 2022. URL https://www.developer.amazon.com/en-US/docs/alexa/conversations/about-alexa-conversations.html. Accessed: 2022-04-15.

[12] Amazon Inc. How Alexa Conversations Works, 2022. URL https://www.developer.amazon.com/en-US/docs/alexa/conversations/how-alexa-conversations-works.html. Accessed: 2022-04-15.

[13] J. Shane Culpepper, Fernando Diaz, and Mark D. Smucker. Research Frontiers in Information Retrieval: Report from the Third Strategic Workshop on Information Retrieval in Lorne (SWIRL 2018). *SIGIR Forum*, 52(1):34–90, 2018.

[14] Dietmar Jannach, Ahtsham Manzoor, Wanling Cai, and Li Chen. A Survey on Conversational Recommender Systems. *CoRR*, abs/2004.00646:1–35, 2021.

[15] Chongming Gao, Wenqiang Lei, Xiangnan He, Maarten de Rijke, and Tat-Seng Chua. Advances and Challenges in Conversational Recommender Systems: A Survey. *CoRR*, abs/2101.09459:1–33, 2021.

[16] Qibin Chen, Junyang Lin, Yichang Zhang, Ming Ding, Yukuo Cen, Hongxia Yang, and Jie Tang. 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)*, EMNLP-IJCNLP '19, pages 1803–1813, 2019.

[17] Dongyeop Kang, Anusha Balakrishnan, Pararth Shah, Paul Crook, Y-Lan Boureau, and Jason Weston. Recommendation as a Communication Game: Self-Supervised Bot-Play for Goal-oriented Dialogue. 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)*, EMNLP-IJCNLP '19, pages 1951–1961, 2019.

[18] Zhao Yan, Nan Duan, Peng Chen, Ming Zhou, Jianshe Zhou, and Zhoujun Li. Building Task-Oriented Dialogue Systems for Online Shopping. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI '17, pages 4618—4625, 2017.

[19] Zuohui Fu, Yikun Xian, Yongfeng Zhang, and Yi Zhang. *Tutorial on Conversational Recommendation Systems*, page 751–753. ACM '20. Association for Computing Machinery, 2020.

[20] Yongfeng Zhang, Xu Chen, Qingyao Ai, Liu Yang, and W. Croft. Towards Conversational Search and Recommendation: System Ask, User Respond. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, pages 177–186, 2018.

[21] Yueming Sun and Yi Zhang. Conversational Recommender System. In *The 41st International ACM SIGIR Conference on Research &; Development in Information Retrieval*, SIGIR '18, page 235–244. Association for Computing Machinery, 2018.

[22] Shirley Anugrah Hayati, Dongyeop Kang, Qingxiaoyang Zhu, Weiyan Shi, and Zhou Yu. INSPIRED: Toward Sociable Recommendation Dialog Systems. *CoRR*, abs/2009.14306:1–20, 2020.

[23] Raymond Li, Samira Ebrahimi Kahou, Hannes Schulz, Vincent Michalski, Laurent Charlin, and Chris Pal. Towards Deep Conversational Recommendations. *CoRR*, abs/1812.07617:1–17, 2018.

[24] Filip Radlinski, Krisztian Balog, Bill Byrne, and Karthik Krishnamoorthi. Coached Conversational Preference Elicitation: A Case Study in Understanding Movie Preferences. In *Proceedings of the Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, SIGDIAL '19, pages 1–8, 2019.

[25] Jesse Dodge, Andreea Gane, Xiang Zhang, Antoine Bordes, Sumit Chopra, Alexander H. Miller, Arthur Szlam, and Jason Weston. Evaluating Prerequisite Qualities for Learning End-to-End Dialog Systems. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, ICLR '16, 2016.

[26] Johanne R. Trippas, Damiano Spina, Lawrence Cavedon, Hideo Joho, and Mark Sanderson. Informing the Design of Spoken Conversational Search: Perspective Paper. In *ACM SIGIR Conference on Human Information Interaction and Retrieval*, CHIIR '18, page 32–41, 2018.

[27] Liu Yang, Minghui Qiu, Chen Qu, Jiafeng Guo, Yongfeng Zhang, W. Bruce Croft, Jun Huang, and Haiqing Chen. Response Ranking with Deep Matching Networks and External Knowledge in Information-seeking Conversation Systems. *CoRR*, abs/1805.00188, 2018.

[28] Jeffrey Dalton, Chenyan Xiong, and Jamie Callan. CAsT 2019: The Conversational Assistance Track Overview. In *Proceedings of the 28th Text REtrieval Conference*, TREC '19, 2019.

[29] Jeffrey Dalton, Chenyan Xiong, and Jamie Callan. CAsT 2020: The Conversational Assistance Track Overview. In *Proceedings of the 29th Text REtrieval Conference*, TREC '20, 2020.

[30] Jeffrey Dalton, Chenyan Xiong, and Jamie Callan. CAsT 2021: The Conversational Assistance Track Overview. In *Proceedings of the 30th Text REtrieval Conference*, TREC '21, 2021.

[31] Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. KILT: a Benchmark for Knowledge Intensive Language Tasks, 2020.

[32] Jheng-Hong Yang, Sheng-Chieh Lin, Chuan-Ju Wang, Jimmy J. Lin, and Ming-Feng Tsai. Query and Answer Expansion from Conversation History. In *Proceedings of the 28th Text REtrieval Conference*, TREC '19, 2019.

[33] Ronak Pradeep, Xueguang Ma, Xinyu Zhang, Hang Cui, Ruizhou Xu, Rodrigo Nogueira, and Jimmy Lin. H2oloo at TREC 2020: When all you got is a hammer... Deep Learning, Health Misinformation, and Precision Medicine. In *Proceedings of the Twenty-Ninth Text REtrieval Conference, TREC 2020, Virtual Event [Gaithersburg, Maryland, USA], November 16-20, 2020*, volume 1266 of *NIST Special Publication*. National Institute of Standards and Technology (NIST), 2020.

[34] Ahmed Elgohary, Denis Peskov, and Jordan Boyd-Graber. Can You Unpack That? Learning to Rewrite Questions-in-Context. In *Empirical Methods in Natural Language Processing*, 2019.

[35] Chia-Yuan Chang, Ning Chen, Wei-Ting Chiang, Chih-Hen Lee, Yu-Hsuan Tseng, Chuan-Ju Wang, Hsien-Hao Chen, and Ming-Feng Tsai. Query Expansion with Semantic-Based Ellipsis Reduction for Conversational IR. In *Proceedings of the 29th Text REtrieval Conference*, TREC '20, 2020.

[36] Rishiraj Saha Roy and Avishek Anand. *Question Answering for the Curated Web: Tasks and Methods in QA over Knowledge Bases and Text Collections*. Morgan & Claypool, 2021.

[37] Daya Guo, Duyu Tang, Nan Duan, Ming Zhou, and Jian Yin. Dialog-to-Action: Conversational Question Answering over a Large-Scale Knowledge Base. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS '18, page 2946–2955, 2018.

[38] Siva Reddy, Danqi Chen, and Christopher D. Manning. CoQA: A Conversational Question Answering Challenge. *CoRR*, abs/1808.07042, 2018.

[39] Chen Qu, Liu Yang, Cen-Chieh Chen, Minghui Qiu, W. Bruce Croft, and Mohit Iyyer. Open-Retrieval Conversational Question Answering. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020.

[40] Pengjie Ren, Zhumin Chen, Zhaochun Ren, Evangelos Kanoulas, Christof Monz, and Maarten de Rijke. Conversations with Search Engines. *CoRR*, abs/2004.14162, 2020.

[41] Denny Vrandečić and Markus Krötzsch. Wikidata: A Free Collaborative Knowledgebase. *Commun. ACM*, 57(10):78—-85, 2014.

[42] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. DBpedia: A Nucleus for a Web of Open Data. In *Proceedings of the 6th International The Semantic Web and 2nd Asian Conference on Asian Semantic Web Conference*, ISWC '07/ASWC '07, pages 722–735, 2007.

[43] Kelvin Jiang, Dekun Wu, and Hui Jiang. FreebaseQA: A New Factoid QA Data Set Matching Trivia-Style Question-Answer Pairs with Freebase. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, NAACL '19, pages 318–323, 2019.

[44] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. In *Proceedings of the 2008 ACM Special Interest Group on Management of Data International Conference on Management of Data*, SIGMOD '08, pages 1247—-1250, 2008.

[45] Chen Qu, Liu Yang, Minghui Qiu, Yongfeng Zhang, Cen Chen, W. Bruce Croft, and Mohit Iyyer. Attentive History Selection for Conversational Question Answering. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM 2019)*, CIKM '19, 2019.

[46] Shi Yu, Zhenghao Liu, Chenyan Xiong, Tao Feng, and Zhiyuan Liu. Few-Shot Conversational Dense Retrieval. *CoRR*, abs/2105.04166, 2021.

[47] Jiao Liu, Yanling Li, and Min Lin. Review of Intent Detection Methods in the Human-Machine Dialogue System. *Journal of Physics: Conference Series*, 1267(1): 012059, 2019.

[48] John Dowding, Jean Mark Gawron, Doug Appelt, John Bear, Lynn Cherny, Robert Moore, and Douglas Moran. GEMINI: A Natural Language System for Spoken-Language Understanding. In *31st Annual Meeting of the Association for Computational Linguistics*, ACL '93, 1993.

[49] Tanja Bunk, Daksh Varshneya, Vladimir Vlasov, and Alan Nichol. DIET: Lightweight Language Understanding for Dialogue Systems, 2020.

[50] Mady Mantha. Introducing DIET: state-of-the-art architecture that outperforms fine-tuning BERT and is 6X faster to train, 2020. URL `https://rasa.com/blog/introducing-dual-intent-and-entity-transformer-diet-state-of-the-art-performance-on-a-l` Accessed: 2022-04-22.

[51] Leif Azzopardi, Mateusz Dubiel, Martin Halvey, and Jeffrey Dalton. Conceptualizing agent-human interactions during the conversational search process. In *The Second International Workshop on Conversational Approaches to Information Retrieval*, 2018.

[52] Svitlana Vakulenko, Claudio Revoredo, Kate nd Di Ciccio, and Maarten de Rijke. QRFA: A Data-Driven Model of Information-Seeking Dialogues. In *Proceedings of the 43rd European Conference in Information Retrieval*, ECIR '19, pages 541–557. Springer International Publishing, 2019.

[53] Norha M. Villegas, Cristian Sánchez, Javier Díaz-Cely, and Gabriel Tamura. Characterizing context-aware recommender systems: A systematic literature review. *Knowledge-Based Systems*, 140:173–200, 2018.

[54] Umberto Panniello and Michele Gorgoglione. Incorporating context into recommender systems: an empirical comparison of context-based approaches. *Electronic Commerce Research*, 12:1–30, 2012.

[55] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[56] Jie Lu, Dianshuang Wu, Mingsong Mao, Wei Wang, and Guangquan Zhang. Recommender system application developments: A survey. *Decision Support Systems*, 74:12–32, 2015.

[57] Hao Ma, Tom Chao Zhou, Michael R Lyu, and Irwin King. Improving recommender systems by incorporating social contextual information. *ACM Transactions on Information Systems (TOIS)*, 29(2):1–23, 2011.

[58] Gediminas Adomavicius and Alexander Tuzhilin. *Context-Aware Recommender Systems*. Springer US, 2011.

[59] Sahar Ebrahimi, Norha Villegas, Hausi Müller, and Alex Thomo. SmarterDeals: A Context-aware Deal Recommendation System based on the SmarterContext Engine. In *Proceedings of the 2012 Conference of the Center for Advanced Studies on Collaborative Research*, CASCON '12, 2012.

[60] Gregory D. Abowd, Anind K. Dey, Peter J. Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a Better Understanding of Context and Context-Awareness, booktitle=Handheld and Ubiquitous Computing. HUC '99, pages 304–307. Springer Berlin Heidelberg, 1999.

[61] S. Arun Nair, Amit Anil Nanavati, and Nitendra Rajput. Conspeakuous : Contextualising Conversational Systems. In *Human-Computer Interaction. HCI Intelligent Multimodal Interaction Environments*, HCI '07. Springer Berlin Heidelberg, 2007.

[62] Norha M. Villegas and Hausi A. Müller. *Managing Dynamic Context to Optimize Smart Interactions and Services*. Springer Berlin Heidelberg, 2010.

[63] Milica Gasic, Catherine Breslin, Matthew Henderson, Dongho Kim, Martin Szummer, Blaise Thomson, Pirros Tsiakoulis, and Steve J. Young. POMDP-based dialogue manager adaptation to extended domains. In *Proceedings of the Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, SIGDIAL '13, 2013.

[64] Pearl Pu, Maoan Zhou, and Sylvain Castagnos. Critiquing Recommenders for Public Taste Products. In *Proceedings of the Third ACM Conference on Recommender Systems*, RecSys '09, page 249–252. Association for Computing Machinery, 2009.

[65] Peter Grasch, Alexander Felfernig, and Florian Reinfrank. ReComment: Towards Critiquing-Based Recommendation with Speech Interaction. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, page 157–164. Association for Computing Machinery, 2013.

[66] Yuichiro Ikemoto, Varit Asawavetvutt, Kazuhiro Kuwabara, and Hung-Hsuan Huang. Tuning a conversation strategy for interactive recommendations in a chatbot setting. *Journal of Information and Telecommunication*, 3:1–16, 11 2018.

[67] Yucheng Jin, Wanling Cai, Li Chen, Nyi Nyi Htun, and Katrien Verbert. MusicBot: Evaluating Critiquing-Based Music Recommenders with Conversational Interaction. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, page 951–960. Association for Computing Machinery, 2019.

[68] Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen-tau Yih, and Michel Galley. A Knowledge-Grounded Neural Conversation Model. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018.

[69] Lizi Liao, Ryuichi Takanobu, Yunshan Ma, Xun Yang, Minlie Huang, and Tat-Seng Chua. Deep Conversational Recommender in Travel. *CoRR*, abs/1907.00710, 2019.

[70] Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. How NOT To Evaluate Your Dialogue System: An Empirical Study of Unsupervised Evaluation Metrics for Dialogue Response Generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, EMNLP '16', pages 2122–2132. Association for Computational Linguistics, 2016.

[71] Jan Deriu, Alvaro Rodrigo, Arantxa Otegi, Guillermo Echegoyen, Sophie Rosset, Eneko Agirre, and Mark Cieliebak. Survey on evaluation methods for dialogue systems. *Artificial Intelligence Review*, 54:755–810, 2021.

[72] Chin-Yew Lin. ROUGE: Ebrahimis. In *Workshop on Text Summarization Branches Out, Post-Conference Workshop of ACL 2004, Barcelona, Spain*, ACL '04, pages 74–81, 2004.

[73] Satanjeev Banerjee and Alon Lavie. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, ACL '05, pages 65–72. Association for Computational Linguistics, 2005.

[74] Ryan Lowe, Michael Noseworthy, Iulian Vlad Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. Towards an Automatic Turing Test: Learning to Evaluate Dialogue Responses. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1116–1126. Association for Computational Linguistics, 2017.

[75] Chongyang Tao, Lili Mou, Dongyan Zhao, and Rui Yan. RUBER: An Unsupervised Method for Automatic Evaluation of Open-Domain Dialog Systems. *CoRR*, abs/1701.03079, 2017.

[76] Ananya B. Sai, Mithun Das Gupta, Mitesh M. Khapra, and Mukundhan Srinivasan. Re-Evaluating ADEM: A Deeper Look at Scoring Dialogue Responses. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):6220–6227, 2019.

[77] Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. Evaluating the State-of-the-Art of End-to-End Natural Language Generation: The E2E NLG Challenge. 2019.

[78] Fenfei Guo, Angeliki Metallinou, Chandra Khatri, Anirudh Raju, Anu Venkatesh, and Ashwin Ram. Topic-based Evaluation for Conversational Bots. *CoRR*, abs/1801.03622, 2018.

[79] Mohammad Aliannejadi, Hamed Zamani, Fabio Crestani, and W. Bruce Croft. Asking Clarifying Questions in Open-Domain Information-Seeking Conversations. ACM, 2019.

[80] Krisztian Balog and Filip Radlinski. *Measuring Recommendation Explanation Quality: The Conflicting Goals of Explanations*, page 329–338. Association for Computing Machinery, 2020.

[81] Ashwin Ram, Rohit Prasad, Chandra Khatri, Anu Venkatesh, Raefer Gabriel, Qing Liu, Jeff Nunn, Behnam Hedayatnia, Ming Cheng, Ashish Nagar, Eric King, Kate Bland, Amanda Wartick, Yi Pan, Han Song, Sk Jayadevan, Gene Hwang, and Art Pettigrue. Conversational AI: The Science Behind the Alexa Prize. 2018.

[82] Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young. Agenda-Based User Simulation for Bootstrapping a POMDP Dialogue System. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, NAACL '07, 2007.

[83] Layla El Asri, Jing He, and Kaheer Suleman. A Sequence-to-Sequence Model for User Simulation in Spoken Dialogue Systems. *CoRR*, abs/1607.00070, 2016.

[84] Paul Crook and Alex Marin. Sequence to Sequence Modeling for User Simulation in Dialog Systems. pages 1706–1710, 08 2017. doi: 10.21437/Interspeech.2017-161.

[85] Wikipedia. Markov decision process, 2022. URL `https://en.wikipedia.org/wiki/Markov_decision_process`. Accessed: 2022-04-15.

[86] Alexandre Salle, Shervin Malmasi, Oleg Rokhlenko, and Eugene Agichtein. Studying the Effectiveness of Conversational Search Refinement Through User Simulation. In *Proceedings of the 43rd European Conference in Information Retrieval*, ECIR '21', pages 587–602. Springer International Publishing, 2021.

[87] Weiwei Sun, Shuo Zhang, Krisztian Balog, Zhaochun Ren, Pengjie Ren, Zhumin Chen, and Maarten de Rijke. Simulating User Satisfaction for the Evaluation of Task-Oriented Dialogue Systems. In *Proceedings of the 44th International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '21, pages 2499–2506. Association for Computing Machinery, 2021.

[88] Shengnan Lyu, Arpit Rana, Scott Sanner, and Mohamed Reda Bouadjenek. A Workflow Analysis of Context-Driven Conversational Recommendation. In *Proceedings of the Web Conference 2021*, WWW '21, page 866–877. Association for Computing Machinery, 2021.

[89] Wanling Cai and Li Chen. Predicting User Intents and Satisfaction with Dialogue-based Conversational Recommendations. In *Proceedings of the 28th ACM Conference on User Modeling, Adaptation and Personalization*, UMAP '20, pages 33–42, 07 2020.