User Memory Reasoning for Conversational Recommendation

Hu Xu[1,2*], **Seungwhan Moon**[1], **Honglei Liu**[1], **Bing Liu**[1], **Pararth Shah**[1]

Bing Liu[2] and **Philip S. Yu**[2]

[1]Facebook

[2]Department of Computer Science, University of Illinois at Chicago

{huxu,shanemoon,honglei,bingl,pararths}@fb.com

{liub,psyu}@uic.edu

Abstract

We study an end-to-end approach for conversational recommendation that dynamically manages and reasons over users' past (offline) preferences and current (online) requests through a structured and cumulative *user memory knowledge graph*. This formulation extends existing state tracking beyond the boundary of a single dialog to user state tracking (UST). For this study, we create a new **M**emory **G**raph (MG) $\leftrightarrow$ **Conv**ersational **Rec**ommendation parallel corpus called *MGConvRex* with 7K+ human-to-human role-playing dialogs, grounded on a large-scale user memory bootstrapped from real-world user scenarios. MGConvRex captures human-level reasoning over user memory and has disjoint training/testing sets of users for zero-shot (cold-start) reasoning for recommendation. We propose a simple yet expandable formulation for constructing and updating the MG, and an end-to-end graph-based reasoning model that updates MG from unstructured utterances and predicts optimal dialog policies (*e.g.* recommendation) based on updated MG. The prediction of our proposed model inherits the graph structure, providing a natural way to explain policies. Experiments are conducted for both offline metrics and online simulation, showing competitive results.[1]

1 Introduction

Conversational recommendation system has recently gained traction in the dialog community, in which the model aims to learn up-to-date (online) user preferences, instead of using static (offline) preferences as in the traditional, *e.g.* collaborative filtering (CF). Most existing works focus on combining a static recommender system with a dialog system, by updating user preferences via asking relevant questions (often referred to as "System Ask User Respond (SAUR)" (Zhang et al., 2018). However, this "short-term" update in the model unnaturally isolates users' history and their preference in the current dialog (often forgotten after the dialog is finished). An intelligent system should be able to dynamically maintain and reason over users' knowledge for current (and possibly future) recommendations.

To this end, we introduce a novel concept called *user memory graph* to holistically represent the knowledge about users and associated items. This user memory graph extends the notion of dialog state tracking (DST) beyond the boundary of a single dialog to *user state tracking* (UST). As in Figure 1, such a graph contains static knowledge obtained offline (*e.g.* items, attributes, the history of users and past dialogs) and users' knowledge online (*e.g.* from the current and possibly future dialogs). User memory graph naturally has the following benefits. (1) *Holistic reasoning* considers available knowledge about users and items all together (not just dialog states) to generate policies. We focus on two types of reasoning policies: *graph update policy* and *dialog policy*, where the former updates users' online knowledge and serves as the basis for the later to generate outputs of the agent's current turn. We believe reasoning is the core problem in conversational recommendation because asking a good question or finding a good candidate item needs to explore the "soft match" of the knowledge between users and

[*] Most work was done while the first author was a research intern at Facebook.

[1] The dataset, code and models will be released for future research.

Proceedings of the 28th International Conference on Computational Linguistics, pages 5288–5308
Barcelona, Spain (Online), December 8-13, 2020

items[2](Zhang et al., 2018). (2) *Zero-shot (cold-start) reasoning* for users/items unseen during training: User memory graph naturally separates user/item knowledge from the reasoning process of policy. As such, one can train a user/item agnostic model that can be later applied to another user memory graph (after the model is deployed). In contrast, most CF-based system "overfits" to existing users / items (in their embeddings). (3) *Open space policy* is a key challenge in conversational recommendation because of the innumerable items involved in dialog policy. This requires a flexible space of policy to cover all items (and possibly all valid values and slots [3]) instead of a pre-defined space and user memory graph can serve for that purpose.

User Memory Reasoning for Recommendation: Assuming an agent involves in a conversation with a user e_u. The agent (1) constructs [4] a user memory graph $\mathcal{G}_0 = \{(e, r, e') | e, e' \in \mathcal{E}, r \in \mathcal{R}\}$ based on history knowledge $\mathcal{H}$ of e_u, candidate items $\mathcal{C}$, and their associated slots and values; (2) without loss of generality, updates $\mathcal{G}_{x-1}$ with new knowledge from the x-th turn D_x, in the form of tuples $\mathcal{G}_x \leftarrow \mathcal{G}_{x-1} \cup \{(e, r, e'), \dots\}$; (3) performs reasoning over $\mathcal{G}_x$ to yield a dialog policy π_x that either (i) performs more rounds of interaction to collect users' knowledge, or (ii) recommends items $\mathcal{T} \subset \mathcal{C}$ to the user.

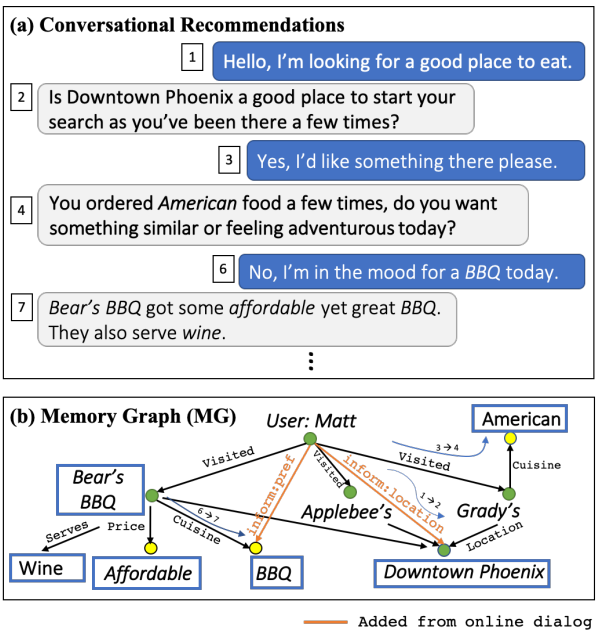


Figure 1: **User memory reasoning for conversational recommendation**. (1) Past (offline) user preferences are captured as an initial Memory Graph (MG). (2) Conversational recommendation allows users to express preferences and requirements through dialogs. (3) Our *MGConvRex* corpus is grounded on user memory, representing user's past history as well as newly added preferences.

In this paper, we first collect a dataset as existing public datasets hardly meet the needs for the following reasons. (1) Lacking users' history and thus dialogs referring to history (*e.g.* the 2nd and 4th turn in Figure 1): One reason is that most datasets aim for task-oriented systems, where users' history and reasoning are not core issues to solve. (2) Lacking fine-grained annotation (for updating the user memory graph): Most public datasets for conversational recommendation are combinations of the datasets for recommender systems and dialogs transcribed separately (Li et al., 2018a; Zhang et al., 2018). The process is not designed for knowledge-grounded dialogs and leads to the hardness of annotating entity-level knowledge. (3) Lacking human-level reasoning: The transcribed policies in existing datasets do not reflect deep reasoning over knowledge (*e.g.* asking a good question). Some actions are taken at the transcribers' will (Li et al., 2018a). Our collected dataset is called **M**emory **G**raph ↔ **Conv**ersational **Rec**ommendation (MGConvRex), which contains 7.6K+ dialogs with 73K turns with fine-grained annotation to meet the above needs.

We first define a simple yet flexible ontology for user memory graph, as detailed in Section 4. Based on that, we propose a model called end-to-end user memory graph reasoner (E2E-UMGR) that performs two types of reasoning: updating graph and generating dialog policies. We cast the former reasoning as a graph generation problem (Li et al., 2018c), which infers new knowledge from unstructured utterances into user memory graph. The later reasoning deals with the challenge of *open space policy* by preserving the structure of the graph and generating policy.

In summary, the **contributions** of this paper are as following: (1) we propose a novel task of user memory reasoning for conversational recommendation; (2) we collect a dataset and propose an ontology

[2]In contrast, task-oriented dialog mostly focuses on hard constraints matching (*e.g.* DB query) on available records.
[3]We reuse the terms from task-oriented dialog, although slots and values are just entities in a user knowledge graph.
[4]The construction procedure for user memory graph is omitted here for brevity, and detailed in Section 4.

to construct user memory graph; (3) we propose an end-to-end baseline for reasoning updates and dialog policy over the user memory graph. Experimental results show that such a reasoning model is promising.

2 Related Work

Conversational Recommendation is one important type of information seeking dialog system (Zhang et al., 2018). Existing studies focus on combining a recommender system with a dialog state tracking system, through the "System Ask User Respond (SAUR)" paradigm. Once enough user preference is collected, such systems often make personalized recommendations to the user. For instance, (Li et al., 2018a) propose to mitigate cold-start users by learning new users' preferences during conversations and linking the learned preferences to existing similar users in a traditional recommender system. (Sun and Zhang, 2018) propose to updates a recommender system in the latent space with the latent space of task-oriented dialog state tracking and tune the dialog policy via reinforcement learning. (Kang et al., 2019) propose a self-play reinforcement learning (RL) setting to boost the performance of a text-to-text dialog model. (Zhang et al., 2018) leverage reviews to mimic online conversations to update an existing user's preference and re-rank items. In (Misu et al., 2010), the user memory/knowledge is represented as a probabilistic state within a fixed hierarchical Markov probabilistic model, which lacks the flexibility of encoding fine-grained knowledge and accumulating new knowledge. (Zhou et al., 2020) demonstrate the usage of user profile and users' interests from ongoing dialog in a social chatbot. However, none of the existing systems (or datasets) aims to build an explicit user memory for reasoning and long-term use.

Task-oriented Dialog Systems are widely studied with multiple popular benchmark datasets (Henderson et al., 2014; Wen et al., 2016; Budzianowski et al., 2018; Eric et al., 2019; Rastogi et al., 2019). Most of the state-of-the-art approaches (Wu et al., 2019; Gao et al., 2019; Chao and Lane, 2019) focus on improving dialog state tracking (DST) with span-based pointer networks for unseen values, which predicts information that is essential for completing a specified task (*e.g.* hotel/air ticket booking, etc.). Note that most research refers to a state in the context of a single dialog. We deal with a user state beyond a single dialog represented by user memory graph.

Datasets for task-oriented systems typically lack users' history, probably because users' history is not very important to correctly locate a record for the current dialog. Although certain types of dialog act, slots, and values are shareable for both task-oriented system and conversational recommendation, the core problem of conversational recommendation is to reason and to rank items or questions to ask.

Graph Reasoning is essential for updating graphs and generating dialog policy from the proposed user memory graph. This paper casts the graph updates as a graph generation problem (Li et al., 2018c) from both structured graph and unstructured utterances. Although our graph generation problem is closer to the link prediction problem (Lü and Zhou, 2011; Martínez et al., 2016) in a social network, the key difference is that our problem focuses on progressively incorporating unstructured utterances into graph instead of inferring missing relations. There are many studies on leveraging knowledge graphs for recommender systems. For example, (Xian et al., 2019) introduced a graph-based recommender (not dialog) system that is trained via reinforcement learning. Graph neural networks are popular in recent years, which aim to learn hidden representations over discrete graph structures(Scarselli et al., 2008; Duvenaud et al., 2015; Defferrard et al., 2016; Kipf and Welling, 2016). Several extensions to the original graph neural network have been proposed (Li et al., 2015; Pham et al., 2017), most notably R-GCNs (Schlichtkrull et al., 2018), which can be applied to large-scale and multi-relational graphs. A few works have recently been proposed to allow knowledge graph reasoning in dialog systems. (Moon et al., 2019a; Moon et al., 2019b) propose a new corpus to learn knowledge graph paths that connect dialog turns. (Tuan et al., 2019) introduce a knowledge-grounded dialog generation task given a knowledge graph that is dynamically updated. However, these works often focus on response generation and do not address the reasoning of user knowledge in conversational recommendations.

3 MGConvRex

MGConvRex aims to contain dialogs that draw the relevance of the user's history and fine-grained user preferences to update the user memory graph. As such, we propose to leverage existing data from

Dialog Act a	Description	Examples
User-side		
Greeting	Greeting to the agent	I'd like to find a place to eat.
Inform	Actively inform the agent your preference	I'd like to find a *thai* restaurant .
Answer	Answer to a question from the agent	I prefer *thai* food.
Reply	Reply to a recommendation	I'll give it a try.
Open question (OQ)	Actively ask an open question about a recommended item.	What kind of food do they serve ?
Yes/no question (YNQ)	Actively ask an yes/no question about a recommended item.	Do they serve *thai* food ?
Thanks	Thanks the agent	Thanks for your help.
Agent-side		
Greeting	Greeting to the user.	How may I help you today ?
Open question (OQ)	Ask an open question about a slot to the user	What kind of food do you prefer ?
Yes/no question (YNQ)	Ask a yes/no question about a value of a slot	I saw you've been to *thai* restaurant, do you still like that ?
Recommendation (REC)	Recommend items to the user.	How about *burger king*, which serves *fast food* ?
Answer (ANS)	Answers user's questions on an item.	They serve *thai* food.
Thanks	Thanks the user	Enjoy your meal.

Table 1: Dialog acts for agent and user $\mathcal{A}$: the spans of items/slot values are italized.

Dataset	All Dialogs			Dialogs w/ History		Dialogs w/o History	
	# of Dial.	# of Turns	Avg. # of Turns	# of Dial.	Avg. # of Turns	# of Dial.	Avg. # of Turns
Train	4985	48457	9.72	2418	9.62	2567	9.81
Dev	263	2466	9.38	121	9.16	142	9.56
Test	2367	23048	9.74	1160	9.62	1207	9.85

Table 2: **Statistics**: Dialogs w/ or w/o History indicates whether scenarios include visited items $\mathcal{H}$.

recommender systems that carry users' past behavior to harvest large-scale dialog scenarios. Then we transcribe and annotate dialogs based on pre-define fine-grained dialog acts (see Table 1), slots, values, and sentiment polarities. The statistics of MGConvRex are in Table 7, with non-overlapping splits of users for training/dev/testing. We notice that the average number of turns for scenarios with users' history are slightly shorter than those without users' history (details of MGConvRex are in appendix).

3.1 Dialog Scenarios

A *scenario* refers to a pre-defined user-agent setting to collect a dialog between two crowd workers, where one plays the user and the other plays the agent. We use yelp datasets (`https://www.yelp.com/dataset`) to generate scenarios. This mitigates the needs of curating synthetic dialog scenarios as in datasets for task-oriented dialog system(Li et al., 2016; Li et al., 2018b).

We assume each item is associated with values and each value is associated with at least one slot. Let $\mathbb{B} = \{0, 1\}$ be a binary number. We define a scenario consisting of the following parts: $(e_u, C, H, V, P, \mathcal{T})$, where e_u is a user, $C \in \mathbb{B}^{|\mathcal{C}|\times|\mathcal{V}|}$ is about the candidate items $\mathcal{C}$ and their associated values $\mathcal{V}$, $H \in \mathbb{B}^{|\mathcal{H}|\times|\mathcal{V}|}$ is about users past history (e_u visited items $\mathcal{H}$[5] and their values) that is known to the agent, $V \in \mathbb{B}^{|\mathcal{V}|\times|\mathcal{S}|}$ indicates values with their associated slots, $P \in \mathbb{B}^{|\mathcal{S}|\times|\mathcal{V}|}$ is the user preference (which value the user prefer for a slot) and $\mathcal{T} \subset \mathcal{C}$ is the ground-truth items.

We create dialog scenarios in the following way: (1) for each user, we draw $|\mathcal{H}| \in [5, 20]$ visited items and $|\mathcal{T}| = 1$ [6] items as the *ground-truth items* $\mathcal{T}$. Use the values and its associated slots of the ground-truth items as *user preference* P. (2) negatively sample $|\mathcal{C}| - |\mathcal{T}|$ items and combine them with the ground-truth items $\mathcal{T}$ as *candidate items* $\mathcal{C}$.

To ensure difficulty of human reasoning, we choose $|\mathcal{C}| \in [10, 20]$ candidate items and enforce certain similarity over candidate items (such as all locations are from the same state) as the ground-truth items. For the same user, we also create a duplicated scenario except that $|\mathcal{H}| = 0$, where the agent player can only use knowledge from the current dialog for recommendation.

3.2 Dialog Acts, Slots, Values and Sentiment Polarities

Dialog Acts ($\mathcal{A}$): Table 1 shows the dialog acts for both the user and the agent. Besides the System Ask User Respond (SAUR) paradigm (Sun and Zhang, 2018; Li et al., 2018a; Zhang et al., 2018), we also propose a User Ask - System Respond (UASR) paradigm that allows users to actively participate in a recommendation. Acts such as *Open question*, *Yes/no question* and *Inform* are designed for this purpose.

[5]To reduce the load of transcribers, a user's past history only contains visited items at this stage.

[6]We use 1 ground-truth item to reduce the load of the transcribers and increase the difficulty of reasoning.

Slots and Values($\mathcal{S}$, $\mathcal{V}$): We select $|\mathcal{S}| = 10$ popular slots with a total of 470+ values for the restaurant domain. To help transcribers use some values naturally in utterances, we change some values (such as price ranges $) into English words ("cheap" etc.).

Sentiment Polarity: We define a user's preference expressed in a conversation as pairs of opinion targets (an item or a value) and their associated sentiment polarities(Hu and Liu, 2004). We adopt 3 types of polarities *pos_on*, *neg_on* and *neu_on* to represent positive, negative and neutral polarity, respectively [7].

4 User Memory Graph

We describe the formulation of a user memory graph. There are many design choices for constructing a user memory graph. Our goal is to model user/item knowledge with extensibility and maintenance.

Slot e_s	**Example Value** e_v
location	Las Vegas, NV; Toronto, ON
category	fast food; burger; thai
price	cheap; expensive
parking	garage; valet; lot
noise	average; quiet
ambience	classy; intimate
alcohol	full bar; beer and wine
good for meal	brunch; lunch; dinner
wifi	paid; free
attire	casual; formal

Table 3: Available slots $\mathcal{S}$ and example of their associated values $\mathcal{V}$.

Entity Types $\mathcal{E}$	**Explanation**
$\mathcal{U}$	user entities
$\mathcal{M}$	memory entities
$\mathcal{I}$	item entities (candidates and visited): $\mathcal{C} \cup \mathcal{H}$
$\mathcal{S}$	slot entities
$\mathcal{V}$	value entities
Relation Types $\mathcal{R}$	
$(\mathcal{U}, \text{has\_memory}, \mathcal{M})$	a user u has a memory entity m
$(\mathcal{M}, \text{visited}, \mathcal{I})$	a memory m is about an item i
$(\mathcal{I}, \text{has\_aspect}, \mathcal{V})$	an item i has a value v
$(\mathcal{V}, \text{is\_a}, \mathcal{S})$	a value v belongs to a slot s
$(\mathcal{M}, \textbf{pos\_on}, \mathcal{V}/\mathcal{I})$	m is positive on a value or item
$(\mathcal{M}, \textbf{neg\_on}, \mathcal{V}/\mathcal{I})$	m is negative on a value or item
$(\mathcal{M}, \textbf{neu\_on}, \mathcal{V}/\mathcal{I})$	m is neutral on a value or item

Table 4: Ontology of graph: bolded relations are used for graph updates or accumulation.

4.1 Construction

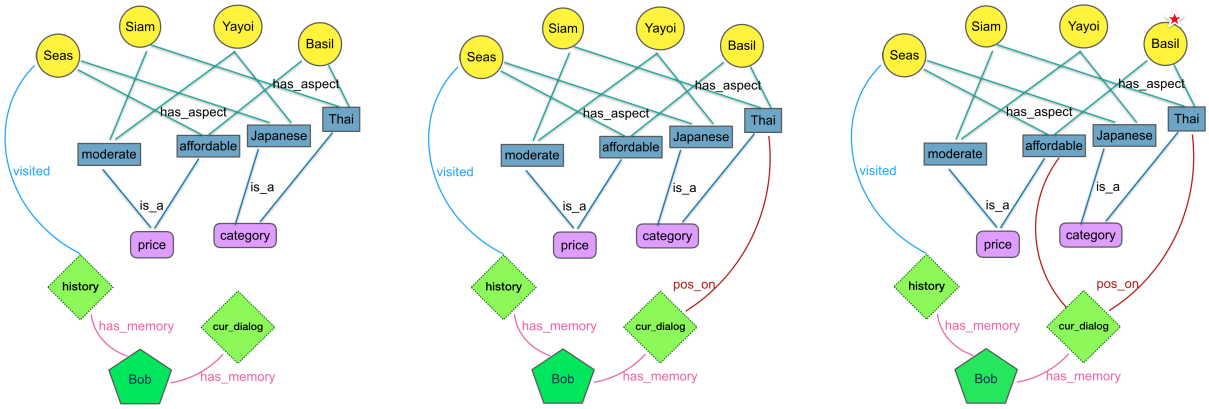
A user memory graph is denoted as $\mathcal{G} = \{(e, r, e') | e, e' \in \mathcal{E}, r \in \mathcal{R}\}$, which is essentially a heterogeneous graph with typed entities and relations. We first define the ontology (or meta entities and relations) in Table 4. The user memory contains available items $\mathcal{I}$ for a dialog scenario. An item i can be associated with multiple values vs with $r_{\text{has\_aspect}}$ relation. Each value is associated with their slot s via $r_{\text{is\_a}}$ relation. In this way, values / slots entities are rather expandable and new values or slots (or even slots of slots) can be easily added in. Further, each user has their entity e_u and several associated memory entities ms. We define memory entity to model an event or experience of a user, such as visiting a restaurant (via entity m_{history}), or having a conversation as in current dialog (via $m_{\text{cur\_dialog}}$). The advantage of allowing multiple memory entities is that a user may have different opinions for the same target (items or values) from their very different experiences (*e.g.* like *Thai* food for lunch but not dinner). To express a user's history on visited items, we use a r_{visited} relation to connect a memory entity with a visited item. As an example, we demonstrate the construction of a user u_{Bob} in the first graph in Figure 2.

4.2 Update

The updates of user memory graph is supposed to leverage the outputs of text encoder (or state tracking) and the previous graph. For simplicity, we use 3 sentiment relations $r_{\text{pos\_on}}$, $r_{\text{neg\_on}}$ and $r_{\text{neu\_on}}$ to update a user memory graph, which associate values/items (opinion target) with the memory entity of the current dialog $m_{\text{cur\_dialog}}$. We believe humans have a more complex memory system and expect future work on modeling user memory (such as error handling).

Initially, we know that u_{Bob} likes *Thai* food and the user memory graph is updated with a new triple $(m_{\text{cur\_dialog}}, r_{\text{pos\_on}}, v_{\text{Thai}})$. Following the second turn of the user, we know that u_{Bob} is still interested in $v_{\text{affordable}}$ restaurants, indicated by a new triple $(m_{\text{cur\_dialog}}, r_{\text{pos\_on}}, v_{\text{affordable}})$. Then the agent can infer a recommendation i_{Basil}, which can be explained by paths: (1) $u_{\text{Bob}} \rightarrow r_{\text{has\_memory}} \rightarrow m_{\text{cur\_dialog}} \rightarrow r_{\text{pos\_on}} \rightarrow v_{\text{Thai}} \rightarrow r_{\text{has\_aspect}} \rightarrow i_{\text{Basil}}$, (2) $u_{\text{Bob}} \rightarrow r_{\text{has\_memory}} \rightarrow m_{\text{cur\_dialog}} \rightarrow r_{\text{pos\_on}} \rightarrow v_{\text{affordable}} \rightarrow r_{\text{has\_aspect}} \rightarrow i_{\text{Basil}}$, and (3) $u_{\text{Bob}} \rightarrow r_{\text{has\_memory}} \rightarrow m_{\text{history}} \rightarrow r_{\text{visited}} \rightarrow v_{\text{Seas}} \rightarrow r_{\text{has\_aspect}} \rightarrow v_{\text{affordable}} \rightarrow$

[7] We do not deal with emotions (*e.g. sad*), although existing works may use sentiment to indicate emotions.



(a) $\mathcal{G}_0$ (b) $\mathcal{G}_1$ (c) $\mathcal{G}_2$

Figure 2: User memory graph construction ($\mathcal{G}_0$) and updates based on the dialog: Agent: what kinds of food do you like ? User: I like **Thai** food. ($\mathcal{G}_1$) Agent: are you *still* interested in **affordable** restaurant ? User: *yes*. ($\mathcal{G}_2$) Agent: how about **Basil**, which is affordable and serves Thai food.

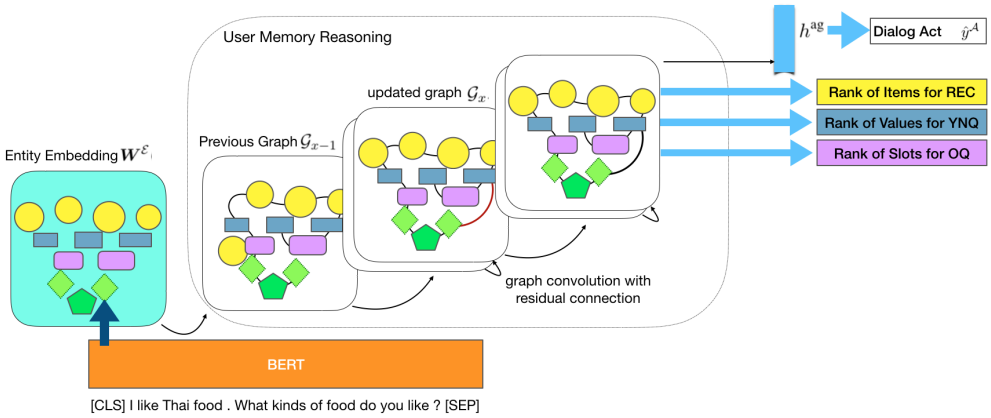


Figure 3: E2E-UMGR: the red edge indicates the generated sentiment relation in the updated graph $\mathcal{G}_x$.

$r_{\text{has\_aspect}} \rightarrow i_{\text{Basil}}$, where the last path draws the relevance from a visited item to the current recommendation. As we can see, sentiment relations serve as the bridge to connect a user to items and enables knowledge for reasoning dialog policy.

5 User Memory Graph Reasoner

We propose a baseline model called end-to-end User Memory Graph Reasoner (E2E-UMGR), which uses previous user memory graph to reason both graph updates (sentiment relations) and dialog policy (see Figure 3). E2E-UMGR simplifies the process of turning unstructured data into structured form and reasons dialog policy directly over graph to resolve the issue of open space policy.

Input: (1) past utterances up to the current turn $D_{1:x}$; (2) previous user memory graph $\mathcal{G}_{x-1}$.

Output: dialog policy $\pi = (\hat{y}^{\mathcal{A}}, \hat{y}^{\mathcal{C}}, \hat{y}^{\mathcal{S}}, \hat{y}^{\mathcal{V}})$ for the current turn, where $\mathcal{A}$, $\mathcal{C}$, $\mathcal{S}$, $\mathcal{V}$ indicate the space of dialog acts, candidate items, slots and values, respectively.

Note that $\hat{y}^{\mathcal{C}}$, $\hat{y}^{\mathcal{S}}$, and $\hat{y}^{\mathcal{V}}$ can be interpreted as the arguments of dialog acts and are essentially rankings over their corresponding entity sets. For example, when $\hat{y}^{\mathcal{A}} =$ *Recommendation*, the top-1 entity $\arg\max_{e_i \in \mathcal{C}}(\hat{y}^{\mathcal{C}})$ will be provided to the user. Similarly, $\hat{y}^{\mathcal{A}} =$ *Open Question* is related to the top-1 slot $\arg\max_{e_s \in \mathcal{S}}(\hat{y}^{\mathcal{S}})$ and $\hat{y}^{\mathcal{A}} =$ *Yes/no Question* is related to the top-1 value $\arg\max_{e_v \in \mathcal{V}}(\hat{y}^{\mathcal{V}})$. As such, the policy space of E2E-UMGR can be determined by the user memory graph where only valid entities can be generated. A structure-preserving model is preferred for reasoning where all entities in policy are generated as a holistic reasoning process.

We first encode all entities $e_j \in \mathcal{E}$ into hidden representations $h^{(0)}$ via an embedding layer and un-

structured utterances via a BERT(Devlin et al., 2019; Xu et al., 2019) encoder:

$$h_j^{(0)} = \boldsymbol{W}^{\mathcal{E}}(e_j), h_{m_{\text{cur\_dialog}}}^{(0)} = \text{BERT}(D_{1:x})_{\texttt{[CLS]}}, \tag{1}$$

where $\boldsymbol{W}^{\mathcal{E}}$ are embedding layers. `[CLS]` indicates the first token of BERT is assigned to the memory entity about the current dialog $m_{\text{cur\_dialog}}$. We believe $m_{\text{cur\_dialog}}$ is the best entity to store the hidden states of the current dialog because sentiment relations are closely associated with this entity. Then we incorporate a Relational Graph Convolutional Networks (R-GCN) (Schlichtkrull et al., 2018) into E2E-UMGR for reasoning both graph updates and dialog policies, where we stack two stages of R-GCNs for two types of reasoning respectively. R-GCN is a GCN (Kipf and Welling, 2016) with typed relations, where each relation is associated with their own weights to enable reasoning over a heterogeneous graph. Each entity is encoded by multiple layers of R-GCN as following:

$$h_j^{'(l+1)} = \text{GELU}\Big(\sum_{r\in\mathcal{R}}\sum_{k\in\mathcal{N}_j^r}\frac{1}{|\mathcal{N}_j^r|}\boldsymbol{W}_r^{(l)}h_k^{(l)}\Big), \tag{2}$$

where $h_j^{(l)}$ is the hidden state of entity e_j in the l-th layer of R-GCN, $\mathcal{N}_j^r$ is entity e_j's neighbors in relation type r and $\boldsymbol{W}_r^{(l)}$ is the weight associated with r in the l-th layer to transform one neighbor $h_k^{(l)}$. Specifically, let $\mathcal{N}_j^{r,\mathcal{G}_{x-1}}$ denote the graph connectivity of the previous user memory graph $\mathcal{G}_{x-1}$ and $\mathcal{N}_j^{r,\mathcal{G}_x}$ denote the updated graph. The R-GCN layer updates the hidden states of each entity with the incoming messages in the form of their neighbors' hidden states type-by-type. Then R-GCN sums over all types before passing through the GELU activation (Hendrycks and Gimpel, 2016). The hidden state of entity e_j in the $(l+1)$-th layer is computed via a residual connection (He et al., 2016) (to keep the original entity information instead of just neighbors' information) and layer normalization.

$$h_j^{(l+1)} = \text{LayerNorm}\Big(h_j^{(l)} + h_j^{'(l+1)}\Big). \tag{3}$$

We apply 2 R-GCN layers on $\mathcal{G}_{x-1}$ (via $\mathcal{N}_j^{r,\mathcal{G}_{x-1}}$) for updating graph and stack 5 R-GCN layers using $\mathcal{G}_x$ (via $\mathcal{N}_j^{r,\mathcal{G}_x}$) for reasoning dialog policies.

To predict updates, we first transform the hidden states of the memory, value and item entities from the second (2) layer: $h_{m_{\text{cur\_dialog}}}^z = \text{Dense}_{m_{\text{cur\_dialog}}}(h_{m_{\text{cur\_dialog}}}^{(2)})$, $h_v^z = \text{Dense}_v(h_v^{(2)})$ and $h_i^z = \text{Dense}_i(h_i^{(2)})$. We use z to indicate a variable is related to graph updates. Then we compute the probability of having a sentiment relation. Using values as an example, the probability and loss is computed as:

$$\hat{y}_{v,r}^z = \text{Sigmoid}\Big(\sum h_{m_{\text{cur\_dialog}}}^z \circ \boldsymbol{W}_r^z \circ h_v^z\Big), \mathcal{L}^{\mathcal{V},z} = \text{LogLoss}(\hat{y}^{\mathcal{V},\mathcal{R}',z}, y^{\mathcal{V},\mathcal{R}',z}), \tag{4}$$

where $\circ$ indicates element-wise (hadamard) products and $\sum$ reduces the hidden size to 1. We let $r \in \mathcal{R}' = \{\text{pos\_on}, \text{neg\_on}, \text{neu\_on}\}$ since we only care about sentiment relations in Table 4. We assume $\hat{y}_{*,r}^z >= 0.5$ indicates having a sentiment relation.

To reason dialog policies, we first update $\mathcal{G}_{x-1}$ with sentiment relations as $\mathcal{G}_x$ and use $\mathcal{N}_j^{r,\mathcal{G}_x}$ for the rest 5 layers. We compute dialog acts based on the hidden state of the overall user memory:

$$h^{\text{ag}} = \frac{1}{|\mathcal{C}\cup\mathcal{S}\cup\mathcal{V}|}\sum_{e_j\in\mathcal{C}\cup\mathcal{S}\cup\mathcal{V}}(\boldsymbol{W}^{\text{ag}}h_j^{(5)} + b^{\text{ag}}), \tag{5}$$

where $\boldsymbol{W}^{\text{ag}}$ and b^{ag} are weight for aggregation layer. The loss for dialog acts is defined as

$$\hat{y}^{\mathcal{A}} = \text{Softmax}\Big(\text{MLP}^{\mathcal{A}}(\boldsymbol{W}^{\mathcal{A}}(h^{\text{ag}}) + b^{\mathcal{A}})\Big), \mathcal{L}^{\mathcal{A}} = \text{CrossEntropyLoss}(\hat{y}^{\mathcal{A}}, y^{\mathcal{A}}), \tag{6}$$

where $\boldsymbol{W}^{\mathcal{A}}$ merges the hidden states of dialog acts and graph, $\text{MLP}^{\mathcal{A}}(\cdot)$ is a multi-layer perception for dialog acts and $y^{\mathcal{A}}$ is the label of dialog act. Further, all item, slot and value entities are trained by log loss for ranking. For example, the loss for candidate items $\mathcal{C}$ is computed as

$$\hat{y}_c = \text{Sigmoid}\Big(\text{MLP}^{\mathcal{C}}(h_c)\Big), \mathcal{L}^{\mathcal{C}} = \text{LogLoss}(\hat{y}^{\mathcal{C}}, y^{\mathcal{C}}), \tag{7}$$

where $\text{MLP}^{\mathcal{C}}(\cdot)$ is the multi-layer perceptron for candidate items (similar for $\mathcal{L}_{\mathcal{S}}$, $\mathcal{L}_{\mathcal{V}}$). The total loss is the sum over all losses for dialog acts, items, slots and values:

$$\mathcal{L} = \alpha\mathcal{L}^{\mathcal{A}} + \beta\mathcal{L}^{\mathcal{C}} + \gamma\mathcal{L}^{\mathcal{S}} + \delta\mathcal{L}^{\mathcal{V}} + \epsilon\mathcal{L}^{\mathcal{V},z} + \zeta\mathcal{L}^{\mathcal{I},z}, \tag{8}$$

where $\alpha, \beta, \gamma, \delta, \epsilon$ and ζ are hyper-parameters to balance losses of different scales. Note that during training and prediction, all invalid entities (*e.g.* not in a user memory graph) are masked out. As we can see, unlike traditional recommender systems, E2E-UMGR dynamically infers graph updates and provides the capability of zero-shot reasoning (given no assumption on users/items in the training set). The policy space is open-ended because entities in policy are determined by the rankings of entities in the user memory graph instead of a pre-defined set for the model.

6 Experiments

6.1 Evaluation Metrics

We propose the following metrics to evaluate E2E-UMGR both offline (against the collected testing dialogs) and online (against a user simulator) running on testing scenarios in MGConvRex.

6.1.1 Offline metrics

All offline metrics assume E2E-UMGR uses ground-truth of past turns to evaluate the current turn.
Act Accuracy & F1 are reported for all predicted dialog acts against annotated turn acts in testing.
Entity Matching Rate (EMR, k@1, 3, 5) measures turn-level top-k entities against the testing set. These metrics evaluate only on correctly predicted dialog acts since the types of predicted entities (items, slots, or values) depend on the predicted dialog acts $\hat{y}^{\mathcal{A}}$.
Graph Generation Accuracy (GG Acc.) measures turn-level exact match of generated sentiment relations. We use this metric to investigate the quality of generated sentiment relations from utterances.
Item Matching Rate (IMR) measures dialog-level predicted items against the ground-truth items.

6.1.2 Online metrics

In addition to offline evaluation, we use a user simulator (see Appendix) to dynamically evaluate the performance of recommendation. This mitigates the assumption in offline metrics that all past turns are correct, which limits the interactive evaluation of conversations. Note that the user simulator can only work on structured data and assumes to consume the ground-truth of an updated graph.
Success Rate tracks whether the interaction with user simulators yields the ground-truth item e_t. We use the scenarios for testing sets used for the offline evaluation. The maximum number of turns is simulated as 11. We ran simulations **3** times and average the results.

6.2 Compared Methods

We investigate two settings in the experiment: (1) using the ground-truth of the updated graph to reason dialog policies only; (2) end-to-end reasoning on both graph updates and dialog policies. The first setting allows us to focus on dialog policies and compare with baselines that can only consume structured data (*e.g.* user simulator). We first define a few rule-based baselines:
RandomAgent: we implement a baseline agent that randomly picks a dialog act and randomly picks a candidate item/slot/value as the dialog policy.
RecAgent: this agent always chooses *Recommendation* as the optimal dialog act to enact and select a random item that has not been tried in candidate items (memorize all trials). This is a strong (yet annoying) baseline without any collection of user preferences.

The following baselines map updated user memory graph to dialog policies.
Memory Network(Sukhbaatar et al., 2015; Bordes et al., 2016): we adapt memory network and encodes the user memory graph as triples. The memory can be updated as new triples added. Note that memory networks cannot deal with open space policy because of attention-based aggregation of triple memories. As such, we enumerate all possible combinations of dialog acts and entities in user memory as the space of policy. Specifically, all items in a scenario are indexed as $i_1, i_2, \ldots$ to differentiate candidate items

	Offline Evaluation							Online Evaluation
Methods	**Act Acc.**	**Act F1**	**EMR**			**IMR**	**GG Acc.**	**Success Rate**
			@1	@3	@5			
RandomAgent	18.17	18.24	1.5	1.5	1.5	6.55	-	6.0
RecAgent	25.89	6.86	2.7	2.7	2.7	39.16	-	39.21
Pretrained Emb.	64.3	54.79	13.75	29.02	36.7	9.97	-	9.73
Memory Network	59.46	53.78	13.85	29.46	35.82	4.73	-	6.31
UMGR	65.7	56.54	33.92	48.47	52.54	67.93	-	**71.03**
- Prev. User Act Only	63.47	54.64	33.66	46.69	50.59	**69.71**	-	69.76
- No Dialog Acts	42.37	32.72	31.52	43.66	46.89	67.6	-	66.1
- Static $\mathcal{G}$	64.31	55.25	18.03	36.9	45.31	27.5	-	37.26
E2E-UMGR (Proposed)	64.54	59.32	26.22	45.19	50.78	49.55	52.75	-
- (with graph ground-truth)	**66.47**	**62.01**	**34.98**	**49.83**	**54.17**	66.58	52.75	-

Table 5: Results: EMR stands for entity matching rate, which compares all types of predicted entities against annotated ones when the dialog act is predicted correctly; GG Acc. stands for graph generation accuracy, measuring turn-level exact match of sentiment relations; IMR stands for item matching rate, which evaluates predicted items against the ground-truth item across all turns in a dialog. Online evaluation (via simulator) can only be conducted on structured inputs. So we create UMGR (an variant of E2E-UMGR without NLU) that only works for structured inputs.

for policy generation. The inputs of the memory network are the encoded dialog acts (from human annotations or user simulators). We adopt 5 hops for memory networks.

Pretrained Embeddings: we pre-train the graph embeddings and utilize these as entity encodings for predicting dialog policy (without any R-GCN layers in E2E-UMGR). The graph embeddings are trained from all scenarios in the training set using TransE (Nickel et al., 2016). We use this baseline to show that pre-trained graph embedding alone is sub-optimal for a particular user's scenario.

UMGR: this is a variant of E2E-UMGR without utterance encoding (BERT) and graph updates (layer 1 and 2). To encode dialog acts $\boldsymbol{a}$ from users, we further add a LSTM encoder $h_a = \text{LSTM}(\boldsymbol{W}^{\mathcal{A}}(\boldsymbol{a}))$ and concatenate with h^{ag} to predict dialog acts. To enable zero-shot reasoning on items, all items further share the same embeddings and UMGR can only use the graph structure to reason policy. All sizes of hidden states are 384. The maximum number of past acts is set as 10. Factors of losses α, β, γ and δ are set as 1, 10, 10, 100 based on the scales of corresponding losses. We choose the batch size to be 160.

- Prev. User Act Only: this ablation study of UMGR only uses the most recent dialog act from the user. We use this to show how many past dialog acts are needed for policy generation.

- No Dialog Acts: this study removes the dialog acts encoder, investigating the importance of acts.

- Static $\mathcal{G}$: this study uses the initial user memory graph without any updates during the conversation. We use this study to demonstrate that dynamic updates of the user memory graph are crucial.

E2E-UMGR (Proposed): this is the proposed E2E-model in Section 5. We use the pre-trained BERT from (Xu et al., 2019) on restaurant domains. When training E2E-UMGR, we allow the last 2 layers of BERT to be trainable to learn representations for all utterances (up to 10) into the `[CLS]` token of BERT. Further, all hidden states of entities are computed as the averaged hidden states of our BERT model on the textual form of entities. This is because graph updates require to align unstructured utterances with entities. For the loss, we choose ϵ and ζ to be 1000. During training, we pass the ground-truth $\mathcal{G}_x$ to E2E-UMGR; during inference, we use the generated sentiment relations to update $\mathcal{G}_{x-1}$ as $\mathcal{G}_x$.

- (with graph ground-truth): this ablation study uses the updated ground-truth graph during inference. This helps to investigate the performance drop for errors from graph updates in the above method.

6.3 Results and Discussion

The results are summarized in Table 5. Overall, it can be seen that the proposed baseline E2E-UMGR outperforms other baselines in both offline and online evaluation.

We first investigate E2E-UMGR with both graph updates from unstructured utterances and generation of dialog policy. **Ablations**: *E2E-UMGR -(with graph ground-truth)* outperforms UMGR, probably

because it has access to utterances and all entities are further encoded from their textual forms. The performance of *E2E-UMGR* dropped mostly on IMR and EMR@1, because of errors in graph updates. We expect more complex models in future for error handling in user memory graph.

We then focus on setting (1) that uses ground-truth of graph for policy reasoning. **Ablations**: We notice that dynamically updating the user memory graph with users' new preference is crucial, as indicated by *UMGR - static* $\mathcal{G}$ that forbids updating user memory graph. It can also be seen that removing the previous dialog context does degrade the performance as expected (*UMGR - Prev. User Act Only*), although UMGR still maintains a competitive performance. Similarly, while *UMGR -No Dialog Acts* does not take past dialog acts as input, its results on non-act prediction metrics are relatively competitive. These studies indicate user memory graph contains enough knowledge for reasoning policy.

UMGR vs. Memory Network. We notice that memory networks may not be suitable for complex reasoning over a user memory graph. This may be caused by (1) triples in memory are disconnected, which limits the possibility of joint reasoning of multiple triples; (2) memory network is not structure-preserving, which leads to hardness of aligning entities in triples with the output policy, such as ranking items; (3) dynamic memory is harder than static one in existing research (Bordes et al., 2016; Eric and Manning, 2017; Madotto et al., 2018). Memory network may not be suitable for zero-shot reasoning.

UMGR vs. Rule-based Agent. *RecAgent* is a good baseline on recommendation with the advantage of remembering failed recommendations. But this is not scalable to more candidates.

UMGR vs. Pre-trained Graph Embeddings. We confirm that static pre-trained graph embeddings provide general representations of memory graphs but have a limited capability of reasoning for a particular user's scenario. This study indicates UMGR has the capability for a personalized recommendation.

Discussion We first examine the generated dialog acts. E2E-UMGR typically asks a few questions and then makes a few recommendations. We observe that E2E-UMGR may make more recommendations than expected from agent workers in MGConvRex. This may be caused by the frequent patterns of dialog acts in conversational recommendation: different types of non-recommendation acts are frequently followed by a recommendation act. As a result, a neural network prefers frequent patterns to rare and diverse patterns. We believe encouraging more diverse and detailed patterns is an important direction to improve. Meanwhile, we argue that human performance on reasoning is very limited given the vast amount of candidate items in the real-world recommendation. We expect research on automatic reasoning and explainable models over scalable user knowledge in future work.

Visualization of Item-level Reasoning. In the appendix, we further visualize the prominence scores of candidate items for recommendations at each turn. At the beginning, the prominence scores (and thus the ranking among the candidate items) are soft-initialized to reflect the user's offline preferences. As the dialog progresses and the system collects (or confirms) new user knowledge or a request, E2E-UMGR dynamically updates the ranking of items, reflecting the online preferences.

7 Conclusion

This paper proposes a problem of user memory graph reasoning for conversational recommendation. We expect to release a conversational recommendation dataset with a grounded user memory graph from the behaviors of real-world users. User memory graph has the benefits of accumulating knowledge for a user to reason dialog policy. We propose a baseline model called E2E-UMGR that performs reasoning over such a user memory graph for graph updates and dialog policy in open space. Experimental results demonstrate the effectiveness of E2E-UMGR over a wide spectrum of metrics.

Acknowledgments

We thank Rajen Subba, Alborz Geramifard, and Hao Zhou for insightful discussions. Thanks to Gerald Demeunynck for the discussion and improvement on the process of data collection. Bing Liu (UIC)'s work was supported in part by two grants from National Science Foundation: IIS-1910424 and IIS-1838770, a DARPA Contract HR001120C0023, and a research gift from Northrop Grumman. Philip S. Yu (UIC)'s work was supported in part by NSF under grants III-1763325, III-1909323, and SaTC-1930941.

References

Antoine Bordes, Y-Lan Boureau, and Jason Weston. 2016. Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683*.

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

Guan-Lin Chao and Ian Lane. 2019. Bert-dst: Scalable end-to-end dialogue state tracking with bidirectional encoder representations from transformer. In *Annual Conference of the International Speech Communication Association (INTERSPEECH)*.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. 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)*, pages 4171–4186.

David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. 2015. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232.

Mihail Eric and Christopher D Manning. 2017. Key-value retrieval networks for task-oriented dialogue. *arXiv preprint arXiv:1705.05414*.

Mihail Eric, Rahul Goel, Shachi Paul, Adarsh Kumar, Abhishek Sethi, Peter Ku, Anuj Kumar Goyal, Sanchit Agarwal, Shuyag Gao, and Dilek Hakkani-Tur. 2019. Multiwoz 2.1: Multi-domain dialogue state corrections and state tracking baselines. *arXiv preprint arXiv:1907.01669*.

Shuyang Gao, Sanchit Agarwal Abhishek Seth and, Tagyoung Chun, and Dilek Hakkani-Ture. 2019. Dialog state tracking: A neural reading comprehension approach. In *Special Interest Group on Discourse and Dialogue (SIGDIAL)*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014. The second dialog state tracking challenge. In *Special Interest Group on Discourse and Dialogue (SIGDIAL)*.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177.

Dongyeop Kang, Anusha Balakrishnan, Pararth Shah, Paul A Crook, Y-Lan Boureau, and Jason Weston. 2019. 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)*, pages 1951–1961.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2015. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*.

Xiujun Li, Zachary C Lipton, Bhuwan Dhingra, Lihong Li, Jianfeng Gao, and Yun-Nung Chen. 2016. A user simulator for task-completion dialogues. *arXiv preprint arXiv:1612.05688*.

Raymond Li, Samira Ebrahimi Kahou, Hannes Schulz, Vincent Michalski, Laurent Charlin, and Chris Pal. 2018a. Towards deep conversational recommendations. In *Advances in Neural Information Processing Systems*, pages 9725–9735.

Xiujun Li, Sarah Panda, Jingjing Liu, and Jianfeng Gao. 2018b. Microsoft dialogue challenge: Building end-to-end task-completion dialogue systems. *arXiv preprint arXiv:1807.11125*.

Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. 2018c. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*.

Linyuan Lü and Tao Zhou. 2011. Link prediction in complex networks: A survey. *Physica A: statistical mechanics and its applications*, 390(6):1150–1170.

Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. 2018. Mem2seq: Effectively incorporating knowledge bases into end-to-end task-oriented dialog systems. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1468–1478.

Víctor Martínez, Fernando Berzal, and Juan-Carlos Cubero. 2016. A survey of link prediction in complex networks. *ACM computing surveys (CSUR)*, 49(4):1–33.

Teruhisa Misu, Komei Sugiura, Kiyonori Ohtake, Chiori Hori, Hideki Kashioka, Hisashi Kawai, and Satoshi Nakamura. 2010. Modeling spoken decision making dialogue and optimization of its dialogue strategy. In *Proceedings of the SIGDIAL 2010 Conference*, pages 221–224, Tokyo, Japan, September. Association for Computational Linguistics.

Seungwhan Moon, Pararth Shah, Anuj Kumar, and Rajen Subba. 2019a. Opendialkg: Explainable conversational reasoning with attention-based walks over knowledge graphs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 845–854.

Seungwhan Moon, Pararth Shah, Rajen Subba, and Anuj Kumar. 2019b. Memory grounded conversational reasoning. 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): System Demonstrations*, pages 145–150.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016. Holographic embeddings of knowledge graphs. *AAAI*.

Trang Pham, Truyen Tran, Dinh Phung, and Svetha Venkatesh. 2017. Column networks for collective classification. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2019. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2008. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

Yueming Sun and Yi Zhang. 2018. Conversational recommender system. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 235–244. ACM.

Yi-Lin Tuan, Yun-Nung Chen, and Hung-yi Lee. 2019. DyKgChat: Benchmarking dialogue generation grounding on dynamic knowledge graphs. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1855–1865, Hong Kong, China, November. Association for Computational Linguistics.

Tsung-Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gasic, Lina M. Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2016. A network-based end-to-end trainable task-oriented dialogue system. In *European Chapter of the Association for Computational Linguistics (EACL)*.

Chien-Sheng Wu, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher, and Pascale Fung. 2019. Transferable multi-domain state generator for task-oriented dialogue systems. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Yikun Xian, Zuohui Fu, S. Muthukrishnan, Gerard de Melo, and Yongfeng Zhang. 2019. Reinforcement knowledge graph reasoning for explainable recommendation. In *SIGIR*.

Hu Xu, Bing Liu, Lei Shu, and S Yu Philip. 2019. Bert post-training for review reading comprehension and aspect-based sentiment analysis. 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)*, pages 2324–2335.

Yongfeng Zhang, Xu Chen, Qingyao Ai, Liu Yang, and W Bruce Croft. 2018. Towards conversational search and recommendation: System ask, user respond. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 177–186. ACM.

Li Zhou, Jianfeng Gao, Di Li, and Heung-Yeung Shum. 2020. The design and implementation of xiaoice, an empathetic social chatbot. *Computational Linguistics*, 46(1):53–93.

A MGConvRex Dataset

This section describes the construction of the *MGConvRex* dataset. *MGConvRex* aims to contain dialogs that draw relevance of the user's history and fine-grained user preferences to update the user memory graph. As such, we propose to leverage existing data from recommender systems [8] that carry users' past behavior to harvest large-scale dialog scenarios. Then we define fine-grained dialog acts, slots, values and sentiment polarities to turn unstructured utterances into structured knowledge for graph updates.

A.1 Wizard-of-Oz Collection

We build a wizard-of-oz system to randomly pair two crowd workers to engage in a chat session, where each scenario $(e_u, C, H, V, P, \mathcal{T})$ is split into two parts: $(P, \mathcal{T})$ for the user and (e_u, C, H, V) for the agent. The goal of a conversation is like a game between the user and the agent, where the agent needs to reason the user's current preference and find the ground-truth item and the user can tell information from preference P or confirm a recommended item $e_i \in \mathcal{T}$ but cannot tell the ground-truth directly.

A.1.1 Motivation

Getting irrelevant restaurant recommendations is a frustrating experience. The ideal recommendation system should be able to provide better recommendations by understanding your current needs, your restaurant preferences, and your restaurant history.

A.1.2 Overview

In this project, you will generate a dialog between an imaginary person (*user*) and an imaginary recommendation system (*assistant*[9]). You will play one of the two roles, that will randomly be assigned to you. You will automatically get paired with someone else who will play the other role.
User: A user is expected to interact with an assistant to get a restaurant recommendation. The user will already know his/her general restaurant preferences and also the exact name of the restaurant he/she wants to go to. Further, information about restaurants that the user has visited in the past will be available and shown to the user.
Assistant: An assistant is expected to interact with the user and work towards recommending a restaurant the user wants to go to in the future. The assistant will have access to information about restaurants that the user has previously visited and a list of candidate restaurants.

A.1.3 Task

You will be randomly assigned a single role: either *user* or *assistant*. You will see your assignment in the top left corner of the screen, "You are: the user" or "You are: the assistant".
User: You will interact with the assistant, to get the correct restaurant recommendation from the assistant. You will be provided with the following information:

- Restaurant preference over 10 characteristics (or slots).
- The restaurant you will go to: "Ground-Truth restaurant".
- You will optionally have information about restaurants that you have visited in the past.

As a user player, you are expected to:

- Answer the questions the assistant asks about your preference.
- Reject incorrect restaurant recommendations.
- Ask questions about the recommended restaurant to justify why you accept or reject the recommendation.
- If needed, use the information in your *visited restaurant* to help inform the assistant about your preference.

[8] We focus on the restaurant domain at this stage.
[9] We term agent as "assistant" in guidelines.

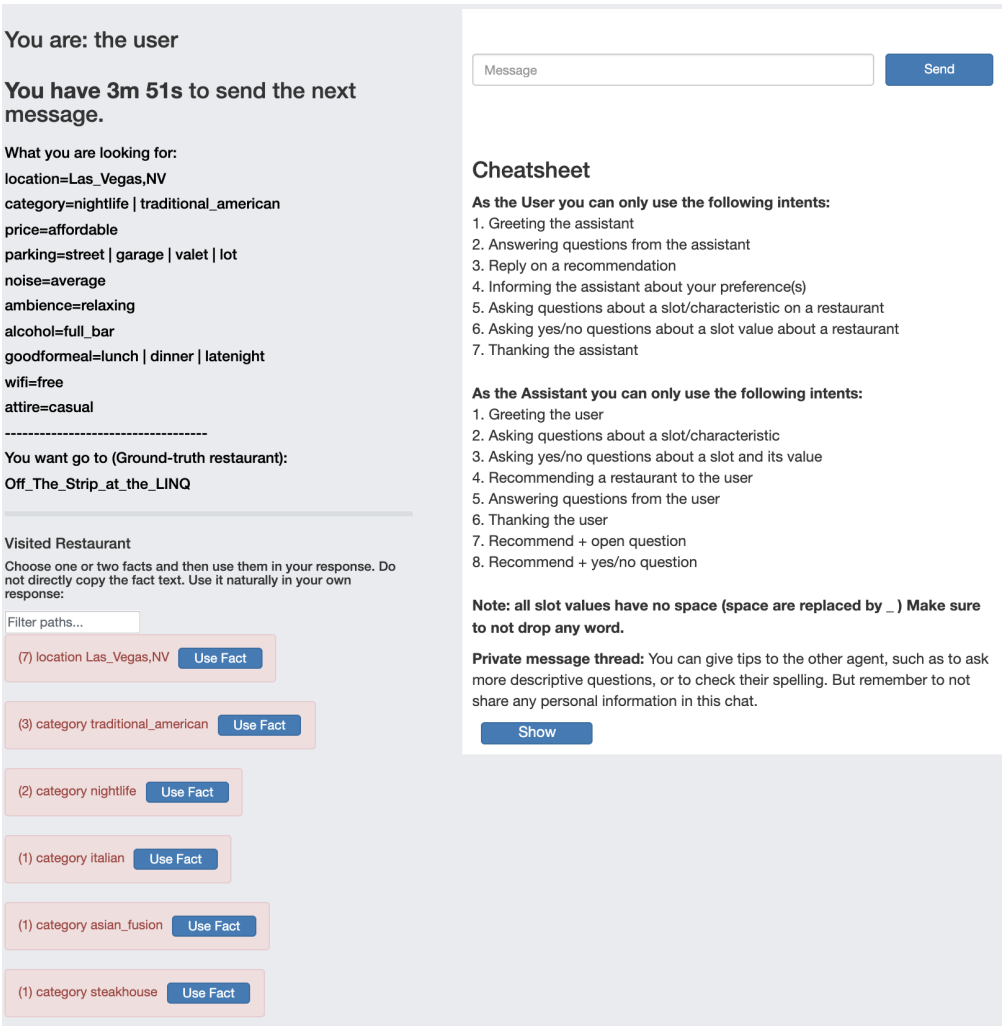


Figure 4: Screenshot of transcription UI for User.

- The frequency of characteristics (or slots) shared by multiple restaurants are indicated in (...), *e.g.* "(3) parking lot" means this user has been to 3 restaurants with parking lots.
- When you use information from your *visited restaurant*s in one of your responses, make sure to click the "Use Fact" button.

Assistant: You will interact with the User, to give the desired recommendation (ground-truth restaurant) to the user. You will be provided with the following information:

- Name of the user.
- A list of candidate restaurants, and their characteristics (slots). One of the restaurants in this list is the desired or *ground-truth restaurant* the user is looking for.
- Optionally, the characteristics (slots) and values of the restaurants the user has visited (*visited restaurants*). (See the definitions of slots below). The frequency of slots shared by multiple restaurants are indicated in (...), *e.g.* "(3) parking lot" means this user has been to 3 restaurants with parking lots. The visited restaurants' section may or may not be given to you. If it is given, your goal is to utilize (by clicking "Use Fact") the information from *visited restaurant*s as much as possible to provide the desired recommendation to the user.

To make an efficient recommendation, you are expected to:

- Ask the user questions about their restaurant preference.

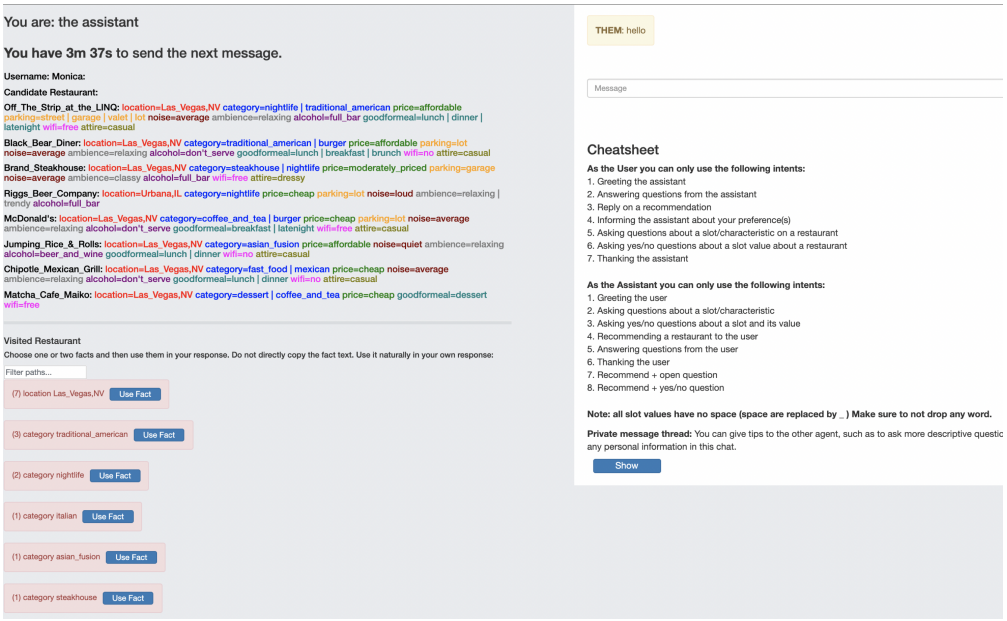


Figure 5: Screenshot of transcription UI for Assistant.

- If the visited restaurants are available, *investigate* their slots and values to reduce the number of questions you may need.
- Recommend restaurants to the user based on your knowledge of their preference, their visited restaurants, the information of the candidate restaurants, and from the answers the user gives to your questions.
- Intelligently apply the information the user gives to you to guide your conversation.
- Recommend the desired restaurant.

A.1.4 Instructions

This section describes the details of transcription. In general, transcribers are required to follow pre-defined *dialog acts*, *slots* and *values*, but free to make up utterances based on these pre-defined metadata.
Dialog Acts are the intents of one utterance from a player. Note that the user and assistant have their own set of dialog acts, as shown in Table 1. You can only use these pre-defined dialog acts in your utterance.
Slots refer to 10 pre-defined characteristics of restaurants.
Values: one slot is further associated with multiple *values*, such as a slot *Parking* can take value *street*. Note that a slot can take multiple values at the same time. In the UI, these values are separated by "|". For example, *Parking* = *street* | *garage* means that a restaurant has both street and garage parking. DO NOT include "|" in your responses, instead, use one or multiple values naturally in the utterance. *e.g.* "I prefer street or garage parking." You do not have to write out all the values of a slot in one utterance. For example, the *category* slot usually has many values and you do not need to list them all in your utterance.

You will need to write the values exactly as you see them in the UI, including the underscores "_" and commas "," and excluding "|". For example, type "Bonfyre_American_Grille" but not "Bonfyre American Grille". The full lists of values and their slots are at the ends of guidelines.
Items and their Names: Each item (restaurant) has an item name and has multiple values and their associated slots. An item is typically associated with a *recommendation* act from the assistant side. When recommending a restaurant (item), you are expected to mention the restaurant name (*item name*), which follows the same rule as writing a value in an utterance.

A.1.5 Important Notes

During transcribing, it is important to keep these things in mind:

- A dialog can end with either a user or an assistant response.

- The person who plays the user, however, will be the one to terminate the session by pressing the button "Dialog is done!"
- The user should NEVER give all of their preference to the assistant in a single utterance.
- The user should NEVER give the *ground-truth restaurant* to the assistant.
- When you use content from the *visited restaurant*s in your response, make sure to click the corresponding "Use Fact" buttons before sending your response. The click will be recorded.
- If the user player has sent more than 10 responses (20 including the responses from the assistant), it is up to the user player to decide whether to stop the current dialog or to continue.

The following actions should be avoided.

- Do not engage in the transcribed dialog with the other person about the transcription task itself and do not go off-topic.
- Do not share any of your personal information. Always be "in your character", i.e., speak as the *user* or the *assistant*.
- NO INDECENCY / DISRESPECT / HARASSMENT. Keep your messages decent and respectful towards the other person. Any violations will result in a ban on further tasks.
- Do not directly copy any of the utterances from this guideline or UI.
- Do not repeat /template your answer, that is to say, do not create one set of responses ahead and then make small changes to them over and over. Please always generate unique and new responses.

A.1.6 Feedback

After the transcription of one dialog is over, both sides need to give feedback about the transcribed dialog, including:

1. Rate the dialog (1-5) based on the smoothness and coherence of the whole dialog and whether it closely follows this guideline.
2. Rate the other side (1-5): whether the other side closely follows this guideline.
3. (Optional) feedback about this transcription task.

A.2 Annotation Guideline

In this task, you will get a transcribed dialog between a *user* and an *assistant*, in which the assistant helps the user find the desired restaurant to go to. You will annotate the utterances with *dialog acts*, *slots*, *values*, *item names* and *sentiment* on values or item names. For your reference, the transcription guideline is detailed in Section A. This annotation task will be further supported by a QA process before and during the annotation to resolve hard cases.

A.2.1 Task

In this annotation task, you are required to label the following data:

1. dialog quality: *good* or *bad* about the whole dialog.
2. dialog acts (or utterance-level intents), as defined in Table 1.
3. Label spans of values (or item name) from each utterance and their corresponding slots (or item).
4. Utterance-level sentiment of each utterance, and optionally span-level sentiment towards a value (or item name) if it is different from the utterance-level sentiment.

You first need to read through the dialog once and label the overall dialog quality, and if it is *good*, label dialog acts. Then you need to read through the utterances again and label spans of values (or item name), their corresponding slots (item), and sentiment.

A.2.2 Dialog Quality

For the entire dialog, you will need to label the dialog quality as either *good* or *bad*. This step is to further ensure the quality of the transcribed dialog. If the dialog quality is labeled as *bad*, you can skip annotating the current dialog further.

A.2.3 Dialog Acts

Each utterance must have at least one dialog act. The dialog acts are pre-defined in Table 1. Note that there are different sets of dialog acts for the roles of *user* and *assistant*. If you believe one utterance is associated with multiple dialog acts, you need to label all of them. We summarize a few important tips for user and assistant separately as following.

Dialog Acts for User: There are a few key differences among *reply*, *answer*, *inform*, *open* and *yes/no question*.

- *reply*, *open* and *yes/no question* are always related to a (previously) recommended restaurant (from the assistant). The item name (restaurant name) may or may not show up in the to-be-labeled utterance.
- *answer* and *inform* are always related to a value. Note that the value (span) may not show up in an answer (*e.g.* "Yes, I like that location.")
- *open question* DOES NOT have a value show up in the utterance but only the explicit or implicit slot (*e.g.* "what type of food do they serve ?" [*category*]) , whereas *yes/no question* must have a value show up (*e.g.* "do they serve *Italian* food ?").
- *inform*, *open* and *yes/no question* indicates a user actively providing information, while *reply* and *answer* indicate a user passively giving information.

Dialog Acts for Assistant: The key differences among *recommendation*, *open question*, *yes/no question* and *answer* are as following:

- *recommendation* and *answer* are always related to a restaurant (item). A *recommendation* act may have additional values show up, besides the restaurant name. The restaurant typically may not show up in *answer* (*e.g.* "it serves italian food.")
- *open* and *yes/no question* are always only about slots. But the slot itself may not show up in the utterance directly (*e.g.* "what kinds of food do they serve ?").
- *yes/no question* always has a value show up: "do you like *italian* food ?"

Note that you always need to annotate the true intent of having an utterance, not the surface form of an utterance. For example, a *recommendation* can have a surface form that looks like a question (e.g., "how about burger_king ?" and "why not try burger_king ?").

A.2.4 Spans of Values, their Slots, Items, and Sentiment

We expect you to label spans of words that are values (of slots) in the utterance (or item names), all possible values that you can label are listed at the end of this guideline [10]. You are also required to label slots when the values are not shown in an utterance (e.g., an *open question*) by just labeling the slots on utterance-level (similar to label a dialog act). Finally, label utterance-level sentiment (one of *positive*, *negative* and *neutral*) and span-level sentiment (if it differs from utterance-level sentiment).

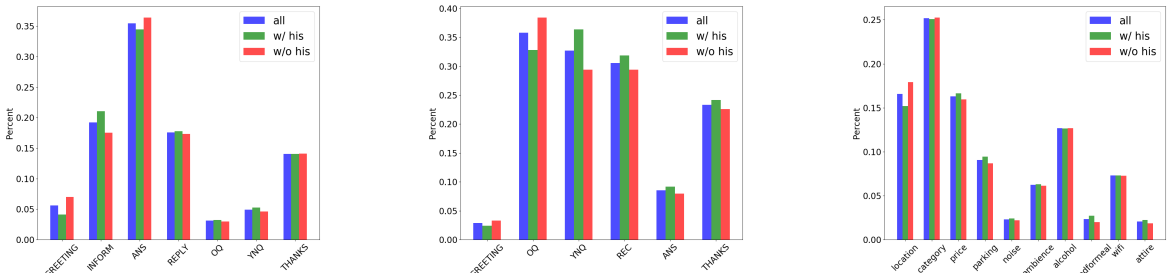
We expect you to perform the following steps (after you finish labeling dialog acts):

1. label spans of words as values (or item names).
2. select the corresponding slot (or item).
3. label utterance-level slot (*open question*).
4. label utterance-level sentiment and check and label span-level sentiment.

[10] We omit the list in this appendix for brevity.

Role	Utterance	Acts
User	hello.	Greeting
Assistant	Hello Will! Are you still living in the **Phoenix,AZ**$_{\text{location}}$ area. ?	YNQ
User	Yes. [*pos_on*]	ANS
Assistant	Ok great! Do you want a **full bar**$_{\text{alcohol}}$ with your meal?	YNQ
User	No just **beer and wine**$_{\text{alcohol}}$[*pos_on*] are fine.	ANS
Assistant	My system shows **The Nash**$_{\text{item}}$ restaurant. They also offer **free**$_{\text{wifi}}$ wifi.	REC
User	Is the food **cheap**$_{\text{price}}$[*pos_on*] over there? I'm tight on budget.	YNQ, inform
Assistant	It is on the **cheap**$_{\text{price}}$ side of the restaurants.	ANS
User	Great. I'll try them out. Thanks.	Reply & Thanks
Assistant	Thank you and enjoy your meal	Thanks

Table 6: An example dialog in MGConvRex with slots and sentiment polarities annotated.



(a) Distribution of user dialog acts

(b) Distribution of agent dialog acts

(c) Distribution of slots asked in Open and Yes/no questions from the agent

Figure 6: Distribution of dialog acts and slots.

A.2.5 Important Notes

- dialog utterances may have typos (e.g., extra spaces, cases), correct and label the spans to the best of your ability, even if errors are present.
- Do not label spans about slots (e.g., *location*, *category*, *price*, etc.) itself, such as words "where", "located at", "kinds of", "price range", "parking" etc. Labeled spans should only be about pre-defined values or item names (restaurant names).
- Utterances from the *assistant* side DO NOT have sentiments.
- Only utterances from a user fall in these dialog acts have sentiment: *answer*, *inform*, and *reply*.

A.3 Dataset Processing and Analysis

A.3.1 Data Cleaning

After annotation, the data will go through a data cleaning process via scripts to fix typos and illegal combinations of dialog acts, items, slots, and values. The cleaned data will be integrated with scenarios of each transcription task to form the final datasets.

A.3.2 Statistics

Besides the statistics of MGConvRex in Table 7, we further study the distributions of dialog acts and slots from the assistant side to learn more about the preferred behavior of crowd workers. From Figure 6a, we can see that a user worker mainly uses the *Answer* act to the agent. More importantly, the user

Dataset	All Dialogs			Dialogs w/ History		Dialogs w/o History	
	# of Dial.	# of Turns	Avg. # of Turns	# of Dial.	Avg. # of Turns	# of Dial.	Avg. # of Turns
Train	4985	48457	9.72	2418	9.62	2567	9.81
Dev	263	2466	9.38	121	9.16	142	9.56
Test	2367	23048	9.74	1160	9.62	1207	9.85

Table 7: **Dataset Statistics**: Dialogs w/ or w/o History indicates whether scenarios include visited items $\mathcal{H}$.

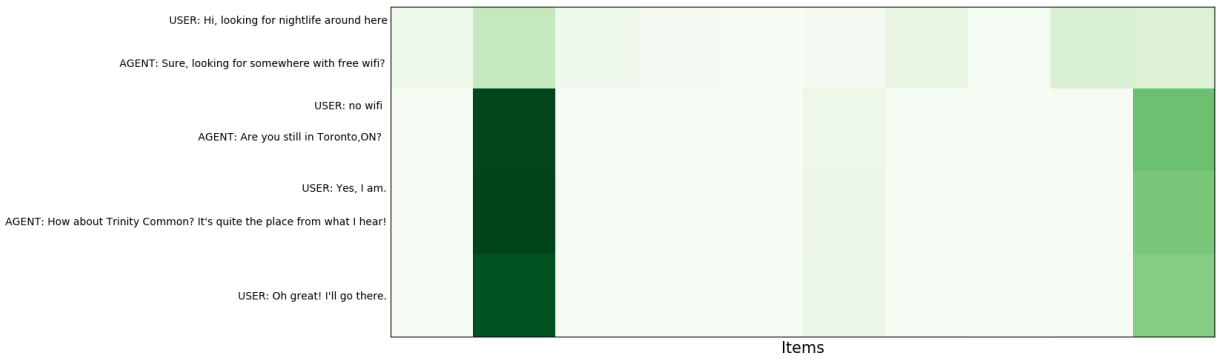


Figure 7: Visualization of item-level conversational reasoning, given an example dialog. Darker color indicates more salient items for recommendation at each given turn (row), predicted by our E2E-UMGR model.

player is very active and likes to use the *Inform* act without being asked a question. User players can sometimes even more cooperative by examining the user memory and inform more salient preference, as indicated by more *inform* in scenarios with history. From Figure 6b, we can see that an agent worker use both *Open question* and *Yes/no questions* to collect preference. Yes/no questions are more frequent in scenarios with history to confirm users' preferences. Figure 6c shows the distribution of slots for *Open and Yes/no question*s asked by the agent player. *category, location, and price* are their mainly used slots for collecting user preference and distinguish different candidate items in $\mathcal{C}$. We further demonstrate one example dialog is shown in Table 6.

A.4 User Simulator

Our online evaluation is conducted against user simulators under a simulation environment in our developed framework. Here we brief one simulator as in Algorithm 1. Note that although our pre-defined dialog acts for user can be either passive or active (as in Table 1), we mostly focus on a passive user (less likely to use *Inform*, *Open / Yes/no question*, *e.g.* 0.2 chance to make an *inform* act) because an active user can vary domain-by-domain and hard to implement. In the implementation, the user mostly follows the dialog acts from the agent and provide information accordingly in a case-by-case fashion.

A.5 Visualization of E2E-UMGR

We further explore the dynamics of recommendation. We plot the saliency of recommending an item in Figure 7. At the beginning of the dialog, the prominence scores (and thus the ranking among the candidate items) are soft-initialized to reflect the user's offline preferences, as indicated in the user memory graph. We can see that E2E-UMGR can almost predict the ground-truth item. As the dialog progresses and the system collects (or confirms) new user knowledge or a request, E2E-UMGR dynamically updates the ranking of items, reflecting the online preferences.

Algorithm 1: Algorithm for User Simulator

Input : a, e_i, e_s, e_v from the agent; P and $\mathcal{T}$ from scenario
Output: a', e'_i, e'_s, e'_v, o' from the user

def *RandomGreetInform*():
    if *random* < 0.8 **then**
        $a' \leftarrow$ *GREETING*
    end
    else
        $a' \leftarrow$ *INFORM*
        $e'_v, o' \leftarrow$ RandomValue(), *pos_on*
    end
    return a', e'_v, o'
$a', e'_i, e'_s, e'_v, o' \leftarrow$ Nones
switch a **do**
    case *INIT* // first turn
     do
        $a', e'_v, o' \leftarrow$ RandomGreetInform()
    end
    case *REC* **do**
        if $e_i \in \mathcal{T}$ **then**
            $a', o' \leftarrow$ *REPLY*, *pos_on*
            SetDialogSuccess()
        end
        else
            $a', e'_s \leftarrow$ *OQ*, RandomSlot()
        end
    end
    case *OQ or YNQ* **do**
        $e'_v, o' \leftarrow$ FindValueOpinion(P)
    end
    case *ANS* **do**
        if P *has* e_v **then**
            $a', o' \leftarrow$ *INFORM*, *pos_on*
        end
        else
            $a', o' \leftarrow$ *INFORM*, *neg_on*
        end
    end
    case *THANKS* **do**
        if *IsDialogSuccess*() **then**
            $a', o' \leftarrow$ *THANKS*, *pos_on*
        end
        else
            $a', e'_v, o' \leftarrow$ RandomGreetInform()
        end
    end
end
return a', e'_i, e'_s, e'_v, o'