Learning-to-Explain: Recommendation Reason Determination Through Q20 Gaming

Xianchao Wu*
xiancwu@microsoft.com
Microsoft Development Co., Ltd.
Tokyo, Japan

ABSTRACT

Real-world chatbots face challenges for performing explainable product recommendations. First, large-scale labeled corpora are frequently required to capture users' natural language expressed intentions. Second, usage threshold is high for user-teaching due to information asymmetry between users' requirements and chatbot's supplies. Third, it is non-trivial for the chatbot to collect users' requirements step-by-step. In order to tackle these challenges, we propose a *learning-to-explain* framework which constructs a *question chatbot* that asks limited number of option-attached questions. The framework starts from product-attribute tables of target domains and conditionally ranks the next option-attached question using entropy-based and reinforcement-learning algorithms. The question-answer pair that brings the most probability jump to the top product is taken as the major hint for rule-based recommendation reason construction. We testify our explainable recommendation framework in scenarios of general gift recommendation and task-oriented hotel/restaurant booking recommendation.

KEYWORDS

explainable product recommendation, explainable gift recommendation, Q20 games, reinforcement learning, question ranking

ACM Reference Format:
Xianchao Wu. 2019. Learning-to-Explain: Recommendation Reason Determination Through Q20 Gaming. In *Proceedings of SIGIR 2019 Workshop on ExplainAble Recommendation and Search (EARS'19).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Chatbots for personalized product recommendation [12, 15] is absorbing intensive attentions in both research and industry fields. For example, user profiles are supposed to be constructed through emotion-oriented (such as Microsoft's XiaoIce [8, 15]) and task-oriented (such as Amazon's Echo, Google Home) conversations. As far as fine-grained user profiles are built, existing recommendation algorithms can be employed for satisfying users' consuming intentions that are included in (or, mined from) user profiles by prepared products.

However, the processes for understanding and satisfying users' requirements are not-trivial. First, labeled data are required for classifying users' intentions. For example, when the chatbot does not understand or miss-understand the intentions (which happens quite frequently), normal conversation is replaced by awkward chatting. Second, users' requirements and chatbot's product list are independently updating timely. Due to this, it is quite often that information asymmetry happens: the user does not know what to ask and the chatbot does not know what to reply. Third, in a normal conversation, user's intentions are expressed through non-contiguous queries. The long-distance dependencies among consuming intentions and detailed constraints challenge the chatbot's ability for information collecting.

On the other hand, recommend products and attach appropriate reasons [13] are becoming more and more important for its user friendly interface and for utilizing similarities of among users and/or products (e.g., buy related products together and obtain a discount) based on users' buying or browsing histories. The benefits of employing chatbots for explainable product recommendation are straight-forward: (1) conversational sessions supply an abundant way of contents for recommendation reason construction, and (2) user profiles make it feasible to compute the similarities or distances of projecting users into dense spaces.

In this paper, we propose a novel *learning-to-explain* (LTE) framework for chatbot's explainable product recommendation. Our LTE framework is designed to rank products and determine their recommendation reasons by asking the user a sequence of option-attached questions through a chatbot. The idea is largely motivated by the Q20 games [2, 4, 11]: the chatbot tries to guess the entity (such as, famous people, animals, plants, or movies) in the user's mind through asking no more than 20 questions. Based on user's answer to the former question, the next question is ranked in a way that is supposed to bring the maximum information gain regardless of user's selection. Each question can be derived explicitly or implicitly from the attributes of products. Every time the framework receives user's answer to current question, it updates the ranking of the candidate products to ensure that products containing user's selected attributes are ranked higher. We also remember the "probability jump" (or, "information gain", referring Equation 4 and 5) of candidate products based on user's answer. Finally, when the number of questions reaches a threshold or the first candidate product is assigned with a high enough probability, we stop the questioning process. The top candidate product is attached with a question-answer pair that caused the maximum probability jump. They are shown together to the user as the final recommendation result (refer to Figure 1, 7, and 9 for examples).

EARS'19, July 25, 2019, Paris, France

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

We limit our discussion to (1) general gift recommendation, and (2) task-oriented service booking recommendation. For gift recommendation, the user is supposed to know for *whom to buy* (such as family members or friends) but do not know *what to buy*. Then, the chatbot's task is to instantly construct a gift-receiver's profile through asking a sequence of option-attached questions. The questions are related to gender and hobbies. This is more alike "mind-reading" based recommendation: the chatbot (1) paints the characteristics of the gift-receiver that is reflected in the user's mind and (2) recommends a gift based on a reason that depends on user's specific answers to a list of questions. For service booking recommendation, we use the Multi-WOZ [1] corpus and construct question chatbots for five domains.

Figure 1 illustrates two real examples of the question-answer processes based on our LTE framework. We collected 82 products and 20 questions that are related to "Harry Potter" from famous e-commerce websites. In these two examples, the first two questions are always the same of first asking whom to buy and how much to spend. After asking several questions, the chatbot respectively recommends product "Divination Planner" majorly due to the selection of "hufflepuff" in question 4 and product "Wand" for the selection of "gryffandor" in question 5. Since users' selections are diverse and not pre-given, we propose this LTE framework which is designed to collect users' detailed requirements through a conversational way. The background consideration is alike specifying user's preferred products by playing "jigsaw puzzles" in a Q20 game way, where each question is a piece of hint that links the candidate products with current gift-receiver.

Traditional search engine or collaborative filtering [3] based methods are not well suitable for these scenarios. The first is that search engines will need target product related keywords for keyword-product matching. The second is that collaborative filtering will make a recommendation based on other/same people's buying history. However, we prefer a user-friendly way of first talking to the user and then making an explainable suggestion based on asking a number of questions. Note that there are gaps of information richness of between the chatbot and the users. The users are supposed to be unfamiliar with the details of the products and they need a user-friendly way to obtain these information. These motivated our LTE framework for chatbots aiming at user-specific and user-friendly product recommendation.

The novel ideas in our framework include: (1) we dynamically detect user's detailed requirements of gift buying and service booking, (2) we attach selectable options to the questions, so that the role of users is switched from "information supplier" to "option learner and selector" and (3) we propose entropy-based and reinforcement learning based methods for ranking questions based on user's answering history. The target of this LTE framework is to recommend the optimal products while asking as few questions as possible.

The *explainable* property of our framework is reflected in two aspects, (1) the sequence of option-attached questions that shows the explicit textual guidance of the recommendation process and (2) the final shown of the question-answer pair that helped the most for determining the top candidate product.

U: Help me find a gift related to "Harry Potter"
Q1: Sure, who are you shopping for?
  1.myself; 2.friend; 3.someone; 4.aunt petunia
U: 2
Q2: How much do you want to spend? 1.$; 2.$$; 3.$$$
U: 2
Q3: if you were shopping together in Diagon Alley,
  where would they be drawn to?
  1.weasleys' wizard wheezes joke shop;
  2.madam malkin's robes for all occasions;
  3.flourish & blotts books;
  4.florean fortescue's ice cream parlour
U: 4
Q4: Guess, which house would they be sorted into?
  1.ravenclaw; 2.hufflepuff; 3.gryffandor; 4.slytherin
U: 2
Q5: Which OWLS would they get the highest marks in?
  1.potions; 2.history of magic;
  3.defence against the dark arts; 4.herbology
U: 2
R: I would like to recommend you: Divination Planner,
  guided by the selection of "hufflepuff" in Q4.

U: Help me find a gift related to "Harry Potter"
Q1: Sure, who are you shopping for?
  1.myself; 2.friend; 3.someone; 4.aunt petunia
U: 1
Q2: How much do you want to spend? 1.$; 2.$$; 3.$$$
U: 3
Q3: Where would you prefer to spend your time?
  1.common room; 2.quiddich pitch; 3.leaky cauldron; 4.library
U: 4
Q4: Would you say you are a :
  1.witch; 2.wizard; 3.generally magical; 4.a cool muggle
U: 2
Q5: Guess, which house would you be sorted into?
  1.ravenclaw; 2.hufflepuff; 3.gryffandor; 4.slytherin
U: 3
R: I would like to recommend you: Wand,
  guided by the selection of "gryffandor" in Q5.

Figure 1: Two real examples under our LTE framework for "Harry Potter" related gift recommendation. U = user, Q and R are respectively questions and recommendations with reasons from the chatbot.

2 LEARNING-TO-EXPLAIN FRAMEWORK

Our LTE framework manages chat flow, dialog state and question ranking through leveraging the 20-question-game idea [2, 4, 11, 14]. Figure 2 shows a reference answer matrix D_i which is designed to link the product set P_i and the question set Q_i for each category i (e.g., "Harry Potter" category, general gift category, or service booking domains). One element in D_i is denoted as v^i_{mn} in which m is the index for a product p_m and n is the index for a question q_n. The question set Q_i comes from (1) the attributes and their values of products, and/or (2) hand-made questions that help collecting

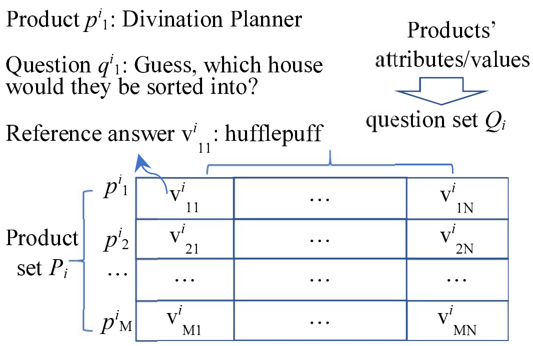

Figure 2: The matrix that links *products* and *questions*.

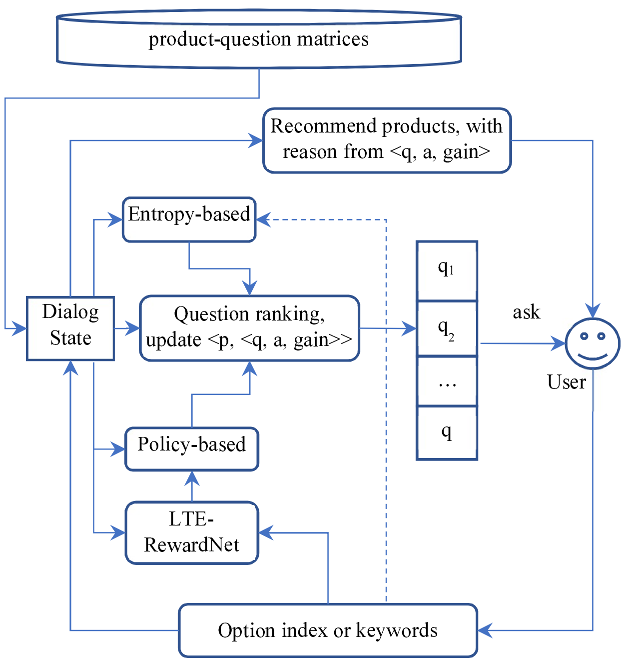

Figure 3: Our LTE framework as a question chatbot.

the characteristics of gift-receivers (e.g., questions listed in Figure 1).

The LTE framework (Figure 3) includes two modules: (1) constructing the database of P, Q and their relation matrices D and (2) applying the database through a chatbot for question ranking, user's response processing, question-answer gain collecting, and candidate product recommending.

3 ENTROPY-BASED QUESTION SELECTION

The entropy-based question selection method was described in [4, 11] for famous people (who exists in real or virtual world) guessing. In this paper, we adapt this method to the field of explainable product recommendation. The major idea is to select the next question that can prune as many candidate entries as possible, no matter the user's answer is *yes*/*no* for binary-answer questions (such as, "is he over 10 years old?") or any specific selection of the options that are attached to multiple-choice questions (as used in Figure 1).

Initially, for one category i, we assign a prior popularity weight $w(\cdot)$ to each candidate product p^i_m ($1 \leqslant m \leqslant M$). The weight can come from (1) search frequency in search engine, or (2) review score or ordering frequency in e-commerce websites. Then, we normalize $w(\cdot)$ by:

$$w'(p^i_m) = \frac{w(p^i_m)}{\sum_{m=1}^{M} w(p^i_m)}. \quad (1)$$

For one candidate product p^i_m, we compute its contribution $Y^l_{m,n}$ of selecting a question q_n:

$$Y^l_{m,n} = \frac{f^l_{m,n} + \alpha I(v^{i,l}_{m,n} = yes)}{\sum_{l=1}^{L_n} \{f^l_{m,n} + \alpha I(v^{i,l}_{m,n} = yes)\}}. \quad (2)$$

Here, $f^l_{m,n}$ stands for the frequency that users selected option l of question q_n for a final chosen of product p^i_m. $I(\cdot)$ stands for an indicator function that returns 1 when $v^{i,l}_{m,n}$ equals to *yes* and 0 otherwise. We introduce a parameter α here to balance users' selections and the reference answer included in the product-question matrix D_i. Extremely, when α is 0, we only trust users' historical selections and ignore the reference answers. On the other hand, when we set α to be extremely large, we only trust the reference answers and users' historical usage information is (almost) ignored. This is a way of learning from the users to better assist the booking service. It is straightforward to extend α to a time-decay function $\alpha(t)$ where t stands for time.

When ranging over each candidate p_m, we obtain Y^l_n which stands for the importance of option l in question q_n:

$$Y^l_n = \sum_{m=1}^{M} w'(p^i_m) \times Y^l_{m,n}.$$

We set the negative variance of Y^l_n for ranking question q_n:

$$w(q_n) = -\sum_{l=1}^{L_n} (Y^l_n - \frac{\sum_{l=1}^{L_n} Y^l_n}{L_n})^2.$$

A slightly different way is to use the negative Shannon entropy for the Multivariate Bernoulli distribution of options:

$$M_{m,n} = \sum_{l=1}^{L_n} Y^l_{m,n} log_2 Y^l_{m,n}.$$

Then, the weight of a question q_n is defined as:

$$w'(q_n) = \sum_{m=1}^{M} w'(p^i_m) \times M_{m,n}. \quad (3)$$

After ranking the questions, we can either select the top-1 question or randomly pack one from the top-N question candidates. Basing on user's answer to current selected question q_{n_0}, we update the weight for each candidate product p^i_m:

$$w(p^i_m) = w'(p^i_m) \times \sum_{l=1}^{L_{n_0}} I(v^{i,l}_{m,n_0} = a) \times w'(q_{n_0}).$$

Here, a stands for user's answer of current question q_{n_0}. Since $w(p^i_m)$ will be 0 in case user's answer a does not match any options of question q_{n_0}, we smooth $I(\cdot)$'s return to be a tiny value (e.g., 0.01) instead of 0 to avoid the complete deleting of that product candidate. Then, we normalize $w(p^i_m)$ to generate next stage $w'(p^i_m)$ using Equation 1 again. We attach subscripts t and $t+1$ to respectively denote current stage and next stage $w'(p^i_m)$, i.e., $w'_t(p^i_m)$ and $w'_{t+1}(p^i_m)$. Consequently, the $t+1$ step "information gain" for a product p^i_m caused by user's answer a to current question q_{n_0} is:

$$g_{t+1}(p^i_m, q_{n_0}, a) = w'_{t+1}(p^i_m) - w'_t(p^i_m). \tag{4}$$

The final maximum "information gain" will be selected from the list of $g(\cdot)$. We skip the category index i for most variables except p^i_m for simplicity.

4 POLICY-BASED REINFORCEMENT LEARNING

4.1 General Environment

Policy-based reinforcement learning algorithms have been used in [2, 4] for 20-question games such as guessing of famous people with limitations of only allowing game players to answer *yes*, *no*, or *unknown* for the single-attribute questions. We adapt this method to suitable to multiple option attached questions for explainable product recommendation.

Alike the original 20-question game, we formulate the process of question ranking as a finite Markov Decision Process (MDP) expressed by a 5-tuple $\langle S, A, P, R, \gamma \rangle$, where S is the continuous dialog state space and one state $s \in S$ stands for a vector that stores the probabilities of the products, $A = \{q_1, ..., q_n\}$ is the set of all available actions (i.e., questions in our scenario), $P(S_{t+1} = s' | S_t = s, A_t = q)$ is the state transition probability matrix, $R(s, q)$ is the reward function and $\gamma \in [0, 1]$ is a decay factor to discount the long-time returns. In our policy-based reinforcement learning algorithm, at each time-step t, the question chatbot takes an action q_t under the current state s according to the policy function $\pi_\theta(q_t|s)$. After applying q_t to s and receiving user's answer to q_t (i.e., interacting with the environment, or sampling proper answer by reference during model training), the question chatbot receives a reward score r_{t+1} and the dialog state updates from s to s'. This quadruple $\langle s, q_t, r_{t+1}, s' \rangle$ is taken as an episode in the reinforcement learning process.The long-time reward R_t of time-step t is traditionally defined to be $R_t = \sum_{k=0}^{T-t-1} \gamma^k r_{t+k+1}$.

In our LTE framework, a dialog state s_t keeps track of time t's confidences (i.e., probabilities) of candidate products $\{p^i_m\}$ to be sent to the user. Specifically, $s_t \in \mathbb{R}^M$, $\forall s_{t,m} \geq 0$ and $\sum_{m=1}^{M} s_{t,m} = 1$. Here, $s_{t,m}$ denotes the confidence that product p^i_m is user's prefer at time-step t. Initially, s_0 can take the prior distribution of candidate products as we described in the entropy-based question ranking method.

Given the product set $P_i = \{p^i_m\}$ and the question set $Q = \{q_n\}$, we compute the normalized confidence of user's answer over the optional candidates attached in each question q_n. That is, the transition of dialog state is defined as:

$$s_{t+1} = s_t \odot \beta.$$

Here, $\odot$ is the dot product operator, β depends on the user's answer (selection) x_t to the question q_t (with a index of n_t in the question set $\{q_n\}$) which is selected by the question chatbot at time-step t. We define $\beta = [Y^{\{l\}}_{1,n_t}, ..., Y^{\{l\}}_{M,n_t}]$ when user selected option(s) $\{l\} \subset [1, ..., L_{n_t}]$ for current question q_t and $Y^{\{l\}}_{m,n_t} = \sum_{l \in \{l\}} Y^l_{m,n}$ takes a similar definition in Equation 2. Through this way, the confidence $s_{t,m}$ of a candidate product p^i_m is updated to $s_{t+1,m}$ based on user's answer $\{l\}$ to the selected question q_t at time-step t.

Algorithm 1: Joint training of LTE-RewardNet R, policy π_θ, and value net V

$\mathcal{M} \leftarrow \phi$ ▹ initialize episode memory
randomly initialize R, V, and π_θ
for *1* ***to*** *Z (epoch number)* **do**
  $p^i_m \leftarrow$ weighted sampling from $\{p^i_m\}$, $S_1 \leftarrow \{\}$, $S_2 \leftarrow \{\}$
  for t ***in*** $[1, T]$ **do**
    $q_t \leftarrow \pi_\theta(\cdot|s_{t-1})$ ▹ select from available questions
    $s_t \leftarrow s_{t-1} \odot \beta$ ▹ environment updating
    $(s_t, \mathcal{Q}_t, \mathcal{L}_t) \rightarrow S_1$ **if** *argmax*(s_t) == p^i_m **then**
      break ▹ find product and stop current session
  $r_T \leftarrow \kappa$ for a win or $-\kappa$ for a loss ▹ win means *argmax*(s_t) == p^i_m
  for $(s_t, \mathcal{Q}_t, \mathcal{L}_t)$ ***in*** S_1 **do**
    $r_{t+1} \leftarrow R(\langle s_t, \mathcal{Q}_t, \mathcal{L}_t, p^i_m \rangle)$ ▹ compute reward
    $(s_t, \mathcal{Q}_t, \mathcal{L}_t, r_{t+1}) \rightarrow S_2$
  for $(s_t, \mathcal{Q}_t, \mathcal{L}_t, r_{t+1})$ ***in*** S_2 **do**
    $r'_{t+1} \leftarrow \text{sigmoid}(\sum_{k=0}^{T-t-1} \gamma^k r_{t+k+1})$ ▹ accumulated reward from the future
    update V by (s_t, r'_{t+1})
    $v_{t+1} \leftarrow V(s_t)$
    $(s_t, \mathcal{Q}_t, \mathcal{L}_t, p^i_m, r'_{t+1} - v_{t+1}) \rightarrow \mathcal{M}$
    if $|\mathcal{M}| > K_1$ **then**
      update R by mini-batches in $\mathcal{M}$
      update π_θ by mini-batches in $\mathcal{M}$

4.2 LTE RewardNet

In order to allow our policy-based reinforcement learning algorithm to (1) taking explicitly former asked questions as a precondition for the next question predicting, and (2) utilizing users' historical selections for next question ranking, we propose a neural network based *LTE-RewardNet* which takes a quadruple $\langle s_t, \mathcal{Q}_t, \mathcal{L}_t, p^i_m \rangle$ as input and produces the next step reward r_{t+1}. Our LTE-RewardNet is a MLP with sigmoid output to learn the appropriate immediate rewards during training. Algorithm 1 shows the training details.

In particular, similar to word embeddings, we embed (with the size of 50, respectively) the questions $\mathcal{Q}_t$ and their corresponding answers $\mathcal{L}_t$. The embedded vectors are concatenated together with s_t to train the reward net R with a squared difference loss function. The reward net is further employed for training the policy function for question ranking for constructing our question chatbot. The policy network is trained using REINFORCE algorithm [10] under cross-entropy loss function.

Besides, we introduce a value network that scores the goodness of current state s_t. The value network is used to estimate how *good* is current state by itself to be chosen in an episode. The value network uses squared difference loss function, taking accumulated reward r'_{t+1} as the reference score (refer to Line 16). After updating, the new estimated state score v_{t+1} is subtracted from r'_{t+1} (Line 18) for updating R and π_θ, respectively.

There are four *for* loops in Algorithm 1: the outside one (Lines 3 to 21) controls the number of epochs Z, the second one (Lines 5 to 9) apply the policy network for available question selecting (we limit that one question can be used only once during one dialog session) and update the dialog state (environment) consequently. The results are stored in a set S_1 for next step usages. The third loop (Lines 11 to 13) applies the reward net R to obtain the immediate reward. The final loop (Lines 14 to 21) is to update the parameters in the policy and reward networks through picking mini-batches from the episode memory $\mathcal{M}$.

During testing, the $t+1$ step "information gain" for all products $\{p^i_m\}$ caused by user's answer a to current question q_{n_0} is:

$$g_{t+1}(\{p^i_m\}, q_{n_0}, a) = s_{t+1} - s_t. \quad (5)$$

5 RELATED WORK

In [14], reinforcement learning algorithms was used for ranking 20 questions for famous person guessing, in which they treated the game as a task-oriented dialog system and applied hard-threshold (yes or no) guessing with an assumption that the player supply only explicit answers. Our work can be considered as a *reverse* of completing tasks for dialog system developing through a style of 20 questions game process. 20 questions game was applied in [2] to knowledge acquisition to implicitly setup some questions to the user under a condition that the target person is already ranked with significantly high probability. The answer to these questions are unknown yet to the target person in the knowledge graph. A policy-based deep reinforcement learning algorithm was proposed in [4] for famous person guessing. The applications described in these three papers are all famous person guessing and they all used reinforcement learning algorithms for next question selecting.

Besides the 20-question research topics, there are much more related work regarding slot filtering for task-oriented dialog systems, from corpora to algorithms and further to systems. A multi-domain wizard-of-oz dataset with 10k dialog sessions (as used in this paper) has been released in [1] together with a reporting of a set of benchmark results of belief tracking, dialogue acting and response generating. A direct comparison to slot-filtering methods is bias since they mainly focus on the natural language understanding part by using machine learning approaches for next slot predicting. Our motivation is centralized on the question ranking part by employing policy and reward networks for learning the implicit constraints among questions conditioned by users' dynamic answers.

In addition, an end-to-end task-oriented dialog system was proposed in [7] that is trained using existing dialog sessions to supply multiple questions in one turn. The AirDialogue travel/flight dataset which contains 400k goal-oriented conversations was presented in [9], state-of-the-art dialogue models on it can only achieve a scaled score of 0.22 which is significantly lower than human scores of 0.94. This also reflects that current dialogue models are still fragile for novel domains.

	Harry Potter	general gifts
# questions	20	47
# products	82	500
avg. options / question	3.4	3.9
avg. linked # questions / product	15.5	6.1

Table 1: Statistics of products and questions in the "Harry Potter" and general gift categories.

In comparison, we argue that our LTE framework is less-training-data dependent, scalable to new domains with less human costs and closer to real-world usage in terms of both low usage threshold and keeping a high accuracy. Also, note that, we focus on the question ranking algorithms in this paper and then try to collect "information gains" from the interactive gaming processes for next step recommendation reason construction. Currently, the recommendation reasons are constructed in a rather simple way. It will be interesting to further develop deep generation models conditioned on the textual information of the questions, their answers from the user, the name/descriptions of the recommended products, and in particular the "information gains".

In addition, the *system ask - user respond* (SAUR) paradigm proposed in [13] is one pioneer work towards conversational search and recommendation. They utilized a multi-memory network architecture to keep track of user's initial request and the historical questions and answers (in one session or from personalized user profile) to *reasoning out* the ranking of the next candidate question and the retrieving of the next candidate products. The reasoning process is empowered by both sequential modeling and attention mechanisms. Our learning-to-explain framework follows the line of conversational searching and recommending. One major difference is that our "questions" are attached with multiple selectable options which making the "conversation" process to be more alike available information based selection. It will be interesting to try to combine our approach into this SAUR paradigm: the chatbot still gives options of each question and accepts natural language based answers as well for short-cutting of user's intention determination and for proper product retrieving.

6 EXPERIMENTS

6.1 Setup and Results for Gift Recommendation

We testify our LTE framework for gift recommendation, one gift set specially related to "Harry Potter" and the other gift set is more general oriented. Table 1 illustrates some statistical information of these two datasets. The questions are constructed in a manual way and the linking of the options in the questions to the product candidates are constructed manually as well. The linking of options to products is quite dense in "Harry Potter" category of 15.5, i.e., more than 75% of questions are averagely linked to each product. On the other hand, the linking for the general gift domain is rather sparse, only 13% (=6.1/47) of questions are linked to each product on average. We have shown examples of running gift recommendation in Figure 1, under the entropy-based question ranking method.

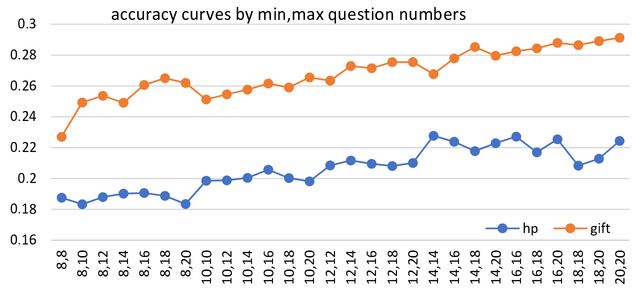

Figure 4: Self-play accuracies for Harry Potter domain (hp) and general gift domain (gift), using entropy-based method.

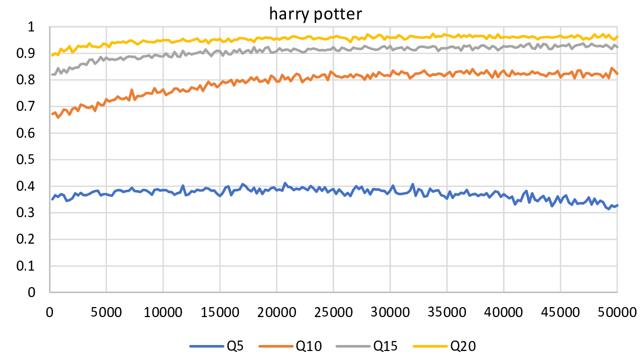

Figure 5: Self-play accuracies for Harry Potter domain (hp) , using reinforcement learning method.

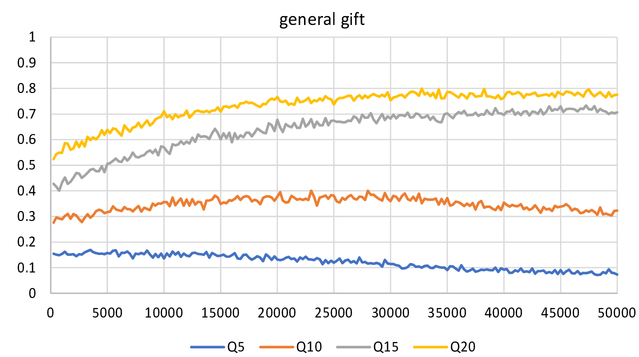

Figure 6: Self-play accuracies for general gift domain (gift), using reinforcement learning method.

We use self-playing style idea in Algorithm 1 to construct episodes for reinforcement training of the reward, value, and policy networks. A similar idea of first sampling a product (Line 4 in Algorithm 1) and then running the process by sampling answers from reference answers (Line 7 in Algorithm 1) can be employed for accuracy evaluating. After the terminate condition ($t = T$ or $argmax(s_t) == p_m^i$ in Line 5 and 8 in Algorithm 1) is matched, whenever the recommended product matches the initially selected product, we record it as a successful recommendation (Line 10 win means that final $argmax(s_t) == p_m^i$) and a failed recommendation otherwise. Both entropy based method and reinforcement learning methods can be employed for running the process to finally yield recommendation products and their information gains for recommendation reason construction.

In our policy network and reward network, we employ MLP with three hidden layers with identical hidden sizes of 1,000. We use the ADAM optimizer [5] with learning rate of 1e-3 for policy network and 1e-2 for the reward network. We select the discount factor γ to be 0.99 for computing long-term rewards. We train the multi-domain question chatbots with 50,000 epochs. Each evaluation of the question chatbot records the chatbot's performance with a greedy policy for 2,000 independent episodes in which 2,000 target products are weighted sampled using $w(p_m^i)$. We follow [1] to construct question sequences for each dialog session to be used for pre-training our LTE framework.

Figure 4 depicts the accuracy curves of Harry Potter category and general gift category, using entropy-based question ranking method. One recommendation evaluation metrics is that, we prefer using as few questions as possible for completing a rather high-accuracy recommendation. We range over two parameters, the minimum and maximum number of questions allowed in one recommendation session. In the figure, we observe that for the Harry Potter category, the accuracy tends to be stable at around 14 questions. For the general gift domain, more questions will yield higher accuracy. These aligns with Table 1, since the number of questions and products are more than that of Harry Potter category. However, note that the final accuracies are only around 22% of Harry Potter category and 30% of general gift domain. These results are significantly worse than the reinforcement learning approach as we will show the experiment results below, in Figure 5 and 6.

In Figure 5, the accuracies are significantly better than the entropy-based method. Noting that even we only use 5 questions, the result of more than 30% is better than that of using 20 questions in the entropy-based method. The accuracy doubles as we use maximum 10 questions, instead of 5 questions. After 50,000 epochs, the accuracy is more than 80%. When we employ all the 20 questions, the accuracies starts from 89.25% and finally reached 96.25%. In real applications, we only adopt a maximum of 10 questions for a trade-off of quick product recommendation and a rather high-accuracy.

We observe nearly similar tendencies in Figure 6. The accuracies of using 20 questions finally reached 77.5%, significantly better than 32.3% of using 10 questions and 70.6% of using 15 questions. Also note that, when we only utilize 5 questions maximum in a conversation session, the accuracies are only 15.45% at the beginning of the training and further drop to 7.35%. These reflects that when we employ too few questions in one episode, the reinforcement learning method is not stable and not guarantee to learn a good model. These results are also align with the larger number of products and questions than the Harry Potter category, as illustrated in Table 1.

Table 2 compares the entropy-based method and the reinforcement learning algorithm on categories of "Harry Potter" and "general gift". For the reinforcement learning method, we evaluated at epochs being 50,000 and on the test set of performing 2,000 sampling and self-play sessions. For the entropy-based method, we evaluate the categories using exacting 20 questions.

	entropy	reinforcement learning R
harry potter	22.5	95.7
general gifts	29.2	77.4

Table 2: Comparison of succeed rates (%) evaluated by self-play, across "Harry Potter" and "general gift" categories and method configurations.

U: Help me find a gift.
Q1: Sure, who are you shopping for?
1.family; 2.significant other; 3.coworker; 4.myself
U: 1
Q2: How old are they? 1.kid; 2.teen; 3.adult; 4.retirement-age
U: 4
Q3: How much do you want to spend? 1.$; 2.$$; 3.$$$
U: 3
Q4: Which chore do they hate the most?
1.cleaning; 2.cooking; 3.laundry; 4.mowing the lawn
U: 2
Q5: If they were on a desert island,
which thing would they most want to have with them?
1.soccer ball; 2.a bed; 3.fishing pole; 4.nintendo; 5.ereader
U: 3
Q6: Where do they live? 1.city; 2.suburbs; 3.country
U: 1
Q7: What's their preferred method of exercise?
1.netflix marathon; 2.triathlon; 3.yoga; 4.swimming
U: 3
Q8: Do they pay extra for guac? 1.always; 2.no way
U: 2
Q9: When things are broken, what do they do?
1.hire a professional; 2.DIY it; 3.leave it broken
U: 2
Q10: When things are broken, what do they do?
1.hire a professional; 2.DIY it; 3.leave it broken
U: 2
R: I would like to recommend you:
Noise-Cancelling Headphones,
guided by the selection of "fishing pole" in Q5.

Figure 7: One real examples under our LTE framework for general gift recommendation, employing the reinforcement learning method. U = user, Q and R are respectively questions and recommendations with reasons from the chatbot.

For intuitive expression of general gift recommendation, we show a real example in Figure 7. The questions are aiming at instantly collecting information from gift-receivers that are hidden in current user's mind, such as social network group, age range, sports, hobbies and so on. In addition, budget information is also asked. The chatbot finally recommended the user "Noise-Cancelling Headphones" and user's answer of "fishing pole" contributed the most. In addition, we checked the list of "information gain" and find that user's answer of "yoga" contributed the second after "fishing pole". Both of these hints collected from the user are linked with the final product. Thus, it is reasonable to extend the final recommendation reason to include more than one pair of question-answer. As far as the instant conversation session-based "information gains" are collected by Equation 4 and 5, the consequent recommendation reason generation can be rich and colorful.

	Harry Potter	General Gift
% finished one session	0.6097=567/930	0.3144=305/970
% ordered	0.4550=258/567	0.2590=79/305
% ordered without reasons	0.4574=118/258	0.4304=34/79
% ordered with reasons	0.5426=140/258	0.5696=45/79

Table 3: Comparison of real-world succeed ratios, across "Harry Potter" and "general gift" categories under reinforcement learning algorithm.

Gift recommendation is good at making a explainable recommendation based on "mind-reading". However, one difficulty is related to manual question construction and the linking of questions to a given product list. We are wondering if our LTE framework can be adapted to large-scale and multi-domain product recommendation, without the preparing of manual data.

Finally, we are wondering how is our LTE framework works in real-world facing real-world users. We respectively released our "Harry Potter" and "general gift" recommendation through a million-user chatbot. We compare the following three metrics:

- the ratio of users that completed a conversation session;
- the ratio of users that finally bought the products recommended;
- the difference of between attaching explaining reasons and without it.

Table 3 shows real-world evaluation results of these two categories. For "Harry Potter" domain, more than 60% of users finished completely one recommendation session and the ratio of ordering after the session is 45.50%. With recommendation reasons, the ordering ratio improved 54.26% - 45.74% = 8.52%. These reflect that the proposed explainable LTE framework performs efficiently in real-world application. Also, for the general gift recommendation, the overall finishing ratio is 31.44%, possibly due to that the topics and questions related to general gifts are less attractive than the strong topic of "Harry Potter". With recommendation reasons, we obtained 13.92% improvements. An extension to larger user coverage is our future work.

6.2 Setup and Results for Multi-WOZ

In order to testify our LTE framework using automatically constructed questions, we choose the Multi-WOZ corpus [1]. The corpus is aiming at training slot filtering models for task-oriented multi-domain dialog systems. We pick five domains *attraction*, *hospital*, *hotel*, *restaurant*, and *train* since the number of products and their attribute ontology are in a reasonable size.

We collect attributes of the products and then use pattern based method (where the slots of questions are underlined in the hereafter examples) to automatically generate questions based on these attributes. For the attraction category, questions are alike *shall the address be located in the grafton centre?* and *are you looking for a*

category	# product	# question (combined)	# avg. *yes*
attraction	79	119 (2)	4.5
hospital	66	88 (0)	2.7
hotel	33	43 (6)	7.2
restaurant	110	146 (2)	4.4
train	2,828	31 (5)	7.0

Table 4: Statistics of products and questions in the five categories of the Multi-WOZ corpus.

	entropy	R_0	R	R+ dialog sessions
attraction	44.7	52.7	69.9	-
hospital	70.5	82.7	93.1	-
hotel	72.0	75.4	88.9	91.9
restaurant	43.3	37.5	61.1	63.9
train	59.2	64.2	73.4	-

Table 5: Comparison of succeed rates (%) evaluated by self-play (at epochs = 50,000; on the test set of performing 2,000 sampling and self-play sessions), across categories and method configurations.

museum?. For the hospital category, we extract keywords from the names of the hospitals and example questions are alike *shall the hospital related to eye?*, *general?*, or *oncology?*. Especially for this hospital domain, our LTE framework of supporting natural language requirements as inputs is essential: users do not have rich time for answering the questions one by one and we should extract keywords from their queries directly.

For the hotel domain, we extract keywords included in addresses to construct questions. Also, the area, stars, type, internet, parking and price ranges are included for single and combined question building. Example questions are alike *do you prefer guesthouse or hotel?*, *do you prefer a hotel that is cheap, moderate or expensive?*. For the restaurant domain, we take keywords in addresses, areas, price range and food-styles for question constructing. Examples are alike *do you prefer european, asian oriental, african, or north american?* or *how about near 108 Regent Street City Centre?*. Note that the address related questions should (and can) be ranked based on relative distance of between real-world users and the candidate restaurants in a style of location-based service. Finally, for the train domain, we allow the users to select the leaving from and arrive by time points, departures and destinations, and price ranges.

We make use of the frequencies of the products appearing in the *data.json* file of Multi-WOZ corpus as the initial weights $w(p^i_m)$ of each p^i_m. In addition, we also initialize $f^l_{m,n}$ as used in Equation 2 by the frequency that user selected option l in the dialog sessions for question q_n and the user finally booked product q^i_m. $f^l_{m,n}$ is employed in our entropy-based baseline method. The sequences of slots included in the dialog sessions are also included for training the policy π_θ for question ranking in Algorithm 1. We compare three configurations: the entropy based method, the LTE-RewardNet method and the LTE-RewardNet method pre-trained by dialog sessions under the self-play configuration. The entropy based method has been reported to yield more than 90% of accuracy for 20-question-games such as guessing famous people [11]. The results are shown in Table 5. We could pre-train the LTE-RewardNet for two categories of hotel and restaurant since the other three categories include too few dialog sessions that both match the target products and the questions (Table 4).

We use R to denote the LTE model in which the reward net is updated jointly with the policy model. In contrast, we use R_0 to denote the non-joint model which only returns rewards alike the entropy based method. That is, in Line 20 of Algorithm 1, the reward network is not trainable and not updated.

Since the number of candidate products and questions are significantly different among the categories, a direct accuracy across categories is not meaningful. We mainly focus on the relative improvements of introducing reinforcement learning methods for long-distance reward modeling and how existing dialog sessions can bring additional benefits. The major observation from Table 5 is that our proposed LTE-RewardNet R significantly outperforms the entropy based method with relative improvements of from 14.2% to 25.2%. Comparing R_0 with R, we observe that R_0 is still significantly better (except the restaurant category) then the entropy-based method. Yet, R outperforms R_0 significantly showing us that the joint updating method as expressed in Algorithm 1 is essential for capture long-term rewards.

Table 5 also reflects that the non-parametric entropy-based method without training is rather less suitable for these task-oriented domains in which the number of questions is small, the constraints among questions are less observable, and the product-question matrices are rather sparse. In the entropy-based baseline method, we found that individual questions will continue be sent to the user to seek answers since no internal latent relations are leveraged or modeled for speeding up.

For the hotel and restaurant domains, when we further include dialog sessions with question sequences for pre-training the LTE-RewardNet, we further obtained 2.8% to 3.0% improvements. This reflects that (1) additional dialog sessions can bring benefits to train a better question chatbot and (2) making use of 10k dialog sessions from Multi-WOZ corpus only brings us a quite limit improvements. Thus, these comparison confirms the effectiveness of starting from zero-dialog sessions and still train a rather robust question chatbot. Especially for domains of hospital and hotel, our accuracy excesses 90% which is supposed to be applicable for real-world product suggestion.

Figure 8 shows the learning curve of accuracy conditioned on epochs (maximum 50,000) for hotel and restaurant domains under three configurations. The entropy-based method is drawn as well for a reference. For the restaurant domain, the entropy based method outperforms the reinforcement methods before 15,000 epochs, after that, the reinforcement learning method significantly improved with more learning epochs. For the hotel domain, the joint method R achieves comparable accuracy even at the beginning guided by the product-question matrix. In these two domains, we find that R is robust no matter there are dialog sessions for pre-training or not.

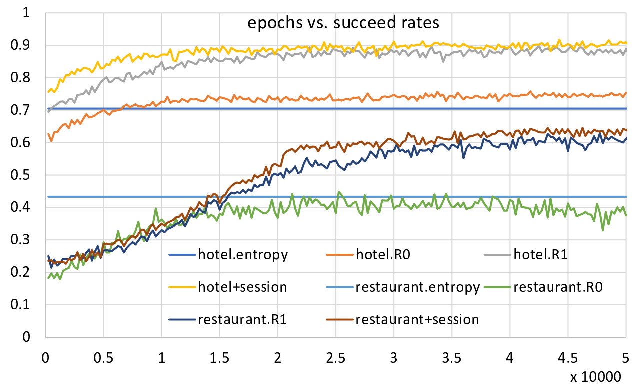

Figure 8: LTE's policy-learning curve of hotel/restaurant accuracy conditioned on epochs. R0=R_0, R1=R.

A: Hi there, how can I help you? 1.hotel, 2.hospital,
3.restaurant, 4.train, 5.attraction
U: 1
Q1: Okay, do you have a specific area you want to stay in?
1.centre, 2.east, 3.west, 4.south, 5.north
U: 1
Q2: Which price range do you prefer?
1.economic, 2.business, 3.luxury
U: 1
Q3: How many people are you booking for?
U: 6.
Q4: How many rooms do you prefer to live?
U: 3.
Q5: Do you need parking?
U: yes, 2 cars (or, no).
Q6: Do you prefer using free wifi?
U: yes (or, no).
Q7: How many nights would you like to stay?
U: 3.
Q8: Which date would you like to check in?
(show a calendar with constrained available dates)
U: November 13

R: I find the following three hotels for you:
1.hotel X, 2.hotel Y, 3.hotel Z.
The recommendation is guided by the selection of "6" people for Q3, and "3" rooms for Q4.
U: The first one please.
A: Booking was successful. ...

Figure 9: Multi-WOZ's hotel booking under our LTE framework, using reinforcement learning algorithm.

Figure 9 illustrates an example of hotel booking under our LTE framework using reinforcement learning algorithm. The algorithm learned to ask Q4 of *How many rooms do you prefer to live?* and consequently Q5 of *Do you need parking?* at the condition that there will be 6 people for this booking (Q3). The recommendation reason here is mainly related to the number of people and the number of rooms for booking. Note that, there are totally three candidate hotels (with URLs) for the user to have a detailed check to make a selection. These are aiming at supplying user-friendly recommendation experience.

7 CONCLUSION

We have described an end-to-end LTE framework for *question chatbot* constructing aiming at explainable product recommendation. One "end" is product-attribute tables and the other "end" is multi-domain question chatbots. Our framework is suitable for situations where (1) well-designed questions are manually prepared yet no available real-world dialog data (for gift recommendation), and (2) only product-attribute tables are available and there are no or quite few real-world dialog sessions belong to that target domain (for Multi-WOZ data). By constructing a question chatbot, we switch the terminal users of from requirement suppliers to simple decision makers for both time saving and usage-threshold lowering. Furthermore, with the recording of "information gain" yielded by question-answer pairs to candidate products, instant recommendation reasons are constructed.

In order to construct question chatbots through our LTE framework, we propose a reward network to keep track of long-term rewards. Through a joint training of the reward network and the question ranking policy network, we construct multi-domain chatbots without using any dialog data. Compared with entropy-based non-parametric method, our policy-based reinforcement learning algorithm is robust and can both leverage available dialog sessions and existing single-sentence slot filtering methods. Experiments on specific topic (Harry Potter) and general category gift recommendation gave us accuracies of 96.25% and 77.5%, repectively. Experiments on a large-scale multi-domain corpus [1] show accuracy of more than 90% for hospital/hotel domains.

Extremely, 20 questions of one conversation session for recommendation can cover a maximum of $2^{20} = 1,048,576$ products. One extension of our framework is to apply it to product recommendations in e-commerce fields in which product-attribute tables are rich yet are short of real-world dialog sessions for slot filtering of long-tail domains. For example, the Amazon product data [6] has 9 million products, 237 million links (such as 'users who bought X also bought Y') and 144 million reviews. We take this as our future work.

Our framework can further be combined with the *system ask - user respond* paradigm [13] so that (1) the user's answers can be eight a selection of options included in questions or directly natural language sentences, and (2) the memory based reasoning architecture can be further empowered by the reinforcement learning algorithms to perform self-playing for enhance the final models for conversational retrieving and recommending.

REFERENCES

[1] Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. 2018. MultiWOZ - A Large-Scale Multi-Domain Wizard-of-Oz Dataset for Task-Oriented Dialogue Modelling. In *Proceedings of EMNLP*. 5016–5026. http://aclweb.org/anthology/D18-1547

[2] Yihong Chen, Bei Chen, Xuguang Duan, Jian-Guang Lou, Yue Wang, Wenwu Zhu, and Yong Cao. 2018. Learning-to-Ask: Knowledge Acquisition via 20 Questions. In *Proceedings of ACM SIGKDD*. ACM, 1216–1225. https://doi.org/10.1145/3219819.3220047

[3] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. *CoRR* abs/1708.05031 (2017). arXiv:1708.05031 http://arxiv.org/abs/1708.05031
[4] Huang Hu, Xianchao Wu, Bingfeng Luo, Chongyang Tao, Can Xu, wei wu, and Zhan Chen. 2018. Playing 20 Question Game with Policy-Based Reinforcement Learning. In *Proceedings of EMNLP*. 3233–3242. http://aclweb.org/anthology/D18-1361
[5] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014). arXiv:1412.6980 http://arxiv.org/abs/1412.6980
[6] Julian McAuley, Rahul Pandey, and Jure Leskovec. 2015. Inferring Networks of Substitutable and Complementary Products. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '15)*. ACM, New York, NY, USA, 785–794. https://doi.org/10.1145/2783258.2783381
[7] Janarthanan Rajendran, Jatin Ganhotra, Satinder Singh, and Lazaros Polymenakos. 2018. Learning End-to-End Goal-Oriented Dialog with Multiple Answers. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, 3834–3843. https://www.aclweb.org/anthology/D18-1418
[8] Heung-Yeung Shum, Xiaodong He, and Di Li. 2018. From Eliza to XiaoIce: Challenges and Opportunities with Social Chatbots. *CoRR* abs/1801.01957 (2018). arXiv:1801.01957 http://arxiv.org/abs/1801.01957
[9] Wei Wei, Quoc Le, Andrew Dai, and Jia Li. 2018. AirDialogue: An Environment for Goal-Oriented Dialogue Research. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, 3844–3854. https://www.aclweb.org/anthology/D18-1419
[10] Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Machine Learning*. 229–256.
[11] Xianchao Wu, Huang Hu, Momo Klyen, Kyohei Tomita, and Zhan Chen. 2018. Q20: Rinna Riddles Your Mind by Asking 20 Questions. In *Proceedings of (Japan) NLP*. Okayama, 1312–1315. http://anlp.jp/proceedings/annual_meeting/2018/pdf_dir/E7-4.pdf
[12] Zhao Yan, Nan Duan, Peng Chen, Ming Zhou, Jianshe Zhou, and Zhoujun Li. 2017. Building Task-Oriented Dialogue Systems for Online Shopping. In *AAAI 2017 Conference on Artificial Intelligence*. https://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14261
[13] Yongfeng Zhang and Xu Chen. 2018. Explainable Recommendation: A Survey and New Perspectives. *CoRR* abs/1804.11192 (2018). arXiv:1804.11192 http://arxiv.org/abs/1804.11192
[14] Tiancheng Zhao and Maxine Eskenazi. 2016. Towards End-to-End Learning for Dialog State Tracking and Management using Deep Reinforcement Learning. In *Proceedings of SIGDD*. Los Angeles, 1–10. http://www.aclweb.org/anthology/W16-3601
[15] Li Zhou, Jianfeng Gao, Di Li, and Heung-Yeung Shum. 2018. The Design and Implementation of XiaoIce, an Empathetic Social Chatbot. *CoRR* abs/1812.08989 (2018). arXiv:1812.08989 http://arxiv.org/abs/1812.08989