Towards Conversational Search and Recommendation: System Ask, User Respond

Yongfeng Zhang[1], Xu Chen[2], Qingyao Ai[3], Liu Yang[3], W. Bruce Croft[3]
[1]Department of Computer Science, Rutgers University, New Brunswick, NJ 08901
[2]School of Software, Tsinghua University, Beijing, China 100084
[3]College of Information and Computer Sciences, University of Massachusetts Amherst, MA 01003
yongfeng.zhang@rutgers.edu,xu-ch14@mails.tsinghua.edu.cn,{aiqy,lyang,croft}@cs.umass.edu

ABSTRACT

Conversational search and recommendation based on user-system dialogs exhibit major differences from conventional search and recommendation tasks in that 1) the user and system can interact for multiple semantically coherent rounds on a task through natural language dialog, and 2) it becomes possible for the system to understand the user needs or to help users clarify their needs by asking appropriate questions from the users directly.

We believe the ability to ask questions so as to actively clarify the user needs is one of the most important advantages of conversational search and recommendation. In this paper, we propose and evaluate a unified conversational search/recommendation framework, in an attempt to make the research problem doable under a standard formalization. Specifically, we propose a *System Ask – User Respond* (SAUR) paradigm for conversational search, define the major components of the paradigm, and design a unified implementation of the framework for product search and recommendation in e-commerce. To accomplish this, we propose the Multi-Memory Network (MMN) architecture, which can be trained based on large-scale collections of user reviews in e-commerce. The system is capable of asking aspect-based questions in the right order so as to understand the user needs, while (personalized) search is conducted during the conversation, and results are provided when the system feels confident. Experiments on real-world user purchasing data verified the advantages of conversational search and recommendation against conventional search and recommendation algorithms in terms of standard evaluation measures such as NDCG.

KEYWORDS

Conversational Search; Conversational Recommendation; Product Search; Dialog Systems; Memory Networks; Personalized Agent

ACM Reference Format:
Yongfeng Zhang, Xu Chen, Qingyao Ai, Liu Yang, W. Bruce Croft. 2018. Towards Conversational Search and Recommendation:, System Ask, User Respond. In *The 27th ACM International Conference on Information and Knowledge Management (CIKM '18), October 22–26, 2018, Torino, Italy*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3269206.3271776

CIKM '18, October 22–26, 2018, Torino, Italy

ACM ISBN 978-1-4503-6014-2/18/10…$15.00
https://doi.org/10.1145/3269206.3271776

1 INTRODUCTION

Among the many techniques that compose an intelligent Web, a *Conversational System* (such as Google Now, Apple Siri, and Microsoft Cortana) is one that serves as the direct interactive portal for end-users, which is expected to revolutionize human-computer interaction. With recent progress on NLP and IoT, such systems have also been deployed as physical devices such as Amazon Echo, opening up more opportunities for applications in a smart home.

Due to users' constant need to look for information to support both work and daily life, a *Conversational Search System* will be one of the key techniques. Conversational search aims at finding or recommending the most relevant information (e.g., web pages, answers, movies, products) for users based on textual- or spoken-dialogs, through which users can communicate with the system more efficiently using natural language conversations.

Conversational search and recommendation are technically very similar in e-commerce settings. Figure 1 shows an example of conversational search for product search/recommendation. In this task, user-system interactions can be classified into three stages, i.e., *initiation*, *conversation*, and *display*. In the first stage, user initiates a conversation with an initial request, e.g., by telling the system what category of product she is looking for; in the second stage, the system asks the user about her preferences on certain product aspects, estimates user needs based on the feedback, and conducts search during the conversation. When the system feels confident about the results, they will be displayed to the user in the third stage. However, the second and third stages could be repeated if the displayed results do not satisfy the user needs.

One may see that every operation during the conversation requires carefully designed models, including question formulation, user need estimation, and search/recommendation. As explained later, we develop a unified framework and provide one of its model implementations for conversational search in the product domain. It should be noted that the aspects are automatically extracted, and the system does not simply ask about the aspects in a random order. Instead, it determines which aspect to ask at each time with a carefully trained strategy, so that the system can always ask the most important question to improve its confidence about user needs and search results, thus the conversation can be kept as short as possible, and the user needs can be satisfied as soon as possible.

Conversational search is closely related to several other research topics such as dialog systems, traditional web search, and faceted search. Recent conversational search systems are well integrated with state-of-the-art dialog system models, and by focusing on the search task, the system takes advantage of conversations to understand the user needs accurately. Conversational search also

exhibits several advantages compared with other search paradigms. For example, faceted search usually requires the existence of structured knowledge, and traditional web search usually relies on query suggestion – or relies on the user himself to reformulate the query – so as to understand the user needs. By actively asking questions, conversational search has the advantage of understanding user needs more efficiently, but this also brings about challenges related to asking the right question at the right time, as well as inferring user preferences from unstructured responses automatically.

Fortunately, recent advances on representation learning, dialog systems, neural information retrieval, and neural logical inference have made it possible for us to tackle with these challenges. In this work, we propose a System Ask – User Respond (SAUR) paradigm for conversational search and recommendation. We then propose a Multi-Memory Network (MMN) architecture as an implementation of the paradigm, which conducts question prediction and search in a parallel manner. The architecture is further generalized into a personalized version (PMMN) for personalized conversational search with individual users. Experiments on real-world Amazon purchase datasets verified the effectiveness of our approach.

In the following, Section 2 reviews related work, Section 3 presents the problem formalization, Section 4 introduces the (P)MMN models, and Section 5 presents the experimental results. Concludes and discussions of future work are presented in Section 6.

2 RELATED WORK

2.1 Conversational Search and Recommendation

Conversational search and recommendation are relatively new research topics, but the basic concepts date back to some of the most early works in the community. For example, Croft and Thompson [9] designed I^{3}R (Intelligent Intermediary for Information Retrieval) – an expert intermediary system that takes activities to communicate with the user during a search session similar to what is done by a human intermediary; Belkin et al [3] designed the MERIT system – an *interactive information retrieval* system that used script-based conversational interaction for effective search.

With the emerging of various conversational devices, and the progress of neural NLP research – especially on natural language dialog systems – conversational search based on direct user-system dialoging has achieved new attention in the recent years. Radlinski and Craswell [26] proposed a theoretical framework for conversational search, which described some basic design philosophies for conversational search systems. Kenter and de Rijke [16] formalized conversational search as a machine reading task for question answering. Yang et al [41, 42] conducted next question prediction and response ranking in conversations. Spina and Trippas et al studied the ways of presenting search results over speech-only channels [31] and transcribing the spoken search recordings [34] to support conversational search. Christakopoulou et al [6] proposed an interactive recommendation protocol that collects like/dislike feedback from users to refine the recommendations.

Despite the extensive attention that conversational search and recommendation have received, our understanding of the nature of conversational search is still limited, and there lacks a conversational search paradigm to integrate the search task with recent (neural) NLP techniques – which is one of the goals of this work.

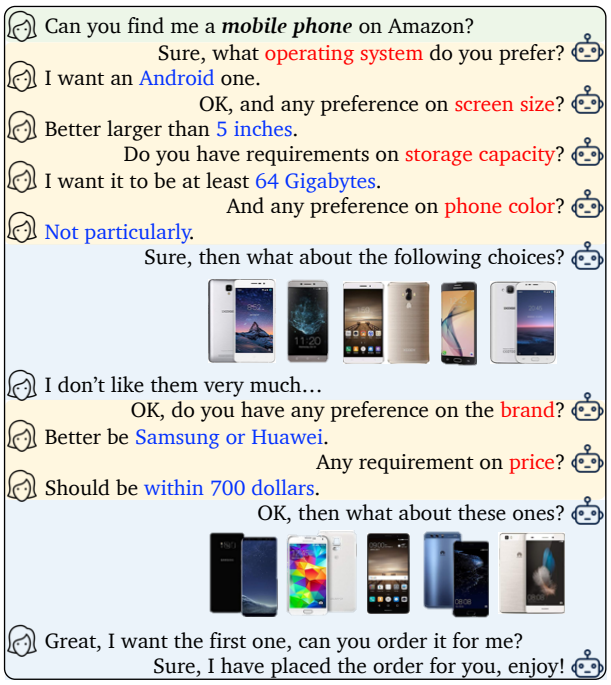

Figure 1: Example for conversational search in e-commerce product search or recommendation scenario (best in color).

One of the major factors affecting conversational search research is the lack of large-scale data for model training and analysis. Usually, data collected from volunteers (e.g., the MISC dataset [33]) is not sufficient to support (neural) model training. In this work, however, we propose to leverage large-scale user textual reviews to train practical conversational search models.

2.2 Product Search and Recommendation

There has been a long track of research efforts for product recommendation in e-commerce [27], and considerable work has been done on product search in e-commerce based on structured knowledge [20]. Despite their importance in e-commerce, searching with structured knowledge is not enough because both user requests and product descriptions can be in natural language, and developing structured knowledge bases for all products can be expensive. Duan et al [11, 12] proposed a probabilistic mixture model by analyzing product search logs, and proposed to extend a product database with language modeling to support conditional search on specifications. To bridge the language gap between product descriptions and user queries, Gysel et al [35] proposed a latent vector space model to map queries and products into the same latent space for product retrieval. Ai et al [2] noticed that the product search task can be very personalized, and different users may choose different products even on the same query. As a result, they further proposed a hierarchical embedding model for personalized product search.

2.3 Dialog Systems and Memory Networks

We also briefly review the research on dialog systems in NLP – a closely related task with conversational search and recommendation. Traditional approaches to dialog systems have mostly been

statistical methods, such as the partially observable Markov decision processes (POMDP-based) for dialog modeling [43]. Recently, neural approaches to dialog systems have attracted much attention in the NLP community [4, 14, 36], and several standard toy tasks for neural dialog modeling and evaluation have been proposed [10, 37].

With the ability of logical inference on textual inputs, Memory Network (MemNN) [13, 32, 38] has been one of the state-of-the-art approaches to dialog systems [4, 10, 19], as well as several other closely related tasks, including question answering [5, 17, 19, 40], language learning [39], and machine reading [24, 29], etc. Memory networks exhibit close alignment with the conversational search task – by learning word embeddings and using attention mechanisms to select important signals from query and item, it helps to alleviate vocabulary mismatch between user responses and machine knowledge, and to improve search performance by focusing on the important signals for the current conversation.

3 PROBLEM FORMALIZATION

3.1 A System Ask User Respond (SAUR) Paradigm

We believe a key advantage of conversational search/rec is that the system can ask questions from the users actively, so as to understand the user needs accurately, and to increase its confidence with the search results. Based on this philosophy, we design a conversational search/rec paradigm as shown in Figure 2.

After the user initiates a conversational search by providing an initial request (which happens only once in the flow), the system conducts search with the search module based on the request and the candidate items. If the system is not confident with the results, then it will generate a question to ask based on the question module, which also considers the user request and item representations.

After user gives a response to the question, the system enters the loop again, but now, the system not only considers the user's initial request for search and question generation, but also the newly collected question-answer pair, which contains new information about the user needs in this search task. In each of the following loops when the system is not confident, both the search and question generation modules will take initial request, candidate items, and all of the previously collected question-answer pairs into consideration.

Once the system is confident of the results in a certain loop, it will display the results to the user. According to the interactive interface used in practice, we may choose to display different numbers of results, e.g., to display the top-1 result in voice-based interface such as Amazon Echo, or top-5 results in visual interface such as Echo Show. The conversation will stop if the user is satisfied with the results, otherwise, the system will enter the loop again by asking a new question to understand the user needs better.

3.2 Notations and Statement of the Problem

Suppose the system provides conversational search service for M users $\mathcal{U} = \{u_1, u_2 \cdots u_M\}$ over N items $\mathcal{V} = \{v_1, v_2 \cdots v_N\}$. In this work, each item $v_j \in \mathcal{V}$ is a product in e-commerce, and each item is accompanied with a textual description T_j. Key notations used throughout the work are summarized in Table 1.

After user u_i purchases item v_j, the user will write a piece of textual review R_{ij} on the product to describe the objective aspects of the product and/or her subjective opinions on the aspects, where

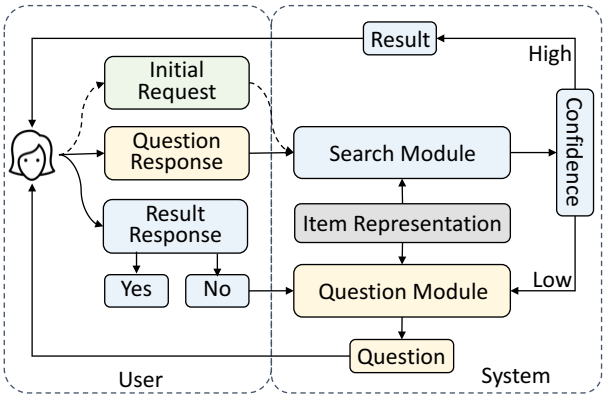

Figure 2: A workflow for conversational search/rec system.

the aspects could be *operating system, screen size*, etc, as in Figure 1. We transform a textual review into a question-answer sequence to simulate the conversation that resulted in this purchase behavior. Specifically, suppose the purchased item v_j is of category c, and the user commented on K product aspects in the review, which – according to the order they were mentioned – are $p_1, p_2 \cdots p_K$, and the values the user specified on these aspects in the review are $q_1, q_2 \cdots q_K$, respectively. We thus build the following conversation for this user-item purchase behavior,

$$\begin{aligned} u_i &\rightarrow Q_0 \mid Q_1 A_1, Q_2 A_2 \cdots Q_K A_K \mid v_j \\ &= Q_0(c) \mid Q_1(p_1) A_1(q_1) \cdots Q_K(p_K) A_K(q_K) \mid v_j \end{aligned} \tag{1}$$

where Q_0 is the initial request constructed from the product category c, $Q_k (1 \le k \le K)$ is the question asked by the system, which is constructed with the aspect p_k, and $A_k (1 \le k \le K)$ is the answer from user, which is the review sentence containing the aspect p_k and value q_k. Finally, item v_j is purchased by the user as the conclusion of the conversation. The system questions are constructed with predefined language patterns in practice, meaning that what the system needs to predict is only the aspect used to generate a question. For intuition, an example conversation generated from the Amazon *Electronics* review dataset is shown as follows.

> Conversation initiated by user `ID=AQGUDK0MSQ95L`
> U: Can you find me a `tablet` on Amazon?
> S: Sure, any requirement on the `network`?
> U: Built-in free wireless data network.
> S: Any preference on the `memory`?
> U: 2GB of internal memory as well as a microSD expansion slot for additional memory.
> S: Any preference on the `battery`?
> U: Battery is removable and user-replaceable.
> Result: Product `ID=1400532620`.

By constructing a conversation in this way, we are actually assuming that those aspects that are mentioned earlier in a review tend to be more important for the user on this item. As a result, we train a model that also generates questions about aspects preferred by the user earlier than other aspects.

It is worthwhile noting that training a practical (neural) conversational model requires large-scale training data, but such real-world conversation data seldom exists. Recent research in conversational

search [16], memory networks [32, 38], and dialog systems for machine reading [4, 10, 19, 37] adopted fully-synthetic data for model training. In this work, we move one step forward by constructing semi-synthetic data from reviews for model learning.

Based on the above notations and data, the conversational search and recommendation system aims at learning models for the following two key tasks:

Question Generation: Given the current and previous states of a conversation, generate the next question to ask. Specifically, a generative model is trained by maximizing the probability of each question in each of the training conversations:

$$\begin{aligned} &P(Q_{k+1}|Q_0, Q_1, A_1, Q_2, A_2 \cdots Q_k, A_k) \\ =&P(p_{k+1}|c, p_1, q_1, p_2, q_2 \cdots p_k, q_k),\ \forall 0 \le k < K \end{aligned} \tag{2}$$

where k enumerates from 0 because we generate a question even when we only have user's initial request Q_0.

Search and Ranking: Given the current and previous states of a conversation, generate a ranking list of items and the confidence score for each item. Specifically, a ranking model is trained by maximizing the probability of the purchased item v_j at each stage for each of the training conversations:

$$\begin{aligned} &P(v_j|Q_0, Q_1, A_1, Q_2, A_2 \cdots Q_k, A_k) \\ =&P(v_j|c, p_1, q_1, p_2, q_2 \cdots p_k, q_k),\ \forall 0 \le k \le K \end{aligned} \tag{3}$$

where k also enumerates from 0 because we do a search with only the initial request Q_0.

4 MULTI-MEMORY NETWORKS (MMN)

In this section, we propose an implementation of the conversational search/rec paradigm. Inspired by [17], we propose a Multi-Memory Network (MMN) architecture for conversational search (shown in Figure 3), which is a unified framework that integrates query/item representation learning and search/question module training into a single model. In this architecture, each item is represented as sentence embeddings based on its textual description, and the user's initial request – together with the currently collected information during the conversation – are used to reason over the items based on attention mechanism, which selects relevant signals to construct memory for search and question generation.

In the following, we introduce the item and query representations, as well as the memory, search, and question modules one by one, and then integrate them into a unified loss function for model learning. In the end, we further propose a Personalized version of MMN (i.e., PMMN) for conversational search and recommendation.

4.1 Item Representations

For a product v_j, we merge its product description and the textual reviews it received as its final textual description T_j. This helps the search module by enriching the system's knowledge about products and reducing the vocabulary mismatch between products and user queries; and also helps the question module to generate questions in languages that are familiar to the users.

To generate item representations for v_j, we insert an end-of-sentence token after each sentence of the description T_j, and then apply a gated recurrent unit (GRU) layer [7] through T_j. In the following, we use t to index the words and use τ to index the sentences of T_j, and the internal states of GRU is defined as,

$$\begin{aligned} \mathbf{z}_t &= \sigma(W_z \mathbf{x}_t + U_z \mathbf{h}_{t-1} + \mathbf{b}_z) \\ \mathbf{r}_t &= \sigma(W_r \mathbf{x}_t + U_r \mathbf{h}_{t-1} + \mathbf{b}_r) \\ \tilde{\mathbf{h}}_t &= \tanh(W_h \mathbf{x}_t + U_h(\mathbf{r}_t \circ \mathbf{h}_{t-1}) + \mathbf{b}_h) \\ \mathbf{h}_t &= \mathbf{z}_t \circ \mathbf{h}_{t-1} + (\mathbf{1} - \mathbf{z}_t) \circ \tilde{\mathbf{h}}_t \end{aligned} \tag{4}$$

where $\mathbf{x}_t, \mathbf{h}_t \in \mathbb{R}^{D\times 1}$ are the input and hidden state output of the network at time step t, and $W_z, W_r, W_h, U_z, U_r, U_h \in \mathbb{R}^{D\times D}$ as well as $\mathbf{b}_z, \mathbf{b}_r, \mathbf{b}_h \in \mathbb{R}^{D\times 1}$ are parameter matrices and bias vectors to be learned. In our case, we have $\mathbf{x}_t = L[w_t] = \mathbf{w}_t \in \mathbb{R}^{D\times 1}$, where L is the word embedding matrix to be learned, w_t is the t-th word of the input sequence, and $\mathbf{w}_t$ is its word embedding in L.

We abbreviate the above computation as $\mathbf{h}_t = GRU(\mathbf{w}_t, \mathbf{h}_{t-1})$, and adopt the hidden states $\mathbf{h}_t$ at all end-of-sentence tokens as the representation of T_j, which are denoted as a sequence of sentence embeddings $\mathbf{s}_1, \mathbf{s}_2 \cdots \mathbf{s}_\tau \cdots$ for item v_j, as shown in Figure 3. Except for GRU, one can also use other sequence modeling techniques such as long-short term memory (LSTM) [15] or recurrent neural networks (RNN) [28]. We take GRU here for its efficiency.

Table 1: A summary of key notations in this work. Note that all vectors are denoted with bold lowercases.

u_i, $\mathcal{U}$	The i-th user and the set of all users in the system
v_j, $\mathcal{V}$	The j-th item and the set of all items in the system
M, N	Number of users $M = \lvert\mathcal{U}\rvert$, and number of items $N = \lvert\mathcal{V}\rvert$
T_j, $\lvert T_j\rvert$	Textual description of item v_j and its number of sentences
R_{ij}	The textual review that user u_i wrote for item v_j
Q_0	The initial request of a conversation
c	The product category specified in initial request
Q_k, A_k	The k-th system question and user answer in a conversation
p_k, q_k	The aspect asked in Q_k, and its value answered in A_k
K	The length (i.e., number of QA pairs) of a conversation
D	Dimension of all embedding vectors in this paper
w_t, $\mathbf{w}_t$	The t-th word and its word embedding vector
$\mathbf{s}_\tau$, $\mathbf{s}^r$	The τ-th sentence embedding in textual description T_j, and their weighted summarization in the r-th memory hop
$\mathbf{c}_k$	Query embedding until the k-th conversational round
$\mathbf{p}_k$, $\mathbf{q}_k$	Embeddings of the k-th aspect and value in a conversation
$\mathbf{m}_j^r$	Memory embedding of item v_j at the r-th hop
κ_s, κ_q	Number of negative samples in search and question modules

4.2 Query Representation

By the end of the k-th round of a user-system conversation, we would have collected the initial request $Q_0(c)$ and k question-answer pairs $Q_1(p_1)A_1(q_1), Q_2(p_2)A_2(q_2) \cdots Q_k(p_k)A_k(q_k)$.

To generate the query representation $\mathbf{c}_k$ at the k-th round, we construct $k+1$ sentences. The first sentence is the initial request $Q_0(c)$, and each subsequent sentence is the concatenation of the corresponding aspect-value pair (p, q), as shown in Figure 3.

Similar to item representations, we also insert an end-of-sentence token at the end of each sentence, and apply the GRU procedure $\mathbf{h}_t = GRU(\mathbf{w}_t, \mathbf{h}_{t-1})$ through the sequence. The difference is that we adopt the final hidden state $\mathbf{c}_k$ as the query representation (as shown in Figure 3), which is the same as conventional LSTM/RNN models without attention, and the final hidden state has included the information embedded in the whole query sequence.

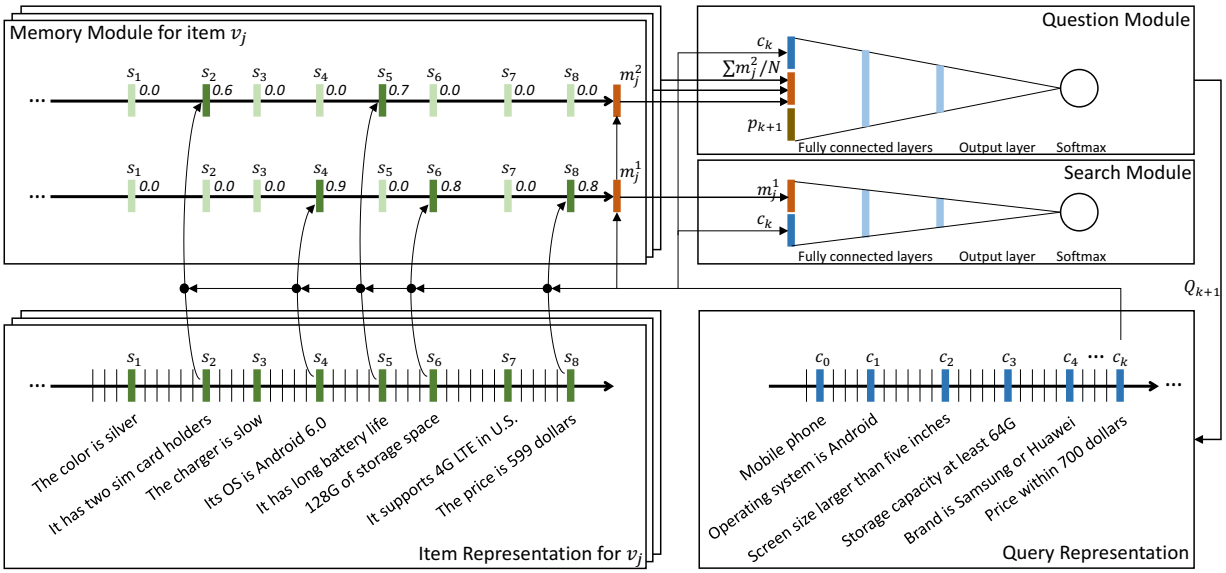

Figure 3: The Multi-Memory Network (MMN) architecture for conversational search and recommendation, including five components: query and item representations, the memory module, as well as the question and search modules.

To guarantee vocabulary alignment between query and item representations, we adopt the same word embedding matrix L and the same GRU model (i.e., sharing the same parameter matrices and bias vectors) as used in item representation module.

4.3 Memory Module with Attention Mechanism

Intuitively, not all sentences in the item representation are relevant to the current search query. As a result, considering all of the sentence embeddings (i.e., the $\mathbf{s}$ vectors) equally would introduce noise to the search and next question generation tasks. Similar to previous work on memory networks [17, 24, 32], we introduce an attention mechanism to construct memory embeddings so that the system can automatically select important signals from each item to support the search and question generation.

In our model, the attention mechanism iterates for two hops and generates two memory embeddings $\mathbf{m}_j^1$ and $\mathbf{m}_j^2$ for each item v_j, and the two memory embeddings are used for search and question generation tasks, respectively. Detailed reasons for this two-hop design are explained in the following.

4.3.1 ***Attention Weights.*** We consider generating memory embedding $\mathbf{m}_j^r$ in the r-th hop ($r = 1, 2$) for item v_j. To calculate the attention weight ω_τ^r for sentence embedding $\mathbf{s}_\tau$, we adopt the sentence embedding $\mathbf{s}_\tau$, the current query representation $\mathbf{c}_k$, and the memory embedding of the previous hop $\mathbf{m}_j^{r-1}$ as inputs, where $\mathbf{m}_j^0 = \mathbf{c}_k$. Specifically, we concatenate the embeddings as a joint embedding $\mathbf{z}_\tau^r = [\mathbf{s}_\tau^\intercal, \mathbf{c}_k^\intercal, \mathbf{m}_j^{r-1\intercal}]^\intercal \in \mathbb{R}^{3D\times 1}$, and adopt a two-layer feed forward network to calculate the attention weight ω_τ^r,

$$\omega_\tau^r = \sigma\left(\mathbf{w}_\omega^\intercal \sigma\left(W_\omega \mathbf{z}_\tau^r + \mathbf{b}_\omega\right) + b_\omega\right) \tag{5}$$

where $W_\omega \in \mathbb{R}^{D\times 3D}$, $\mathbf{b}_\omega$ and $\mathbf{w}_\omega \in \mathbb{R}^{D\times 1}$, as well as the scalar b_ω are parameters to learn, and $\sigma(\cdot)$ is the sigmoid function. For clarity, only the significant weights are shown in Figure 3.

With the attention weights, we apply the weighted average strategy to get the summarized representation for item v_j at hop r:

$$\mathbf{s}_j^r = \sum_{\tau=1}^{|T_j|} \omega_\tau^r \mathbf{s}_\tau \Big/ \sum_{\tau=1}^{|T_j|} \omega_\tau^r \tag{6}$$

where $|T_j|$ is the number of sentences in item description T_j.

Intuitively, the first hop would attend on those sentence embeddings $\mathbf{s}_\tau$ that are relevant to the current query $\mathbf{c}_k$, while by taking $\mathbf{m}_j^1$ into consideration, the second hop would attend on some new information that is relevant to the signals selected in the first hop. This is why the two hops are used for search and question generation, respectively, which will be analyzed in the following.

4.3.2 ***Memory Embedding.*** For high quality search, we expect the search module to rely on the current query $\mathbf{c}_k$ and the summarized signals relevant to the query in the first hop $\mathbf{s}_j^1$, which are already confirmed information in this search conversation. However, estimating the probability of a new aspect to ask in the question module not only needs the query and its directly relevant signals, but also the extended new information $\mathbf{s}_j^2$ in the second hop, which helps to find out the feature related to previously asked ones, and this is why we adopt the two-hop design for the multi-memory network.

As a result, for item v_j we consider the summarized signal $\mathbf{s}_j^r$ at each hop as input vector, and adopt a gated recurrent network to update the memories. Specifically, we have,

$$\mathbf{m}_j^r = GRU(\mathbf{s}_j^r, \mathbf{m}_j^{r-1}),\ r = 1, 2 \tag{7}$$

where $\mathbf{m}_j^0$ is also initialized as $\mathbf{c}_k$, and all the items share the same set of GRU parameters to learn, but this parameter set is independent from that used in Eq.(4).

In this way, we generate the search memory $\mathbf{m}_j^1$ and question memory $\mathbf{m}_j^2$ for each item v_j. The whole memory module is generative and thus is differentiable for model optimization.

4.4 Search Module

Given the current query $\mathbf{c}_k$ and the search memory $\mathbf{m}_j^1$ that has encoded the relevant signals of item v_j, we construct a concatenated embedding $\mathbf{z}_j = [\mathbf{c}_k^\intercal, \mathbf{m}_j^{1\intercal}]^\intercal$, and then employ a two-layer fully connected network for dimension reduction,

$$\mathbf{x}_j = \phi\left(W_s^{(2)}\phi(W_s^{(1)}\mathbf{z}_j + \mathbf{b}_s^{(1)}) + \mathbf{b}_s^{(2)}\right) \tag{8}$$

where $\phi(\cdot)$ is the ELU (exponential linear units) activation function to avoid vanishing gradients [8], and the parameters are $W_s^{(1)} \in \mathbb{R}^{D\times 2D}, \mathbf{b}_s^{(1)} \in \mathbb{R}^{D\times 1}, W_s^{(2)} \in \mathbb{R}^{\frac{D}{2}\times D}$ and $\mathbf{b}_s^{(2)} \in \mathbb{R}^{\frac{D}{2}\times 1}$. As a result, the compact embedding is $\mathbf{x}_j \in \mathbb{R}^{\frac{D}{2}\times 1}$.

Based on this, we further adopt a softmax output layer to calculate the probability of v_j,

$$P(v_j|\mathbf{c}_k, \mathbf{m}_j^1) = \frac{\exp(\mathbf{w}_s^\intercal \mathbf{x}_j + b_s)}{\sum_{j'}\exp(\mathbf{w}_s^\intercal \mathbf{x}_{j'} + b_s)} \tag{9}$$

where the parameter $\mathbf{w}_s \in \mathbb{R}^{\frac{D}{2}\times 1}$ and b_s is a scalar. We also adopt a sigmoid function to calculate the confidence score of item v_j, i.e., $\text{conf}(v_j) = \sigma(\mathbf{w}_s^\intercal \mathbf{x}_j + b_s)$.

To train the search module, we maximize the probability of the eventually purchased item of a conversation against other items. However, directly computing the log-likelihood of item probability with Eq.(9) is not practical because the denominator requires all the (thousands or even millions of) items in each iteration. For efficient training, we adopt the negative sampling strategy to approximate the softmax probability. Negative sampling was first proposed by Mikolov et al. [23] and has now been extensively used for machine learning and information retrieval [1, 2, 18].

Suppose v_j is the actually purchased item of a training conversation, the basic idea is to randomly sample some unpurchased items (i.e., items except for v_j) as negative samples to approximate the denominator of softmax function, and the log-likelihood to be maximized for item v_j until the k-th round of conversation is,

$$\begin{aligned}\mathcal{L}_k^s = \log P(v_j|\mathbf{c}_k, \mathbf{m}_j^1) &= \log\sigma\left(\mathbf{w}_s^\intercal \mathbf{x}_j + b_s\right) \\ &+ \kappa_s \cdot \mathbb{E}_{j'\sim P_s}\left[\log\sigma\left(-(\mathbf{w}_s^\intercal \mathbf{x}_{j'} + b_s)\right)\right]\end{aligned} \tag{10}$$

where $k = 0, 1\cdots K$, κ_s is the number of negative samples, and P_s is the global item popularity distribution raised to 3/4 power [23].

It is interesting to note that the negative sampling strategy actually simulates the practical scenario where we have true negative feedback, e.g., if we displayed the results to the user but the conversation did not stop (i.e., the user is not satisfied with the results), then we can use the already displayed items as true negatives.

4.5 Question Module

The question module aims at correctly predicting the next question to ask. To do so, we also train the model to maximize the probability of the next aspect in a conversation. Given the current query $\mathbf{c}_k$, the word embedding of next aspect $\mathbf{p}_{k+1}$, and the second-hop memory $\mathbf{m}_j^2$ of all items, we construct a concatenated embedding,

$$\mathbf{z}_{k+1} = \left[\mathbf{c}_k^\intercal, \left(\frac{1}{N}\sum\nolimits_{j=1}^{N}\mathbf{m}_j^2\right)^\intercal, \mathbf{p}_{k+1}^\intercal\right]^\intercal \tag{11}$$

where the second term is the average of the second-hop memories of all items, and in cases where the next aspect contains two or more words, we average the word embeddings as the aspect embedding $\mathbf{p}_{k+1}$. Similar to the search module, we also adopt a two-layer feed forward neural network for dimension reduction:

$$\mathbf{x}_{k+1} = \phi\left(W_q^{(2)}\phi(W_q^{(1)}\mathbf{z}_{k+1} + \mathbf{b}_q^{(1)}) + \mathbf{b}_q^{(2)}\right) \tag{12}$$

where $\phi(\cdot)$ is also the ELU function, and the parameters are $W_q^{(1)} \in \mathbb{R}^{2D\times 3D}, \mathbf{b}_q^{(1)} \in \mathbb{R}^{2D\times 1}, W_q^{(2)} \in \mathbb{R}^{D\times 2D}$ and $\mathbf{b}_q^{(2)} \in \mathbb{R}^{D\times 1}$. Thus the softmax output layer for probability estimation is,

$$P(p_{k+1}|\mathbf{c}_k, \mathbf{m}_1^2\cdots\mathbf{m}_N^2, \mathbf{p}_{k+1}) = \frac{\exp(\mathbf{w}_q^\intercal \mathbf{x}_{k+1} + b_q)}{\sum_{k'}\exp(\mathbf{w}_q^\intercal \mathbf{x}_{k'} + b_q)} \tag{13}$$

where $\mathbf{w}_q \in \mathbb{R}^{D\times 1}$ and b_q is a scalar, and the denominator sums over all of the unasked aspects.

For efficiency, we also adopt negative sampling for probability estimation, and the log-likelihood to maximize for aspect p_{k+1} is,

$$\begin{aligned}\mathcal{L}_k^q = \log P(p_{k+1}|\mathbf{c}_k, \mathbf{m}_1^2\cdots\mathbf{m}_N^2, \mathbf{p}_{k+1}) &= \log\sigma\left(\mathbf{w}_q^\intercal \mathbf{x}_{k+1} + b_q\right) \\ &+ \kappa_q \cdot \mathbb{E}_{k'\sim P_q}\left[\log\sigma\left(-(\mathbf{w}_q^\intercal \mathbf{x}_{k'} + b_q)\right)\right]\end{aligned} \tag{14}$$

where $k = 0, 1, 2\cdots K-1$, κ_q is the number of sampled negative aspects, and P_q is the global aspect popularity distribution in the reviews raised to 3/4 power [23].

4.6 The Unified MMN Architecture

Let I_{ij} be the indicator function to indicate if there is a training conversation between user u_i and item v_j, and let K_{ij} be the length of the conversation, i.e., $u_i \rightarrow Q_0 \mid Q_1A_1, Q_2A_2\cdots Q_{K_{ij}}A_{K_{ij}} \mid v_j$, then the final Multi-Memory Network (MMN) architecture for conversational search optimizes the following unified objective function,

$$\mathcal{L} = \sum_{i,j} I_{ij}\cdot\left(\lambda_s\sum_{k=0}^{K_{ij}}\mathcal{L}_k^s + \lambda_q\sum_{k=0}^{K_{ij}-1}\mathcal{L}_k^q\right) + \lambda_\Theta\|\Theta\|_2^2 \tag{15}$$

where $\mathcal{L}_k^s$ and $\mathcal{L}_k^q$ are the search and question prediction objective functions (Eq.(10) and (14)), respectively, $\lambda_s, \lambda_q, \lambda_\Theta$ are regularization coefficients, and Θ is the set of parameters in the model.

Intuitively, we train the model by maximizing the probability of correctly predicting the true item and the next question at each round of each conversation, and we apply an ℓ_2 regularizer to the parameters. Because the whole framework is generative beginning from the word embeddings to the search and question prediction results, so the whole framework is easily trainable based on stochastic gradient descent methods. Specifically, we initialize the word embeddings with Google word2vec, and adopt stochastic gradient descent (SGD) for model training.

Once we have the trained the conversation model, the system can generate the next question to ask by selecting the candidate aspect of the highest probability (Eq.(13)) in each round, and also conduct search by ranking the items in descending order of item probability (Eq.(9)). If the confidence of the top item is higher than a threshold, then the top-n results will be displayed to the user.

4.7 Personalized Multi-Memory Network

An important nature of product search and recommendation is personalization [2], because different users may care about different aspects even for the same product, and they may prefer different items even under the same conversation. To model the inherent

Table 2: Basic statistics of the experimental datasets, where ℓ(Request) is the average length of initial requests, and the number of reviews is also the total number of conversations because each review is transformed into a conversation.

Dataset	#Users	#Items	#Reviews	#Aspect	#Value	#AV pairs	#Request	ℓ(Request)	Training/Testing	
									#Conversations	#Relevant Items per Conv
Electronics	142,421	53,278	365,341	479	500	475,020	989	6.40	255,739/109,602	1.21 ± 0.62/1.13 ± 0.26
CDs & Vinyl	64,847	60,405	427,031	514	747	659,737	694	5.71	298,922/128,109	2.82 ± 5.88/1.46 ± 1.26
Kindle Store	56,847	53,907	285,104	164	359	367,159	4,603	7.07	199,573/85,531	1.94 ± 3.63/1.62 ± 2.21
Cell Phones	21,615	9,292	52,178	325	402	68,709	165	5.93	36,525/15,653	1.66 ± 1.32/1.24 ± 0.16

personalized preferences of users, we slightly modify the MMN architecture to propose a personalized version of the model (PMMN).

Specifically, we introduce an embedding vector $\mathbf{u}_i$ for user u_i. In the search and the question modules, we include the corresponding user embedding when constructing the concatenated vector $\mathbf{z}_j$ (used in Eq.(8)) and $\mathbf{z}_{k+1}$ (in Eq.(11)). The user embeddings are randomly initialized and the whole model is still trained with SGD.

5 EXPERIMENTS

5.1 Dataset Description

As in previous work on product search and recommendation tasks [2, 35], we adopt the Amazon product dataset [22] as the experimental corpus. It includes millions of customers and products, and rich metadata such as reviews, product descriptions and multi-level product categories. The dataset includes 24 sub-datasets of different product types. In this work, we adopt four product category datasets for experiments, which are *Electronics, CDs & Vinyl, Kindle Store*, and *Cell Phones*. The first three are large-scale datasets while the last one is a smaller dataset to test our model performance with particularly sparse data. Basic statistics of the datasets are shown in the first block of Table 2. We adopt the aspect-value pair extraction toolkit by Zhang et al [44, 45] to extract the pairs for each dataset. Then, each review is converted into a conversation $u_i \rightarrow Q_0 \mid Q_1A_1, Q_2A_2 \cdots Q_KA_K \mid v_j$ based on the pairs mentioned in the review (as in Section 3)[1]. The number of aspects and value words, and the number of aspect-value pairs on each dataset are also shown in Table 2, and Figure 4 shows the number of conversations over the length of conversation for each dataset.

5.2 Experimental Setup

5.2.1 ***Initial Request Construction.*** Intuitively, if we adopt the same top-level product category (e.g., "Cell Phone") as the initial request for all conversations in a dataset, then the non-personalized MMN model will produce the same predicted aspect sequence and search results for all testing conversations, which makes the system less useful in practice. Though the PMMN model will predict different aspects for different conversations, we still want the non-personalized MMN to reflect the fact that users may specify different requests when initiating a conversational search.

We follow the three-step paradigm of product search [2, 35] to construct the initial request Q_0 for a conversation. 1) we extract the multi-level category information of item v_j from the metadata, 2) we concatenate the terms as a topic string, and 3) stopwords and duplicate words are removed from the string. The number of constructed initial requests and their average length are in Table 2.

[1]Dataset can be accessed at http://yongfeng.me/dataset

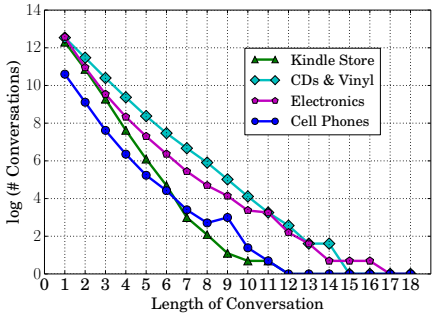

Figure 4: Statistical distribution of the number of conversations over the length of conversations on four datasets.

5.2.2 ***Train-Test Split.*** For each user, we randomly select 70% of his/her reviews to construct training conversations, while the remaining 30% conversations are taken for testing.

To evaluate the search performance with multiple (instead of just one) relevant items, same as [2], those items that are purchased by the user and belong to the initial request are considered as relevant to the conversation. The number of training and testing conversations as well as the average number of relevant items per conversation are shown in the last block of Table 2.

5.2.3 ***Baselines.*** We take the following representative product search or recommendation methods as baselines:

• **QL**: The query likelihood model by Ponte and Croft [25], which is a language modeling approach by ranking documents with the log-likelihood of the query in the document's language models.

• **LSE**: The latent semantic entity model [35], which is an important latent space model for non-personalized product search.

• **HEM**: The hierarchical embedding model [2] for personalized product search. We take the best reported HEM implementation (i.e., using non-linear projected mean for query embedding) and use personalization weight $\lambda = 0.5$ for comparison.

• **HFT**: The Hidden Factors and Topics model for recommendation with textual reviews [21]. The original model is not designed for top-n recommendation, for fair comparison, we apply Bayesian personalized ranking on top of HFT for better top-n performance.

• **EFM**: The Explicit Factor Model for explainable recommendation [44], which also adopts textual reviews for recommendation.

For all of the product search baselines we take the initial request as query for search, which is consistent with the experimental settings in [35] and [2]. For our own model, we experiment with both the non-personalized (**MMN**) and personalized (**PMMN**) versions.

5.2.4 ***Evaluation Measures.*** For the evaluation of next question generation, we calculate the *Hit Ratio* of predicting n aspects at conversation round k ($HR@n, k$). Specifically, when predicting for the k-th round in a conversation, we rank the candidate aspects by

Table 3: Performance comparison on item retrieval tasks, where MAP and MRR are calculated with top 100 items while NDCG is calculated with top 10 items. Starred numbers (∗) are best baseline performances, + and ‡ denote significant improvements against LSE and HEM respectively for Fisher randomization test [30] with $p \leq 0.1$. Bolded numbers are the best performances.

Dataset	*Electronics*			*CDs & Vinyl*			*Kindle Store*			*Cell Phones*		
Methods	MAP	MRR	NDCG	MAP	MRR	NDCG	MAP	MRR	NDCG	MAP	MRR	NDCG
QL	0.276	0.278	0.305	0.009	0.011	0.013	0.009	0.011	0.012	0.078	0.081	0.084
LSE	0.242	0.245	0.246	0.011	0.013	0.014	0.022	0.025	0.022	0.106	0.108	0.098
HEM	0.304*	0.306*	0.322*	0.026*	0.033*	0.032*	0.033	0.037*	0.038	0.118*	0.120*	0.148*
HFT	0.258	0.262	0.278	0.022	0.026	0.028	0.029	0.034	0.036	0.035	0.041	0.043
EFM	0.271	0.276	0.285	0.025	0.027	0.030	0.034*	0.037*	0.040*	0.078	0.085	0.106
MMN	0.275+	0.278+	0.296+	0.018	0.021+	0.022+	0.027	0.031+	0.028+	0.112	0.112	0.127+
PMMN	**0.312**‡	**0.314**‡	**0.328**‡	**0.034**‡	**0.037**‡	**0.039**‡	**0.038**‡	**0.041**‡	**0.044**‡	**0.122**	**0.124**	**0.155**‡

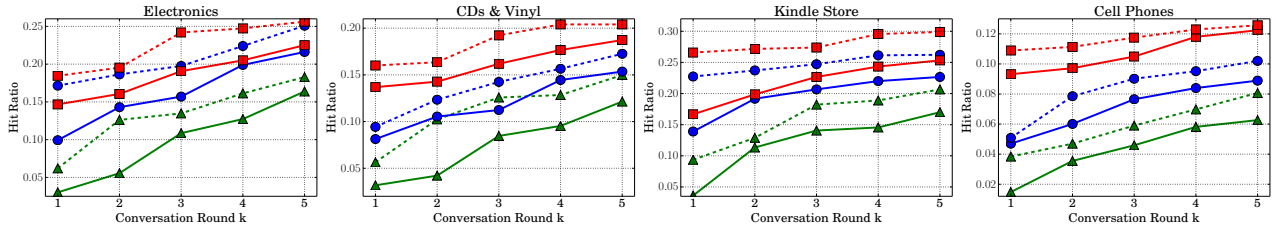

Figure 5: Hit ratio of question prediction on four datasets. The x-axis means predicting the k-th aspect in a conversation, and y-axis is $HR@n$, where $n = 1, 3, 5$ corresponds to each line: green △ line for n=1, blue ○ line for n=3, and red □ line for n=5. Solid lines are results of MMN, and dashed lines are for PMMN. Improvements of PMMN from MMN are significant with $p \leq 0.1$.

probability (Eq.(13)) and adopt the top-n of them as the prediction list, and a list is considered 'hit' if the true aspect is within the list.

To evaluate the search/rec results at conversation round k, we adopt mean average precision ($MAP@k$), mean reciprocal rank ($MRR@k$), and normalized discounted cumulative gain ($NDCG@k$), where MAP and MRR are calculated based on top 100 items, and NDCG is calculated based on top 10 items. Notice that, the parameter k here is not the length of result list but the number of rounds of conversation, which enumerates from 1 to 5.

Intuitively, MRR indicates the expected number of items a user has to explore before finding the first right item. Note that using top 100 items to calculate MAP and MRR does not mean that our system has to display 100 items to users, the setting is only for convenience to avoid baselines from nearly zero MAP or MRR scores.

5.2.5 ***Parameter Settings.*** We primarily set the coefficients $\lambda_s = \lambda_q = 1$, and $\lambda_\Theta = 0.005$ in Eq.(15). Embedding size D is set as 300, and in Section 5.6 we tune D to other choices to see its effect. We set the number of negative samples $\kappa_s = \kappa_q = 5$ in Eq.(10) and (14). The initial learning rate is 0.5 and gradually decreases in training. We use SGD with batch size 200, and use global norm clip with 5 for stable training.

5.3 Evaluation of Question Prediction

We first briefly report the performance of the question prediction component in this subsection, so as to provide an intuition about how the predicted aspects are close to the true aspects mentioned in user reviews. And in the following subsections, we will focus on conversational search and recommendation performance.

For computational efficiency, we randomly sample 20 items to calculate the average second-hop memory in Eq.(11). The performance of question prediction under different choices of n, k pairs are shown in Figure 5, where k is the round of conversation (i.e., we predict the k-th aspect), and n means our model adopts the top-n predicted aspects at each round. Each line in the figure shows how the performance changes with k under a certain n for the MMN and PMMN models.

We see that predicting more aspects (changing n) at a time increases the hit ratio, which is intuitive. We also see that when the number of predictions n is fixed, the performance tends to increase with k, namely, with the progress of the conversation. This is because when the conversation progresses, those aspect-value pairs collected from previous conversation rounds give us more information about the user needs at this conversation, which helps our model to generate more comprehensive query representations for better performance in the question module.

We also see that the model relies on large-scale training conversations to achieve satisfactory prediction performance. For example, on the *Electronics*, *CDs & Vinyl*, and *Kindle Store* datasets, $HR@5, 5$ score can reach around 20% ∼ 30% (the score for *Kindle Store* dataset is high partly due to its smaller aspect prediction space), while for the *Cell phone* dataset it is around 12%. This implies that large-scale conversation data is needed so as to train (deep) memory networks for practical conversational search and recommendation system.

Finally, PMMN achieved significantly better performance than the non-personalized MMN. This verifies our assumption that product search and recommendation tasks can be very personalized, because users may have different vocabulary preferences, and may care about different aspects even for the same product.

5.4 Evaluation of Search/Rec Performance

In this section, we set the length of conversation $k = 3$ and study the search and recommendation performance of different models. Performance under different conversation length settings will be

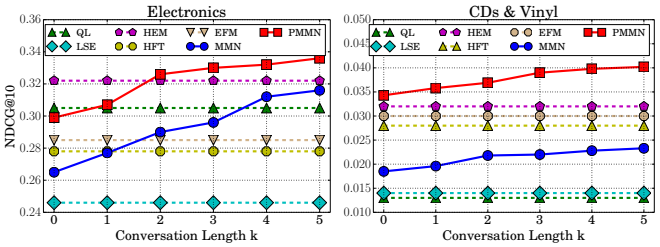

Figure 6: Performance of (P)MMN with different choices of conversation length k. Dashed lines are baseline methods.

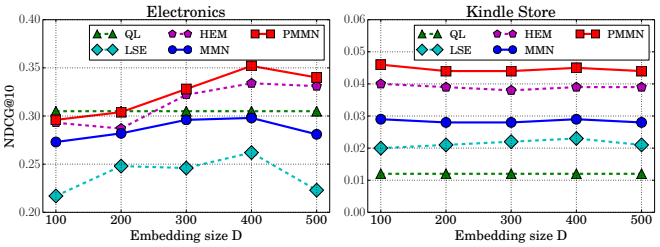

Figure 7: Performance of (P)MMN with different choices of embedding size D. Dashed lines are baseline methods.

analyzed in the next subsection. Table 3 shows the results of both baselines and our models.

Among the baselines, we see that the unigram QL approach achieved comparable performance with the learning-based models (LSE, HEM, HFT, EFM) on *Electronics* and *Cell Phones* datasets, while on *CDs & Vinyl* and *Kindle Store* the learning-based models are significantly better than the unigram approach. This observation is in accordance with the findings in [2]. Empirically, the language of user reviews on electronic devices and cell phones are more consistent with the product properties and aspects, while on music and books, the reviews are more about the user's personal feelings and understandings about the content or plot. As a result, the QL model could suffer more from the vocabulary mismatch problem on the *CDs* and *Kindle Store* datasets. This observation implies the advantage of word/query embedding in latent space models, which has the power of detecting semantic relations between words or phrases to alleviate vocabulary mismatch.

Overall, by considering user modeling for personalized product search, HEM is better than the non-personalized LSE model, with the best baseline performance on all datasets. When comparing our models with the baselines, we see that MMN is better than LSE on all datasets except for the *Cell Phones* dataset, where the improvements are not significant. Furthermore, PMMN is significantly better than the best baseline HEM.

The power of MMN against LSE mainly comes from two aspects. First is the advantage of conversational search over classical retrieval paradigms. In conversational search, the system has the opportunity to actively ask questions and get responses from the user, so as to collect more information and clarify the user intents. The collected information – represented as the aspect-value pairs in this work – can be encoded into the original query in the query representation module, which enhances the systems's understanding of the user needs. Second, not all the sentences in item description are equally important for a given search task, and in the MMN architecture, attention mechanism is leveraged so that the model can select and focus on important signals for better search and recommendation.

On the other hand, the fact that MMN did not compare favorably with HEM further suggests the importance of personalization in product search and recommendation, because user preferences may be very different even on the same product category. However, by enhancing our model to a personalized version, the PMMN approach is better than HEM. Except for the two advantages introduced above, another factor is that we adopted GRU for query and item representation learning, which has been shown to be more effective than standard tanh RNN [7], and has the power of learning semantic relations between sentences than non-linear projected mean used in HEM. Although it is shown in [2] that projected mean is better than RNN for HEM because the keyword queries (same as the initial requests in this work) do not have complicated compositional meanings, but in this work we do need to capture the complex semantic relations between the aspects and values, as well as among the sentences in item descriptions.

5.5 Effect of Conversation Length

In this section, we study the performance of (P)MMN under different conversation length k from 0 to 5, where $k = 0$ means we conduct search using only the initial request with no aspect-value pairs. Results on *Electronics* and *CDs & Vinyl* datasets are shown in Figure 6, and observations on *Kindle Store* and *Cell Phones* are similar.

We see that both the performance of MMN and PMMN increase with the progress of the conversation, which is intuitive because with more aspect-value pairs for query representation learning, the model can learn user preferences more accurately, and gain a higher probability to find the right item with more information describing the user needs.

We also see that when $k = 0$, MMN and PMMN could be better than LSE and HEM respectively on some of the datasets. In this case, our models leverage the initial request for query representation learning and search, which is the same as the baselines, and both MMN and PMMN degenerate to non-conversational search models. This observation indicates the advantage of our multi-memory architecture for user, item, and query modeling, which can extract word/sentence semantic relations based on GRU sequential learning, and select important signals from reviews with attention modeling.

5.6 Effect of Embedding Size

We further study the effect of embedding size D for (P)MMN and the latent space baselines LSE and HEM. Specifically, we fix conversation length $k = 3$ as in Section 5.4, and tune the embedding size in each method from 100 to 500. We did not include HFT and EFM here because their optimal embedding size is much smaller (within 50). The results on *Electronics* and *Kindle Store* are shown in Figure 7 for reference.

We find that the performance regarding embedding size D vary on different datasets. On the *Electronics* and *CDs & Vinyl* datasets, the performance tends to increase at first and then tends to drop, where the best performance is achieved when $D = 300 \sim 400$. While on the *Kindle Store* and *Cell Phones* datasets, the best performance is achieved when $D = 100$, and increasing D does not help to gain better performance.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we propose to conduct conversational search and recommendation based on emerging conversational devices and systems. We believe that one of the most important advantages of conversational search and recommendation against conventional approaches is that the system can actively ask appropriate questions so as to understand the user needs – a fundamental goal of search and recommendation systems. As a result, we proposed a *system ask – user respond* (SAUR) paradigm towards conversational search and recommendation.

Based on this paradigm, we further proposed a multi-memory network architecture as well as its personalized version for conversational search and recommendation, which integrates the power of both sequential modeling and attention mechanisms. Experiments in the Amazon e-commerce scenario based on real-world user purchase datasets verified the performance of our approach against state-of-the-art product search and recommendation baselines.

The research on conversational search and recommendation is still in its initial stage, and this work is just one of our first steps towards intelligent conversational systems, where there is much room for future work and improvements. In this work, we assumed the user-system conversation to be about aspect-value pairs, while in the future, it is necessary for the system to perform more flexible conversations and to handle unexpected user responses appropriately. Except for the product search and recommendation scenario in this work, the proposed paradigm may also be extended to other conversational search and recommendation scenarios, such as conversational academic search, legal search, medical search, or even general web search, and it may even be applied to tasks beyond search and recommendation, such as conversational question answering based on intelligent devices.

ACKNOWLEDGEMENT

The authors thank the reviewers for the valuable comments and constructive suggestions. This work was supported in part by the Center for Intelligent Information Retrieval and in part by NSF IIS-1715095. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsors.

REFERENCES

[1] Qingyao Ai, Liu Yang, Jiafeng Guo, and Bruce Croft. 2016. Analysis of the paragraph vector model for information retrieval. In *ICTIR*. ACM, 133–142.
[2] Qingyao Ai, Yongfeng Zhang, Keping Bi, Xu Chen, and Bruce Croft. 2017. Learning a Hierarchical Embedding Model for Personalized Product Search. In *SIGIR*.
[3] Nicholas J Belkin, Colleen Cool, Adelheit Stein, and Ulrich Thiel. 1995. Cases, scripts, and information-seeking strategies: On the design of interactive information retrieval systems. *Expert systems with applications* 9, 3 (1995), 379–395.
[4] Antoine Bordes, Y-Lan Boureau, and Jason Weston. 2017. Learning end-to-end goal-oriented dialog. *ICLR* (2017).
[5] Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv* (2015).
[6] Konstantina Christakopoulou, Filip Radlinski, and Katja Hofmann. 2016. Towards conversational recommender systems. In *KDD*.
[7] Junyoung Chung and Caglar Gulcehre et al. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv* (2014).
[8] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2016. Fast and accurate deep network learning by exponential linear units (elus). *ICLR* (2016).
[9] W. Bruce Croft and Roger H. Thompson. 1987. I^3R: A New Approach to the Design of Document Retrieval Systems. *JASIS* 38, 6 (1987), 389.
[10] Jesse Dodge, Andreea Gane, and Xiang Zhang et al. 2016. Evaluating prerequisite qualities for learning end-to-end dialog systems. *ICLR* (2016).
[11] Huizhong Duan, ChengXiang Zhai, Jinxing Cheng, and Abhishek Gattani. 2013. A probabilistic mixture model for mining and analyzing product search log. In *CIKM*. 2179–2188.
[12] Huizhong Duan, ChengXiang Zhai, Jinxing Cheng, and Abhishek Gattani. 2013. Supporting keyword search in product database: a probabilistic approach. *Proceedings of the VLDB Endowment* 6, 14 (2013), 1786–1797.
[13] Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401* (2014).
[14] Matthew Henderson, Blaise Thomson, and Steve Young. 2013. Deep neural network approach for the dialog state tracking challenge. In *SIGDIAL*. 467–471.
[15] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
[16] Tom Kenter and Maarten de Rijke. 2017. Attentive memory networks: Efficient machine reading for conversational search. In *CAIR*. ACM.
[17] A. Kumar, O. Irsoy, P. Ondruska, M. Iyyer, and J. Bradbury et al. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *ICML*.
[18] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*. 1188–1196.
[19] Jiwei Li, A. H Miller, Sumit Chopra, Marc'Aurelio Ranzato, and Jason Weston. 2017. Learning through dialogue interactions by asking questions. *ICLR* (2017).
[20] Soon Chong Johnson Lim, Ying Liu, and Wing Bun Lee. 2010. Multi-facet product information search and retrieval using semantically annotated product family ontology. *Information Processing & Management* 46, 4 (2010), 479–493.
[21] Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *RecSys*. 165–172.
[22] Julian McAuley, Rahul Pandey, and Jure Leskovec. 2015. Inferring networks of substitutable and complementary products. In *KDD*. ACM, 785–794.
[23] Tomas Mikolov, Ilya Sutskever, and Kai Chen et al. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.
[24] A. Miller, A. Fisch, J. Dodge, A. Karimi, A. Bordes, and J. Weston. 2016. Key-value memory networks for directly reading documents. *EMNLP* (2016).
[25] Jay M Ponte and W Bruce Croft. 1998. A language modeling approach to information retrieval. In *SIGIR*. ACM, 275–281.
[26] Filip Radlinski and Nick Craswell. 2017. A theoretical framework for conversational search. In *CHIIR*. ACM, 117–126.
[27] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor. 2015. Recommender systems handbook. Springer.
[28] David E. Rumelhart and Geoffrey E. Hinton et al. 1988. Learning representations by back-propagating errors. *Cognitive modeling* 5, 3 (1988), 1.
[29] Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2017. Reasonet: Learning to stop reading in machine comprehension. In *KDD*. ACM, 1047–1055.
[30] Mark D Smucker, James Allan, and Ben Carterette. 2007. A comparison of statistical significance tests for information retrieval evaluation. In *CIKM*.
[31] Damiano Spina, Johanne R Trippas, Lawrence Cavedon, and Mark Sanderson. 2017. Extracting audio summaries to support effective spoken document search. *JAIST* 68, 9 (2017), 2101–2115.
[32] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *NIPS*. 2440–2448.
[33] Paul Thomas, Daniel McDu, Mary Czerwinski, and Nick Craswell. 2017. MISC: A data set of information-seeking conversations. In *CAIR*. ACM.
[34] Johanne R Trippas, Damiano Spina, Lawrence Cavedon, and Mark Sanderson. 2017. A Conversational Search Transcription Protocol and Analysis. In *CAIR*.
[35] Christophe Van Gysel, Maarten de Rijke, and Evangelos Kanoulas. 2016. Learning latent vector spaces for product search. In *CIKM*. ACM, 165–174.
[36] Tsung-Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gasic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. *EACL* (2017), 438–449.
[37] J. Weston, A. Bordes, S. Chopra, A. Rush, and B. Merriënboer et al. 2016. Towards ai-complete question answering: A set of prerequisite toy tasks. *ICLR* (2016).
[38] Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. *ICLR* (2015).
[39] Jason E Weston. 2016. Dialog-based language learning. In *NIPS*. 829–837.
[40] Caiming Xiong, Stephen Merity, and Richard Socher. 2016. Dynamic memory networks for visual and textual question answering. In *ICML*. 2397–2406.
[41] L. Yang, M. Qiu, C. Qu, J. Guo, Y. Zhang, W. B. Croft, J. Huang, and H. Chen. 2018. Response Ranking with Deep Matching Networks and External Knowledge in Information-seeking Conversation Systems. *SIGIR* (2018).
[42] Liu Yang, Hamed Zamani, Yongfeng Zhang, Jiafeng Guo, and W Bruce Croft. 2017. Neural Matching Models for Question Retrieval and Next Question Prediction in Conversation. *NEUIR Workshop* (2017).
[43] Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proc. IEEE* 101, 5 (2013).
[44] Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. 2014. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *SIGIR*. ACM, 83–92.
[45] Yongfeng Zhang, Haochen Zhang, Min Zhang, Yiqun Liu, and Shaoping Ma. 2014. Do users rate or review?: Boost phrase-level sentiment labeling with review-level sentiment classification. In *SIGIR*. ACM, 1027–1030.