Personalized Reason Generation for Explainable Song Recommendation

GUOSHUAI ZHAO, Xi'an Jiaotong University, China
HAO FU and RUIHUA SONG, Microsoft XiaoIce, China
TETSUYA SAKAI, Waseda University, Japan
ZHONGXIA CHEN, University of Science and Technology of China, China
XING XIE, Microsoft Research Asia, China
XUEMING QIAN, Xi'an Jiaotong University, China

Personalized recommendation has received a lot of attention as a highly practical research topic. However, existing recommender systems provide the recommendations with a generic statement such as "Customers who bought this item also bought . . . ". Explainable recommendation, which makes a user aware of why such items are recommended, is in demand. The goal of our research is to make the users feel as if they are receiving recommendations from their friends. To this end, we formulate a new challenging problem called personalized reason generation for explainable recommendation for songs in conversation applications and propose a solution that generates a natural language explanation of the reason for recommending a song to that particular user. For example, if the user is a student, our method can generate an output such as "Campus radio plays this song at noon every day, and I think it sounds wonderful," which the student may find easy to relate to. In the offline experiments, through manual assessments, the gain of our method is statistically significant on the relevance to songs and personalization to users comparing with baselines. Large-scale online experiments show that our method outperforms manually selected reasons by 8.2% in terms of click-through rate. Evaluation results indicate that our generated reasons are relevant to songs and personalized to users, and they attract users to click the recommendations.

CCS Concepts: • **Information systems** → **Personalization**; **Chat**; *Social recommendation*; • **Computing methodologies** → *Natural language generation*;

Additional Key Words and Phrases: Conversational recommendation, explainable recommendation, natural language generation, personalization, recommender system

The work was done when G. Zhao was an intern at Microsoft.
This work was supported in part by NSFC under Grant 61772407, Grant 61732008, Grant 61332018, and Grant u1531141, in part by the National Key Research and Development Program of China under Grant 2017YFF0107700, in part by the World-Class Universities (Disciplines), in part by the Characteristic Development Guidance Funds for the Central Universities under Grant PY3A022, and in part by the National Postdoctoral Innovative Talents Support Program for G. Zhao.
Authors' addresses: G. Zhao, Xi'an Jiaotong University, No. 28 Xianning Rd, Xi'an, Shannxi, 710049, China; email: guoshuai.zhao@mail.xjtu.edu.cn; X. Qian (corresponding author), the Ministry of Education Key Laboratory for Intelligent Networks and Network Security, Xi'an Jiaotong University, No. 28 Xianning Rd, Xi'an, Shannxi, 710049, China; email: qianxm@mail.xjtu.edu.cn; H. Fu and R. Song, Microsoft XiaoIce, No. 5 Danling St, Beijing, 100080, China; emails: {fuha, song.ruihua}@microsoft.com; T. Sakai, Waseda University, 3-4-1 Okubo, Shinjuku-ku, Tokyo 169-8555 Japan; email: tetsuyasakai@acm.org; Z. Chen, University of Science and Technology of China, No. 96 JinZhai Rd, Hefei, 230026, China; email: czx87@mail.ustc.edu.cn; X. Xie, Microsoft Research Asia, No. 5 Danling St, Beijing, 100080, China; email: xing.xie@microsoft.com.

2157-6904/2019/07-ART41 $15.00
https://doi.org/10.1145/3337967

ACM Reference format:
Guoshuai Zhao, Hao Fu, Ruihua Song, Tetsuya Sakai, Zhongxia Chen, Xing Xie, and Xueming Qian. 2019. Personalized Reason Generation for Explainable Song Recommendation. *ACM Trans. Intell. Syst. Technol.* 10, 4, Article 41 (July 2019), 21 pages.
https://doi.org/10.1145/3337967

1 INTRODUCTION

Personalized recommendation is considered an effective approach to solving the "consumer product overload problem" in the digital era. The objective there is to learn which items may be suitable for a particular user rather than the average user. Besides the problem of *what* should be recommended, *why* they should be recommended has received a lot of attention. Explainable recommendation helps improve the effectiveness, efficiency, persuasiveness, and user satisfaction of recommender systems [1, 2]. However, generic explanations ("Customers who bought this item also bought . . . ") and feature-based explanations ("You may like this item because it is good at these features . . . ") proposed by existing recommender systems are designed by predefined forms, and a more natural explanation form could be a piece of free-text that explains to the user in natural language [1]. Thus, we regard recommendations with a generic explanation or some feature-based explanations are not natural and do not necessarily encourage the user to accept in conversation scenarios. In fact, it is possible that users will reject a recommended item that is actually suitable for them, just because the generic recommendation explanation is not appealing. In light of the above consideration, we formulate a problem that we call *reason generation* for explainable recommendation in conversation applications, where our goal is to increase the click-through rate of the recommended items by automatically generating recommendation reasons tailored to this goal. In the present study, we focus on the song recommendation domain, as we are working to improve a conversational song recommendation functionality of the XiaoIce chatbot,[1] which has over 100M users as of May 2018.

What constitutes an effective *reason* for recommending a song to the user? We consider the following requirements: (a) the generated reason is easy to read and understand as natural language text; (b) it is relevant to the recommended song, so the user can understand why that particular song is recommended; and (c) it is relevant to the interests and status of that particular user. For example, a girl who likes pop music (*interests*) may generally prefer to listen to a happy pop song in a major key, but not if she has just broken up with her boyfriend (*status*). This third requirement is particularly important for the recommender to gain the user's trust, so she will start accepting the recommended songs.

Figure 1 demonstrates how our explainable song recommendation actually works in the XiaoIce chatbot. In this example, the user says "I fell out of love. What should I listen to?" XiaoIce responds by recommending a song named *How to Weep Bitter Tears* while providing a reason for the recommendation: "Only lost in love to understand this song." When the user cannot sleep, it recommends a song and provides a reason:"Listen to this song at midnight, I feel full of memories." We would like to make the user feel as if she is receiving a recommendation from her friend, since people often accept recommendations given by their friends [3].

There are several challenges to generating effective reasons in explainable song recommendation via conversations. First, we do not have any existing data that consists of actual song recommendations from friends; instead, what we have are the comments on songs posted to a music website, and only some of them can be regarded as recommendation reasons. Second, we aim to

[1] http://www.msxiaoice.com/.

Fig. 1. A snapshot of XiaoIce, recommending songs to a user followed by corresponding reasons during conversations.

give a personalized reason to a specific user according to the description of his/her general interests and current status, although music websites lack these kinds of user tags. Furthermore, even if we are able to leverage the comments from the music website to generate recommendations for users, simply retrieving and reusing the comments will not satisfy the aforementioned Requirement (c). In particular, such an approach would not be able to handle recommendations of *new* songs (i.e., songs for which we do not have any comment data from the music website) at all.

As an initial investigation into solving the aforementioned challenges, we build datasets that consist of (song, user tag, reason) triplets and propose a method that learns to generate recommendation reasons. First, from a music website, we extract song comments that can be regarded as recommendation reasons using a two-phase method. Second, we collect reasons related to a particular user tag by semantically expanding the tag and then retrieving relevant reasons. Third, we use an encoder-decoder framework with attention [4–7] to generate a recommendation reason for a particular song and a user. Fourth, we automatically score each generated text and filter out bad ones. Our experimental results indicate that our proposed method statistically significantly improves on the DeepFM [8] baseline in terms of overall rating (by 8.9%), relevance to a song (by 3.0%), and personalization (by 16.5%). The average fluency score of generated reasons is as high as 2.93, where 2 means *acceptable* and 3 means *good*. Furthermore, we deploy our proposed methods on the XiaoIce chatbot and observe that the click-through rate of recommended songs improves by at least 8.2% over four different baselines.

Our main contributions are as follows:

- We formulate a novel problem—namely, reason generation for explainable recommendation in conversation applications. The task is to generate a personalized reason to make the user feel as if they are receiving a recommendation from their friends.
- We demonstrate how to overcome the problem posed by a lack of training data for generating conversational reasons in the song recommendation domain. Moreover, our method is extensible to other recommendation domains.
- We propose fusing user tags with the information of songs into the encoder-decoder model with attention to generate personalized reasons. Experiments show the effectiveness of

our method, and its deployment on the XiaoIce chatbot improves the click-through rate substantially.

The rest of this article is organized as follows: We start with an overview of related work in Section 2. Section 3 presents the details of our approach. Experiment results and discussions are given in Section 4, and Section 5 concludes this article.

2 RELATED WORK

Our work is related to two groups of work: explainable recommender systems and conversation systems.

2.1 Explainable Recommender Systems

Instead of simply presenting recommended items to the user in traditional recommender systems [9–30], some researchers have tried to mine the reasons behind recommendations [27, 31–41]. For example, Wang et al. [42] propose a new perspective that leverages knowledge graphs for explainable recommendation. Explainable recommender systems can be grouped into four categories: (1) User-based or item-based explainable systems ("You may like this item because users similar to you like it" and "You may like this item because you liked these similar items"); (2) Social-based explainable systems ("You may be interested in this item because your friend likes it"); (3) Feature-based explainable systems ("You may like this item because it has these features"); (4) Review-based explainable systems ("I love this series. I can't wait for the next book. I love the characters and the story line. I was so glad that the story was a little longer . . . ").

The fundamental idea of collaborative filtering is looking for similar users and recommending the items they are interested in. Thus, it is natural to explain a recommendation by "Customers who bought this item also bought . . . ". Schafer et al. [43] state that a recommender system would be used to explain to a user what type of thing a product is, such as "this product you are looking at is similar to these other products that you have liked in the past," which is the main idea of item-based collaborative filtering [16, 17]. Item-based explanations are usually more intuitive for users to understand, because users are usually familiar with those products that he/she has purchased before. However, item-based explanations are designed by predefined forms, and a more natural explanation form could be a piece of free-text that explains to the user in natural language [1]. Thus, such common reasons are less appealing in conversation scenarios.

Among social-based explainable systems, Wang et al. [44] generate social explanations such as "A and B also like the item." They propose generating the persuasive social explanation by recommending the optimal set of users to be put in the explanation. Kouki et al. [32] present explanations such as "Your friend Cindy likes this item" by leveraging information from social networks. However, in the conversation applications with private settings, there is no available social network information between users.

Feature-based explanations can also be seen as content-based explanations. They provide recommendations by matching user preference with the available item content features [9, 10, 27, 34, 45–48]. For example, Zhang et al. [45] use phrase-level sentiment analysis to mine the explicit features of items and the corresponding sentiment polarity of the user. They propose Explicit Factor Models (EFM) to fit user-item ratings by the latent representations. The features can also be displayed in different forms [47–49]. Chen and Wang [34] extract informative features from item descriptions and then utilize a template based sentence to show the strengths and weaknesses of items such as "They have higher ratings for effective pixels and weight, but are rated lower for price" for a camera. Wu and Ester [47] use word cloud explanation for hotel recommendation generated based on latent topic modeling with textual reviews. Hou et al. [48] use radar charts to

explain why an item is recommended to a user and why others are not recommended. In our conversation application, our explanation is not that formal but as casual as dialogs between friends.

In review-based explainable systems, Chen et al. [37] introduce an attention mechanism to explore the usefulness of reviews and propose a neural attentional rating regression model for recommendation. It can not only predict ratings, but it also learns the usefulness of each review simultaneously. The obtained highly useful reviews provide review-level explanations. The limitation is that it can only rank existing reviews for existing items. Our proposed method can generate new reasons for new items, which alleviates the cold-start issue and data-sparsity issue. Li et al. [50] propose a deep-learning-based framework called NRT that can simultaneously predict precise ratings and summarize the massive reviews to generate abstractive tips. Costa et al. [51] attempt to summarize items' review sentences as explanations by learning large-scale review data. However, in conversation scenarios, the summaries of user reviews are too long to be similar with casual chatting between friends. That is why summaries cannot directly used here.

In addition, in contrast to the above studies, we aim at providing a recommendation reason that is relevant to both the recommended song and to the user in conversation applications in such a style to make the user feel as if they are receiving a recommendation from their friend.

2.2 Conversation Systems

Conversation systems include task-oriented dialog systems [52, 53], which are built for specific tasks such as ordering food, and non-task-oriented systems or chatbots [54], which aim to mimic human-like conversations in open domains. The recent increase in the availability of conversational data has enabled rapid development of chatbots and dialog systems based on data-driven approaches. One approach to this is retrieval-based [55, 56]: match a user input with existing question-and-answer pairs to retrieve the most appropriate responses. Another approach is generation-based [7]: learn a response generation model within a Statistic Machine Translation (SMT) framework from large-scale social conversation data.

A common generation-based approach treats posts as user inputs and comments as responses. Response generation can be regarded as translation from posts to comments. Ritter et al. [57] find that SMT techniques are more suitable than information retrieval approaches for the task of response generation. A basic sequence-to-sequence model proposed by Cho et al. [4] consists of two recurrent neural networks (RNNs): an encoder that processes the input and a decoder that generates the output. Multi-layer cells have been successfully used in sequence-to-sequence models by Sutskever et al. [5]. To allow the decoder more direct access to the input, Bahdanau et al. [6] introduce an attention mechanism. Shang et al. [7] propose a neural network-based response generator for Short Text Conversation using the encoder-decoder framework. Our work utilizes a similar neural model.

Task-oriented or not, when the system would like to recommend an item to the user during a conversation, the recommendation is unlikely to be successful unless the system explains why it is recommending that item to that specific user. To the best of our knowledge, existing studies have not addressed this particular research problem.

3 OUR APPROACH

In this section, we first define the problem and describe our solution to generating a reason that explains why a particular user should listen to a particular song.

3.1 Problem Formulation and System Overview

In the problem of reason generation for explainable recommendation in conversation applications, we assume that a user U has asked the chatbot to recommend a song and that a recommendation

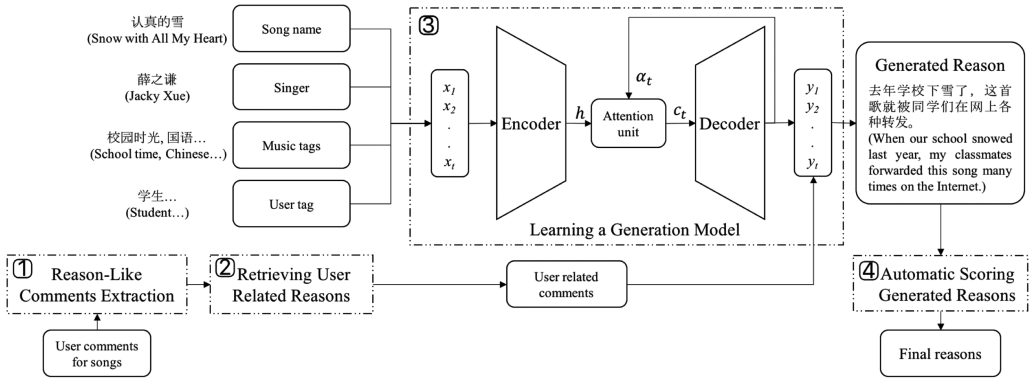

Fig. 2. The flowchart of our proposed solution to generate personalized reasons for a particular song and a particular user.

algorithm has returned a song S that is appropriate for the user. Our target is to generate a reason as a sequence of words $Y = (y_1, y_2, \ldots, y_M)$ to explain why the user U should listen to the song S. The generation model maximizes the probability of Y conditioned on S and U: $p(Y|S, U)$.

To train a generation model, we need a dataset $\mathcal{D} = (S_i, U_i, Y_i)_{i=1}^{N}$, but there is no existing data of this kind. We propose extracting (S_i, Y_i) from comments that users post to songs on a music website. Then, we use a method to retrieve relevant (S_i, Y_i) for U_i and thereby to compose the required dataset 4. For example, in Figure 2, S is *Snow with All My Heart,* sung by Jacky Xue, and it has some tags such as *school time*, *mandarin*, and so on. A user U has the tag *student*, which represents his/her current status and is mined from the user's chat logs. We manage to connect them with the target reason "When our school snowed last year, my classmates forwarded this song many times on the Internet."

When we concatenate the representations of U with S, we can regard them as the source sequence X of a statistic machine translation model. The target sequence is Y. Hence, we apply an encoder-decoder framework with attention to maximize the generation probability $p(Y|X)$. Sometimes, generated texts contain errors that hurt their readability. Thus, we also propose some automatic scoring metrics to filter out such noise.

3.2 Reason-like Comments Extraction

We crawl 80M comments of 1.2M pieces of music from NetEase Music.[2] Most of them are songs. We find that the comments of the following categories are useful as recommendation reasons:

Fact and Opinion. Comments of this kind focus on certain facts of the music and its related entities, e.g., "The music reminds me of Totoro and sounds like Joe Hisaishi." The facts are sometimes described with opinions, e.g., "The rapping part is so difficult that I feel my mouth crooked." These comments are informative recommendation reasons, because they provide insight into the commented music.

Emotion and Experience. Users may express their personal emotions and past experiences when listening to the music, e.g., "I was about to sleep but now too excited to sleep." and "I used to like listening to this song more than watching TV when I was a child." These comments are also good recommendation reasons, as they can easily resonate with other

[2] http://music.163.com/.

users. It should be noted that some comments of this kind are too personal and thus not suitable for a chatbot to utter.

Joke and Story. These comments consist of jokes and stories made up by users, e.g., "Only three persons in this world think you're beautiful: your mom, your dad, and James Blunt." As recommendation reasons, they arouse users' curiosity by making fun of certain facts. However, some of these comments are too long or too detailed for a chatbot and should be discarded.

On a music website, users also converse with each other. Comments within such a conversation are not suitable for recommendation, because they highly depend on context, while recommendation reasons are expected to be self-explanatory. Spam is also excluded for recommendation.

Based on the above observations, we design a two-phase method to extract recommendation reasons from the crawled comments.

In the first phase, we extract seed comments with two phrase lists, namely the allowed list and the blocked list. A comment is considered a seed comment if it contains a phrase in the allowed list (e.g., listen to this song, lyrics, and BGM) and does not contain any phrase in the blocked list (e.g., vote if your like, free music, and playlist). For the length of the seed comments, we set the number of words ranging from 10 to 50. The comments that contain few words do not have enough information to be a reason. Additionally, if the comments have too many words, they are so long that users generally have no patience to read. Comments excluded for seed comments are all considered as non-seed comments. We craft the rules in an iterative manner. In each iteration, we partition the crawled comments into seed comments and non-seed comments. We then look for new rules that improve the partition. The rules make use of various textual features, e.g., length of comment, usage of punctuation and numbers, language, and repeating phrases. We extract 4.4M seed comments in this phase. However, as seed comments only cover a fraction of actual recommendation reasons and might be biased, we apply the second phase to fix this issue.

In the second phase, we train a classifier that takes a comment as input and predict if it can be used as a recommendation reason. We take seed comments as positive samples and a random equal-sized set of non-seed comments as negative samples. Features for a comment consist of the features used in the first phase and a feature vector containing character uni-grams. We try several popular classifiers, such as logistic regression, decision trees, and SVMs. It turns out that logistic regression yields slightly better performance than others. We extract an additional 0.5M comments in this phase.

Finally, 4.9M comments on 0.4M pieces of music are extracted as recommendation reasons. To evaluate quality, we randomly sample 2K comments from the crawled comments and ask 10 volunteers to label if a comment can be seen as a recommendation reason. 33% of comments are labelled as recommendation reasons. The precision and recall of our method are 88% and 31%, respectively. Since our goal is collecting comments to train the generation model, precision is much more important than recall; hence, we believe that this level of performance is sufficient.

3.3 Retrieving User-related Reasons

As mentioned in Section 3.1, a user U is represented as a set of user tags. Given a user with a tag, e.g., *student*, how can we obtain recommendation reasons that are suitable for him/her?

NetEase Music does not provide user tags. Hence, we rely on the user tags that we mine from chat logs. The user tags are a set of predefined keywords covering users' status and interests. Status tags describe the current state or lifestyle of a user, e.g., *break-up*, *student*, and *sleep late*. These tags are extracted from our internal user profiling system. The first two tags are inferred by two binary classifiers: one is whether a user is a student and the other is whether a user is lovelorn. The

classifiers were built based on text features. For the last tag, users who chat more often at night are tagged as *sleep late*. Interest tags represent if a user likes or dislikes certain kind of entities, e.g., color, food, and type of music. We extract entities mentioned by users and corresponding sentiment. Entities with positive sentiment are considered as user interest. We manually select the tags those are related to music and life style for this work. We find these tags cover most of our users. Also, we filter out user tags that are not suitable for song recommendation: for example, those that express dislike (e.g., *dislike working*). The remaining user tags are used for searching NetEase Music to construct the training data (S_i, U_i, Y_i).

To each user tag, we apply query expansion to enhance recall. Specifically, we first project the user tag into a pre-trained Word2Vec model [58] and discover similar words in terms of cosine similarity. For example, given the user tag *student*, we discover similar words including *teacher*, *worker*, *campus*, and *study*. Next, we manually review the expanded words and filter the irrelevant ones, such as *worker* in the above example. Finally, we retrieve reasons by using these queries.

A user is represented by multiple tags. In the retrieving user related reasons step, a random tag is selected from tags assigned to the user. As multiple songs would be recommended in the dialog, it would be boring if the recommendation reasons concentrate on only one or two user tags. To avoid this, picking a random tag for each time is a simple but effective method.

In our experiments, we sample 22 user tags, and the query length after expansion was 14.5 words on average, ranging from 6 to 35.

3.4 Learning a Generation Model

We choose the recurrent attention network to model the generation probability $p(Y|X)$, where $X = (S, U)$. For S_i, we have music tags that are mined from playlists (denoted by $(t_{i,1}, \ldots, t_{i,L})$. We use the top five music tags, i.e., $L = 5$), its singer names (denoted by g_i), and song names (denoted by a sequence $(q_{i,1}, \ldots, q_{i,K_i}$, where K_i is the number of words in song names). Each user has multiple tags, but for each training triplet, just one user tag is utilized as mined from chat logs: $U_i = (u_{i,1})$. Finally we concatenate S and U to compose X:

$$X_i = \{t_{i,1}, t_{i,2}, t_{i,3}, t_{i,4}, t_{i,5}, g_i, u_{i,1}, q^v_{i,1}, \ldots, q^v_{i,K_i}\}, \tag{1}$$

where v indicates the parts with variable length. We arrange the fixed-size features before the variable-length features such as song names, because the encoder-decoder framework with attention requires the input to be sequential to decide which parts of the input to pay attention to. We keep the positions of fixed-length features steady to better retain their corresponding meaning and weights.

An encoder reads X into vector h. Here, a bidirectional RNN [59] is utilized. Given an input sequence with ordering from x_1 to x_T, the forward RNN calculates a sequence of its forward hidden states $\{\overrightarrow{h_1}, \ldots, \overrightarrow{h_T}\}$. Meanwhile, reversing the input as the order from x_T to x_1, the backward RNN calculates a sequence of its backward hidden states $\{\overleftarrow{h_1}, \ldots, \overleftarrow{h_T}\}$. Then, we obtain the final hidden states by concatenating them as $h_j = \{\overrightarrow{h_j}, \overleftarrow{h_j}\}$, which saves the summaries of both the preceding words and the following words.

The attention mechanism aims to find the parts of inputs that should be focused on. Thus, the context vector c is calculated by a weighted sum of the final hidden states:

$$c_t = \sum_{j=1}^{T} \alpha_{t,j} h_j, \tag{2}$$

where h_j is the hidden state of the bidirectional RNN and the weight $\alpha_{t,j}$ is computed by

$$\alpha_{i,j} = \frac{\exp(e_{i,j})}{sum_{k=1}^{T_x} \exp(e_{i,k})}, \tag{3}$$

where

$$e_{i,j} = a(s_{i-1}, h_j), \tag{4}$$

is an alignment model that scores how well the inputs around position j and the output at position i match. The score is based on the RNN hidden stat s_{i-1} and the jth annotation h_j of the input sentence. a is a feedforward neural network that is jointly trained with all the other components of the proposed system. More details can be found in Reference [6].

Given the predicted preceding words $\{y_1, y_2, \ldots, y_{t-1}\}$, context vector c_t, and the RNN hidden state s_t, the decoder calculates a probability of the next word y_t:

$$p(y_t|x_1, x_2, \ldots, x_T, y_1, y_2, \ldots, y_{t-1}) = g(y_{t-1}, s_t, c_t), \tag{5}$$

where $g(\cdot)$ is a softmax activation function and

$$s_t = f(y_{t-1}, s_{t-1}, c_t). \tag{6}$$

Here, f is a non-linear function. It can be a logistic function, an LSTM unit, or a GRU (Gated Recurrent Unit). We use a GRU in our experiments.

Finally, we leverage a beam search strategy to generate the reason given by $x_1, x_2, \ldots, x_T$. For the first node, we go through the softmax activation function and calculate the Top K candidates by $p(y_1|x_1, x_2, \ldots, x_T)$. Then for each candidate y_1, we calculate its next sequences by

$$y_t = \arg\max_{y_i : i \geqslant 2} p(y_i|x_1, x_2, \ldots, x_T, y_1, y_2, \ldots, y_{i-1}), \tag{7}$$

until y_t is equal to the "End of Sentence" symbol. Finally, we randomly select one from the K generated candidates to ensure diversity of our output.

3.5 Scoring-generated Reasons

To guarantee that a generated reason is easy to read and understand, we need to filter out noisy text. Thus, we propose learning a linear regression function of a score based on generation probability, a score based on N-gram language models, a score based on POS (Part-Of-Speech) RNN language models, and a score based on dependency parsing. All scores are re-scaled into 0 to 1 before combination. Finally, we filter out reasons that do not pass a threshold.

3.5.1 Generation Probability. Generation probability measures how well a probability model predicts a sample. A low generation probability indicates that the sample does not fit into our model well. Given a sample $Y = (y_1, y_2, \ldots, y_M)$, it is defined by:

$$S_{gp} = \sum_{i=1}^{M} \log(p(y_i|x_1, \ldots, x_T, y_1, \ldots, y_{i-1}))/M, \tag{8}$$

where y_i is the predicted word by our generation model.

3.5.2 N-gram Language Models. We leverage n-gram language models to measure whether a word or a phrase or a combination is correct. For a sentence with length M, the score based on n-grams (from bi-grams to five-grams) are calculated as follows:

$$S_{ng} = \sum_{n=2}^{5} \sum_{i=n}^{M} \log(p(w_i|w_{i-n+1:i-1})) + \sum_{j=1}^{5} \sum_{i=j+1}^{M} \log(p(w_i|w_{i-j})). \tag{9}$$

where w_i is the ith word of the sentence. The first part is calculated based on ordinary n-grams. In the second part, we use a biterm language model [60], which counts word pairs separated by zero or more words, to measure skip n-grams in the generated text. The two language models are trained on the comments in our training set.

3.5.3 POS RNN Language Model. We train an RNN language model with POS (Part-Of-Speech) tagged training data to describe the grammar correctness of a sentence. We utilize the public API of Language Technology Platform[3] (LTP) for POS parsing [61, 62]. The RNN language model is implemented with a two-layer LSTM neural network. Then, the model can calculate the generation probability over the POS tag sequence as follows:

$$S_{pos} = \sum_{i=1}^{M} \log(p(pos_i|pos_1, pos_2, \ldots, pos_{i-1}))/M, \tag{10}$$

where pos_i is the POS tag of the predicted word.

3.5.4 Dependency Parsing. We leverage the syntax dependence graph generated by using an LTP API to analyze the constituents of a sentence. The dependence graph shows the syntactic relations between different constituents. We collect the frequency of each directed pairwise relation between two word nodes w_a and w_b as $p(r|w_a \rightarrow w_b)$, where r is the syntactic relation between w_a and w_b, from our training corpus. Then, given a sentence, we calculate a score as follows:

$$S_{dp} = \sum_{\forall\{w_a \rightarrow w_b\}} p(r|w_a \rightarrow w_b)/E, \tag{11}$$

where E is the number of directed pairwise nodes in the generated syntax dependence graph of the sentence.

4 EXPERIMENTS

In this section, we describe our dataset, evaluation of auto-scoring methods, and our offline and online experiments to compare different methods.

4.1 Dataset

Our training data are crawled from NetEase Music, one of the most popular music websites in China. As described in Section 3.2, we extract 4.9M reason-like comments for 0.4M songs. Among the crawled songs, we manage to mine song tags for about 7,932 songs. However, we obtain 22 user tags to retrieve personalized reasons for our experiments. By joining the three sets, we finally obtain a dataset of the form (S_i, U_i, Y_i), which contains 2,778 songs (S_i), 206K reasons (Y_i) corresponding to 22 user tags (U_i). These tags include *student*, *electronic music*, *ballad*, *lovelorn*, *sleep late*, *pure music*, *anime*, *girly girl*, *lolli*, *fashionista*, *food aficionado*, *study*, *rain*, *Chinese classes*, *math*,

[3] https://www.ltp-cloud.com/intro/en/.

Table 1. Comparing Different Automatic Scoring Methods in Ranking Fluent Generated Reasons

Scoring Method	nDCG@1	nDCG@5	nDCG@10
Generation Probability (*gp*)	0.901	0.933	0.967
N-gram Language Models (*ng*)	0.920	0.936	0.968
POS RNN Language Model (*pos*)	0.811	0.857	0.929
Dependency Parsing (*dp*)	0.761	0.845	0.924
Regression (all)	0.936	0.945	0.974
Regression (w/o *dp*)	0.940	0.945	0.974
Regression (w/o *ng*)	0.899	0.928	0.965
Regression (w/o *pos*)	0.939	0.945	0.973
Regression (w/o *gp*)	0.930	0.934	0.968
Regression (*dp* & *ng*)	0.922	0.932	0.968
Regression (*dp* & *pos*)	0.828	0.869	0.935
Regression (*dp* & *gp*)	0.895	0.927	0.964
Regression (*ng* & *pos*)	0.924	0.938	0.969
Regression (*ng* & *gp*)	**0.943**	**0.946**	**0.974**
Regression (*pos* & *gp*)	0.904	0.934	0.968

English, *history*, *basketball*, *dog*, *reading*, *painting*, and *singing*. We use this dataset for training our reason generation model.

4.2 Evaluation of Automatic Scoring Methods

We collect human ratings from 1 (bad) to 3 (good) on fluency for 1K reasons related to 30 randomly selected songs. We conduct 5-fold cross validation over the data to investigate the effectiveness of features used for automatic scoring. Upon test data, we rank reasons for each song by their corresponding scores and then evaluate the list by nDCG [63].

As Table 1 shows, when we use each individual feature to rank reasons, the features based on N-gram language models (*ng*) and generation probability (*gp*) are the best two. They are considerably better than the features based on POS RNN language model (*pos*) and dependency parsing (*dp*) in terms of nDCG@1, 5, and 10. When we use linear regression to combine the four features, the nDCG@1 can be further improved to 0.936, and the nDCG@10 can be improved to 0.974.

To clarify the contributions of different features, we built four regression models by removing one individual feature at a time. The worse such a model is, the greater contribution the removed feature has. According to Table 1, the *ng* feature leads to the biggest drop, from 0.936 to 0.899 in nDCG@1. This indicates that such a feature is the most useful in scoring a fluent reason. The *gp* feature is the second-most useful. Surprisingly, by removing the *dp* and *pos* features, the nDCG@1 slightly increases. This indicates that adding the two features may hurt the effectiveness of our regression model.

We conduct additional experiments to combine any two features in regression. Results in Table 1 confirm that the model based on the *ng* and *gp* features performs the best in all metrics. The model also beats the model based on all four features in nDGC@1, 5, and 10. Thus, we use the best regression model as a scoring function in remaining experiments.

After we get the scores of generations, we filter out generations that do not pass a threshold. To find the threshold, in the training set, we set the generations that only get one human rating as the negative samples (noisy texts) and set the generations that get two or three human ratings as

the positive samples. Then, we draw the Precision-Recall curve to find the break-even point (BEP). At last, we get the threshold (i.e., 1.85) at the break-even point.

4.3 Offline Comparison of Reasons

4.3.1 Evaluation Metrics. We conduct offline evaluation to compare the effectiveness of different methods in generating reasons for personalized song recommendation. We randomly sample 30 songs that are not included in training data for testing purposes. They are different from those 30 songs used in Section 4.2. For each method for comparison, we collect the top five results to make a pool for assessments.

As user satisfaction is somehow difficult to explain or decompose, for a given song and a reason, we first ask six assessors to independently give an overall rating on whether the text is bad (i.e., rating 1), acceptable (i.e., rating 2), or attractive (i.e., rating 3) as a reason. Then, we ask them to give detailed ratings ranged from 1 (bad) to 3 (good). The detailed criteria are:

- *Fluency*. The generated reason is easy to read and understand as a natural language text.
- *Relevance*. It is relevant to the recommended song, so the user can understand why that particular song is recommended.
- *Personalization*. It is relevant to the personality, interests, and situation of that particular user.
- *Overall*. It is the overall rating that presents the generated reason could be an attractive reason.

4.3.2 Compared Methods. In offline evaluation, we compare the following methods:

FM. Factorization Machines (FM) [64] are widely used in recommendation systems due to their effectiveness and rich functionality. We build a tensor of (song, user tag, reason) from our training dataset and leverage FM to predict the recommendation probabilities of comments. The top five comments are returned as recommendation reasons.

DeepFM. This is a state-of-the-art method [8] in the area of recommender systems. It combines the power of factorization machines for recommendation and deep learning for feature learning in a new neural network architecture. We build a tensor of (song, user tag, reason) as for FM method and leverage DeepFM to predict the recommendation probabilities of comments.

Retrieval. The comments retrieved in Section 3.3 are used directly as recommendation reasons of associated music.

Generation w/o userTag. To assess the effect of user tags, we consider a simplified version of the proposed method. It just uses $\{S_i, Y_i\}$ to train a generator without user tags and without automatic scoring.

Generation. This is our proposed method, which utilizes a generation model to produce the personalized reasons with user tags for given songs, but without using any automatic scoring function.

Generation w/Scoring. This is the ranked results of *Generation* by using the automatic scoring method, where we adopt the best regression model based on N-gram language model feature and generation probability feature.

For the experimental settings of our methods, RNN size of the encoder and decoder is 512. We set batch size as 512 and epochs as 20. Word embedding is 620. The max length of input is 40 and the max length of output is 20.

Table 2. Comparing Methods of Obtaining Recommendation Reasons by Human Ratings

	Method	FM	DeepFM	Retrieval	Generation w/o userTag	Generation	Generation w/ Scoring
Top-1	Fluency	2.80	2.80	2.83	2.60	2.67	**2.93**
	Relevance	1.97	1.97	1.90	1.80	1.93	**2.03**
	Personalization	1.97	2.00	2.30	2.00	2.30	**2.33**
	Overall	1.90	1.90	1.93	2.00	1.97	**2.07**
Top-3	Fluency	2.84	2.84	2.81	2.40	2.86	**2.90**
	Relevance	1.91	1.91	1.91	1.93	2.03	**2.06**
	Personalization	1.94	1.98	2.26	1.87	2.32	**2.33**
	Overall	1.83	1.83	1.92	1.87	2.03	**2.06**
Top-5	Fluency	**2.88**	**2.88**	2.85	2.32	2.78	2.85
	Relevance	1.91	1.91	1.89	1.92	1.97	**2.05**
	Personalization	1.93	1.96	2.28	1.84	2.25	**2.29**
	Overall	1.85	1.86	1.90	1.80	1.97	**2.05**

The numbers are average ratings.

Table 3. Tukey's HSD Test Results between the Four Methods in Offline Evaluation Upon Top Five Reasons

Method pair		Mean-diff ($X - Y$)			
X	Y	Fluency	Relevance	Personalization	Overall
FM	DeepFM	−0.0000	−0.0089	−0.0113	−0.0083
Retrieval	FM	−0.0267	0.1022	0.1600*	0.0883
	DeepFM	−0.0267	0.0933	0.1467	0.0800
Generation w/o userTag	FM	−0.5600*	−0.2133	−0.3267*	−0.1733
	DeepFM	−0.5600*	−0.2222	−0.34*	−0.1817
	Retrieval	−0.5333*	−0.3156*	−0.4867*	−0.2617*
Generation	FM	−0.1000	0.0933	0.1067	0.0983
	DeepFM	−0.1000	0.0844	0.0933	0.0900
	Retrieval	−0.0733	−0.0089	−0.0533	0.0100
	Generation w/o userTag	0.4600*	0.3067*	0.4333*	0.2717*
Generation w/ scoring	FM	−0.0267	0.1556*	0.1633*	0.1650*
	DeepFM	−0.0267	0.1467*	0.15	0.1567*
	Retrieval	0.0000	0.0533	0.0033	0.0767
	Generation w/o userTag	0.5333*	0.3689*	0.4900*	0.3383*
	Generation	0.0733	0.0622	0.0567	0.0667

The numbers are mean-diff between methods X an Y. "*" means that the difference is statistically significant at $\alpha = 0.05$.

4.3.3 Performance Comparison. Table 2 shows the comparison result based on human ratings and Table 3 shows the corresponding Tukey's HSD (Honestly Significant Difference) test result. Overall, our proposed generation method with scoring is the best; it is the only method that always obtains scores above 2.0 on *Overall* metric, which means it is above being acceptable. Tukey's HSD test shows the gain of our *Generation with Scoring* method over *FM* is statistically significant in terms of *Relevance*, *Personalized*, and *Overall*. It also shows the gain of our *Generation with Scoring* over *DeepFM* is statistically significant in terms of *Relevance* and *Overall*. Although the *Fluency* of the *Generation* method is sometimes worse than *FM*, *DeepFM,* and *Retrieval*, which use existing comments, our automatic scoring method can select more fluent reasons and improves *Fluency* from 2.78 to 2.85 at the top five results. When our model discards user tags, the performance on

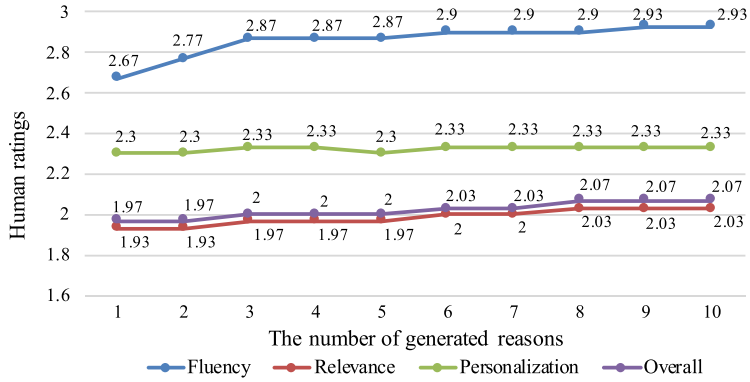

Fig. 3. The impact of the number of generated reasons on the Top-1 result of our *Generation w/Scoring* method.

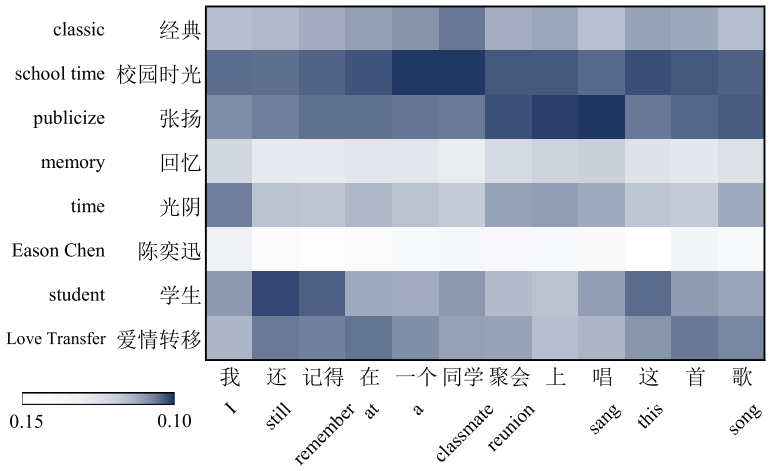

Fig. 4. Visualization of attention weights between an input (S, U) and the output reason Y. They reflect the importance or contribution of a word in generating a reason. Darker color means more important.

all metrics, in particular *Personalizaton*, at the top five results in significant declines. This indicates that our proposed generation method is not only user-sensitive but also performs better in relevance and fluency.

We also show the impact of the number of generated reasons on the Top-1 result of our *Generation w/Scoring* method. As shown in Figure 3, the performance increases with the number of generated reasons. It demonstrates the effectiveness of the proposed automatic scoring methods and indicates that after generating a certain number of reasons (6 in Figure 3), the performance is stable. It implies our method is stable in terms of quality, and we don't need to generate too many reasons to find a good one.

4.3.4 Visualization of Attention Weights. Figure 4 visualizes the attention over a particular generated reason. The song "Love Transfer," by Eason Chen, is tagged with *classic*, *school time*, ..., and *time*. The user is tagged with *student*. Such information composes X as an input. Our model generates a reason Y: "I still remember at a classmate reunion I sang this song." The color of each cell represents the attention weight between a word in X and a word in Y. It can be observed,

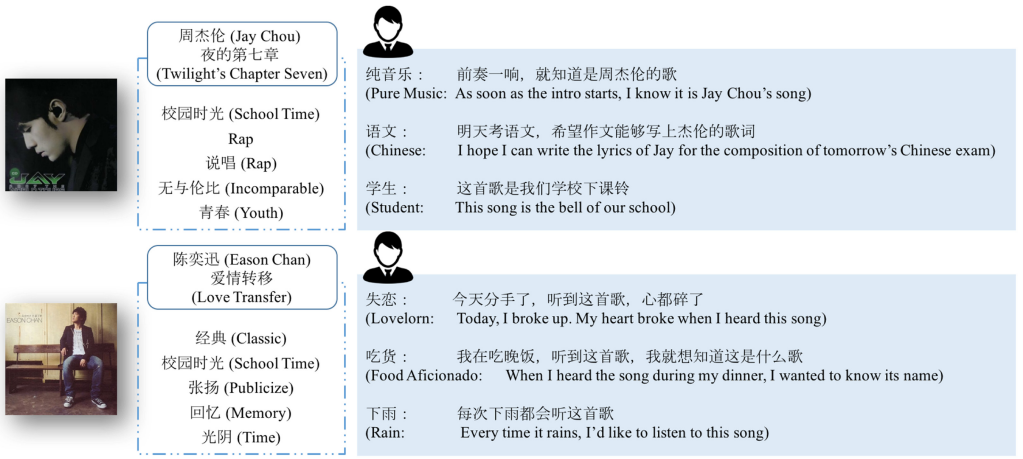

Fig. 5. Our method generates reasons for users with different tags and two songs in the training set. For each song, the left text box shows its singer, song name, and five music tags. The right box contains three user tags and generated reasons for them.

for example, that the attention weight for the pair *school time* and *classmate* and that for the pair *publicize* and *sang* are high, as the words are semantically related. If we sum the weights for each input word (i.e., each line), then two song tags, i.e., *school time* and *publicize*, the song name "Love Transfer" and the user tag *student* are the most salient. This shows that our proposed model can utilize both song information and user tags in generating reasons.

4.3.5 Effectiveness on Known Songs and New Songs. When looking into detailed cases, we find that our method can generate some new and interesting recommendation reasons for both known songs and new ones. When a song is included in training data, e.g., *Twilight's "*Chapter Seven," our approach can generate some reasons that have not appeared in training data. For example, all the generated reasons shown in Figure 5 have no exact same occurrences in existing data. The singer of *Twilight's "*Chapter Seven" is Jay Chou, and its extracted music tags are *school time*, *rap*, and so on. For a user whose tag is *pure music*, our model generates the reason: "As soon as the intro starts, I know it is Jay Chou's song." For a user whose tag is *(liking) Chinese*, our model generates the reason: "I hope I can write the lyrics of Jay for the composition of tomorrow's exam." We show more examples in Figure 5. It can be observed that our model can generate novel and personalized reasons for known songs.

Our method generally works well for a new song that does not have any existing comments, because the singer and/or the tags of that song may not be new. As shown in Table 4, the new song "I Love You but Goodbye," by the singer Pushu, is tagged with *ballad*, *college years*, *rock*, and so on. For a user whose tag is *lovelorn*, our model generates the reason: "Every time I listen to this song, I think of my first love." Although there is no specific comment on the song, such a reason may have resonance for the user who has just been bereft of love. For a user who loves ballads, our model generates the reason: "Every time I listen to this song, I have a very quiet feeling." The examples in Table 4 indicate that our method can generate recommendation reasons that apply to songs outside the training data.

4.3.6 Effectiveness on Multiple Tags. During training, there is no issue of multiple tags, because we construct our training data by retrieving relevant reasons to a specific tag. Our dataset consists of (song, user tag, reason) triplets. When we apply our proposed approach online, a user can have

Table 4. Our Method Generates Reasons for Users with a Single Tag in the Test Set

Input	User Tag	Generated Reason
Singer: 朴树 (Pushu) **Song name**: 我爱你，再见 (I love you but goodbye) **Tags**: 民谣 (Ballad) 大学时光 (College Years) 校园时光 (School Time) 光阴 (Time) 摇滚 (Rock)	失恋 (Lovelorn)	分手后，听着这首歌，感觉自己也是醉了 (After breaking up, listening to this song, I feel drunk) 每次听这首歌都会想到初恋 (Every time I listen to this song, I will think of my first love)
	电音 (Electronic Music)	每次听到这首歌都会有一种震撼的感觉 (Every time I hear this song, I have a feeling of shock) 每次听到这首歌都会热血沸腾 (Every time I hear this song, I feel my blood boils with indignation)
	民谣 (Ballad)	每次听这首歌都会有一种很安静的感觉 (Every time I listen to this song, I have a very quiet feeling) 很喜欢这首歌，很喜欢民谣 (I love the song very much and I love ballad)
	学生 (Student)	以前学校每天中午都会放这首歌 (In the past, the school played this song at noon every day) 校园十佳歌手，我就听了这首歌。[可爱] (I heard this song at the competition of top ten singers of campus. [Cute])
	晚睡 (Sleep Late)	每天晚上睡觉前听这首歌，越听越有感觉，越听越有感觉 (Listen to this song before going to bed every night. The more you listen, the more you feel. The more you listen, the more you feel.) 这首歌是我最喜欢的一首歌，晚安 (The song is my favorite. Good night.)

multiple tags. We will randomly select one tag to generate a reason. As a result, such a simple strategy can diversify the reasons when we recommend different songs to the user. Although so far we lack such data, we try to use our model trained by one user tag to generate reasons with multiple user tags. Table 5 shows the generated reasons for users with multiple tags. It indicates our model can be utilized to generate reasons with multipl tags, but some of the generations seem like a simple mixing of different reasons and are not fluent.

4.3.7 Examples of the Generated Reasons. Here we show some generations of the compared methods in Table 6. These generations are mostly fluent, but the generations of FM and DeepFM methods actually do not match the designed user tag. Most of the generations of Retrieval method can match the user tag, but some noise may be introduced. For example, when a user tag is designed as Study, Retrieval method introduces "Japanese," which is in conflict with the truth that this song is Korean. However, our method generates short reasons in trying to avoid introducing irrelevant content, and its generations can match user tags. Furthermore, our model also may generate bad cases as existing methods. Some generated reasons are not correct sentences or they cannot be understood. For example, "A single man listening to this is truly truly." That is consistent with our evaluation on fluency. The generation model can produce some new sentences that do not exist in the training data, whereas the generation is not perfect all the time.

4.4 Online Evaluation

Our objective is to increase the click-through rate of our recommended songs. The aforementioned offline experiments rely on assessors who are not the actual users of our recommendation service. Therefore, we conduct an online evaluation of methods by comparing the Click-Through Rate (CTR), i.e., the number of clicked songs divided by the number of songs shown to users. We

Table 5. Our Method Generates Reasons for Users with Multiple Tags in the Test Set

Input	User Tags	Generated Reason
Singer: 朴树 (Pushu) **Song name**: 我爱你，再见 (I love you but goodbye) **Tags**: 民谣 (Ballad) 大学时光 (College Years) 校园时光 (School Time) 光阴 (Time) 摇滚 (Rock)	学生，失恋 (Student, Lovelorn)	以前学校每天中午都会放这首歌，现在想想，我的初恋 (In the past, the school played this song at noon every day, now it is recalling my memory, my first love) (After breaking up, listening to this song, I feel drunk) 今天学校放了这首歌，我就知道这首歌是我的初恋 (As soon as the school played this song today, I know this song is my first love)
	晚睡，电音 (Sleep Late, Electronic Music)	晚上睡觉前听这首歌，越听越带感 (Listen to this song before going to bed. The more you listen, the higher you feel.) 半夜听这首歌，感觉好爽 (Listen to this song at midnight, I feel so high.)
	民谣，电音 (Ballad, Electronic Music)	每次听这首歌都会有一种震撼的感觉 (Every time I listen to this song, I have a very quiet feeling) 民谣的歌词都是摇滚的 (All the lyrics of ballad songs are rock and roll)
	失恋，晚睡 (Lovelorn, Sleep Late)	听完这初恋给我听的歌，现在听着听着就想哭了 (After listening to the song recommended by my first love, now I feel crying while listening it.) 听完这首歌就睡觉了，晚安 (Go to bed after listening to this song. Good Night)

generate 40K reasons for CTR comparison on about 1,400 recommended songs that are the most popular songs in our service.

4.4.1 Compared Methods. We conduct online experiments by deploying the following reason-generation methods:

Chat Responses. We use the query that a user asks for a song recommendation (e.g., "I fell out of love. What should I listen to?" in Figure 1) to retrieve a chat response as the recommendation reason.

Mined Reason-like Comments. We use one of the mined reason-like comments (see Section 3.2) that are posted to the recommended song as the reason.

Manually Selected Reason-like Comments. We ask assessors to review all the mined reason-like comments and select those that are acceptable recommendation reasons. We then randomly draw one from them. If we always choose the reason with the highest confidence score, then the conversation does not have diversity and easily make users boring. The random mechanism is a basic operation for XiaoIce. Thus, we randomly drew one reason from the 10 generated reasons.

Generated Reasons w/o userTag. We use $\{S_i, Y_i\}$ to train our generator. Given a song, we generate 10 reasons offline and randomly draw one from them.

Generated Reasons w/Scoring. Given a song and a user tag, we generate 10 reasons offline with the proposed method of Generation w/Scoring. We then randomly draw one from them.

4.4.2 Performance Comparison. We collect user activities on the XiaoIce chatbot from July 28, 2017, to Jan. 31, 2018. Then, we compute the CTRs for the compared methods based on the same songs. We hide the number of impressions due to confidentiality concerns. The results are shown in Table 7.

Table 6. Some Generations of the Compared Methods

Input	User Tag	Methods	Generated Reason
Singer: 少女时代 (Girls' Generation) **Song name**: Party **Tags**: 韩文 (korean) 少女 (Teenage girl) Loli 小萝莉 (little loli) 日韩 (Japan or South Korea)	动漫 (Anime)	FM	听这歌的时候，正开着车，下起了暴雨。忽然觉得好带感 (When I heard this song, I was driving in the car and it was raining. Suddenly I felt a good feeling.)
		DeepFM	听这歌的时候，正开着车，下起了暴雨。忽然觉得好带感 (When I heard this song, I was driving in the car and it was raining. Suddenly I felt a good feeling.)
		Retrieval	这首歌和这部动漫就是两个画风 (This song and this anime are two styles.)
		Generation	当时看动漫的时候听到这首歌，感觉好熟悉 (I heard this song when I watched anime, I feel very familiar.)
	萝莉 (Loli)	FM	中午吃饭的时候学校放这首歌，有点小鸡冻 (At the lunchtime, the school played this song. I felt a little exciting)
		DeepFM	我们学校天天吃饭放这歌[汗] (Our school plays this song every day when we are eating)
		Retrieval	真的好好听 听每首歌都会感觉自己是个少女特想恋爱呢 (The song listens really good. I feel like a girl when I listen to every song, and I'd like to fall in love.)
		Generation	少女心爆棚的歌，好甜啊 (The song arouses romance of girls. It is so sweet.)
	学习 (Study)	FM	其实这首歌是在同学聚会上听别人唱的，我喜欢的那个人唱的蛮好听的[亲亲] (In fact, I heard this song by others at the classmate reunion. The person I liked sang pretty good [kiss])
		DeepFM	同学给我讲完难受的英语，开始播放音乐，一听到就不想看书了 (My classmate played this song after teaching me the difficult English exercises. I don't want to read book when I heard this song.)
		Retrieval	好好学学日语吧，要不都听不懂歌曲了 (Study Japanese hard, otherwise you cannot understand the song.)
		Generation	听着这首歌写作业，好幸福 (Listening to this song, writing homework, I feel so happy)

Table 7. Comparing Five Methods by Deploying Them on the XiaoIce Chatbot Online and Collecting Their Click-through Rates

Method	Click Rate	Improves By
(1) Chat Responses	0.444	+12.8%
(2) Mined Reason-like Comments	0.448	+11.8%
(3) Manual Selected Reasons from (2)	0.463	+8.2%
(4) Generated Reasons w/o userTag	0.437	+14.6%
(5) Generated Reasons w/ Scoring	**0.501**	-

Table 7 shows that our proposed method *Generation w/ Scoring* outperforms the other four methods. In particular, it outperforms even the *Manually Selected Reason-like Comments* by 8.2%, which demonstrates the power of *generating* a reason for a given song-user pair. Moreover, it can be observed that the version of our model that learns without user tags performs the worst, which shows the advantages of learning from the (song, user tag, reason) triplets and hence the importance of personalization. Our proposed method improves the mined reason-like comments, which

are associated with a song by real music website users, by 11.8%. This also confirms the power of personalized recommendation reasons in conversations.

5 CONCLUSION AND FUTURE WORK

We formulate a new challenging problem called reason generation for explainable recommendation in conversation applications, where our goal is to increase the click-through rate of the recommended songs by generating recommendation reasons that make users feel as if they are receiving them from their friends. To this end, we build a dataset that consists of (song, user tag, reason) triplets by joining data from a music website and the XiaoIce chatbot and construct a system that learns to generate recommendation reasons for any given song-user pair. Our experiments indicate that our method statistically significantly improves on the DeepFM [8] baseline in terms of overall rating (by 8.9%), relevance to a song (by 3.0%), and personalization (by 16.5%). The average fluency score of generated reasons is as high as 2.93, where 2 means *acceptable* and 3 means *good*. Furthermore, we deploy our proposed methods on the XiaoIce chatbot and observe that the click-through rate of recommended songs improves by at least 8.2% over four different baselines. The improvements indicate that personalized reasons attract more end-users.

In our future work, we plan to incorporate information from song lyrics to increase the relevance of songs to users and to better handle new songs that lack tags. Moreover, we would like to address user tags to generate a fusion reason to deal with the complex and various needs for users.

REFERENCES

[1] Yongfeng Zhang and Xu Chen. 2018. Explainable recommendation: A survey and new perspectives. Retrieved from: *CoRR* abs/1804.11192 (2018).

[2] Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl. 2000. Explaining collaborative filtering recommendations. In *Proceedings of the CSCW*. 241–250.

[3] Konstantina Christakopoulou, Filip Radlinski, and Katja Hofmann. 2016. Towards conversational recommender systems. In *Proceedings of the ACM SIGKDD*. 815–824.

[4] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the EMNLP*. 1724–1734.

[5] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the NIPS*. 3104–3112.

[6] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. Retrieved from: *CoRR* abs/1409.0473 (2014).

[7] Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of the ACL*. 1577–1586.

[8] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A factorization-machine based neural network for CTR prediction. In *Proceedings of the IJCAI*. 1725–1731.

[9] Mao Ye, Dong Shou, Wang-Chien Lee, Peifeng Yin, and Krzysztof Janowicz. 2011. On the semantic annotation of places in location-based social networks. In *Proceedings of the ACM SIGKDD*. 520–528.

[10] Xiangye Xiao, Yu Zheng, Qiong Luo, and Xing Xie. 2014. Inferring social ties between users with human location history. *J. Amb. Intell. Human. Comput.* 5, 1 (2014), 3–19.

[11] Xueming Qian, He Feng, Guoshuai Zhao, and Tao Mei. 2014. Personalized recommendation combining user interest and social circle. *IEEE Trans. Knowl. Data Eng.* 26, 7 (2014), 1763–1777.

[12] Guoshuai Zhao, Xueming Qian, and Xing Xie. 2016. User-service rating prediction by exploring social users' rating behaviors. *IEEE Trans. Multimedia* 18, 3 (2016), 496–506.

[13] Hyoseok Yoon, Yu Zheng, Xing Xie, and Woontack Woo. 2010. Smart itinerary recommendation based on user-generated GPS trajectories. In *Proceedings of the UIC*. 19–34.

[14] Guoshuai Zhao, Xueming Qian, and Chen Kang. 2017. Service rating prediction by exploring social mobile users' geographical locations. *IEEE Trans. Big Data* 3, 1 (2017), 67–78.

[15] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. 1994. GroupLens: An open architecture for collaborative filtering of netnews. In *Proceedings of the CSCW*. 175–186.

[16] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the WWW*. 285–295.
[17] Xiao Lin, Min Zhang, Yongfeng Zhang, Yiqun Liu, and Shaoping Ma. 2017. Learning and transferring social and item visibilities for personalized recommendation. In *Proceedings of the CIKM*. 337–346.
[18] Peiliang Lou, Guoshuai Zhao, Xueming Qian, Huan Wang, and Xingsong Hou. 2016. Schedule a rich sentimental travel via sentimental POI mining and recommendation. In *Proceedings of the IEEE BigMM*. 33–40.
[19] Mukund Deshpande and George Karypis. 2004. Item-based top-*N* recommendation algorithms. *ACM Trans. Inf. Syst.* 22, 1 (2004), 143–177.
[20] Longke Hu, Aixin Sun, and Yong Liu. 2014. Your neighbors affect your ratings: On geographical neighborhood influence to rating prediction. In *Proceedings of the ACM SIGIR*. 345–354.
[21] Xiaojiang Lei, Xueming Qian, and Guoshuai Zhao. 2016. Rating prediction based on social sentiment from textual reviews. *IEEE Trans. Multimedia* 18, 9 (2016), 1910–1921.
[22] Ling-Yin Wei, Yu Zheng, and Wen-Chih Peng. 2012. Constructing popular routes from uncertain trajectories. In *Proceedings of the ACM SIGKDD*. 195–203.
[23] Jitao Sang, Tao Mei, Jian-Tao Sun, Changsheng Xu, and Shipeng Li. 2012. Probabilistic sequential POIs recommendation via check-in data. In *Proceedings of the ACM SIGSPATIAL*. 402–405.
[24] Cheng-Kang Hsieh, Longqi Yang, Honghao Wei, Mor Naaman, and Deborah Estrin. 2016. Immersive recommendation: News and event recommendations using personal digital traces. In *Proceedings of the WWW*. 51–62.
[25] Guoshuai Zhao, Tianlei Liu, Xueming Qian, Tao Hou, Huan Wang, Xingsong Hou, and Zhetao Li. 2017. Location recommendation for enterprises by multi-source urban big data analysis. *IEEE Trans. Serv. Comput.* (2017), 1–1.
[26] Zhiyong Cheng and Jialie Shen. 2016. On effective location-aware music recommendation. *ACM Trans. Inf. Syst.* 34, 2 (2016), 13:1–13:32.
[27] Qiming Diao, Minghui Qiu, Chao-Yuan Wu, Alexander J. Smola, Jing Jiang, and Chong Wang. 2014. Jointly modeling aspects, ratings and sentiments for movie recommendation (JMARS). In *Proceedings of the KDD*. 193–202.
[28] Jin Huang, Wayne Xin Zhao, Hong-Jian Dou, Ji-Rong Wen, and Edward Y. Chang. 2018. Improving sequential recommendation with knowledge-enhanced memory networks. In *Proceedings of the ACM SIGIR*. 505–514.
[29] Wayne Xin Zhao, Sui Li, Yulan He, Edward Y. Chang, Ji-Rong Wen, and Xiaoming Li. 2016. Connecting social media to e-commerce: Cold-start product recommendation using microblogging information. *IEEE Trans. Knowl. Data Eng.* 28, 5 (2016), 1147–1159.
[30] Guoshuai Zhao, Xueming Qian, Xiaojiang Lei, and Tao Mei. 2016. Service quality evaluation by exploring social users' contextual information. *IEEE Trans. Knowl. Data Eng.* 28, 12 (2016), 3382–3394.
[31] Nava Tintarev and Judith Masthoff. 2007. A survey of explanations in recommender systems. In *Proceedings of the ICDE*. 801–810.
[32] Pigi Kouki, James Schaffer, Jay Pujara, John O'Donovan, and Lise Getoor. 2017. User preferences for hybrid explanations. In *Proceedings of the ACM RecSys*. 84–88.
[33] Jesse Vig, Shilad Sen, and John Riedl. 2009. Tagsplanations: Explaining recommendations using tags. In *Proceedings of the IUI*. 47–56.
[34] Li Chen and Feng Wang. 2017. Explaining recommendations based on feature sentiments in product reviews. In *Proceedings of the IUI*. 17–28.
[35] Yunzhi Tan, Min Zhang, Yiqun Liu, and Shaoping Ma. 2016. Rating-boosted latent topics: Understanding users and items with ratings and reviews. In *Proceedings of the IJCAI*. 2640–2646.
[36] Guoshuai Zhao, Xiaojiang Lei, Xueming Qian, and Tao Mei. 2019. Exploring users' internal influence from reviews for social recommendation. *IEEE Trans. Multimedia* 21, 3 (2019), 771–781.
[37] Chong Chen, Min Zhang, Yiqun Liu, and Shaoping Ma. 2018. Neural attentional rating regression with review-level explanations. In *Proceedings of the WWW*. 1583–1592.
[38] Jingyue Gao, Xiting Wang, Yasha Wang, and Xing Xie. 2019. Explainable recommendation through attentive multi-view learning. In *Proceedings of the AAAI*.
[39] Xiting Wang, Yiru Chen, Jie Yang, Le Wu, Zhengtao Wu, and Xing Xie. 2018. A reinforcement learning framework for explainable recommendation. In *Proceedings of the IEEE ICDM*. 587–596.
[40] Fan Yang, Ninghao Liu, Suhang Wang, and Xia Hu. 2018. Towards interpretation of recommender systems with sorted explanation paths. In *Proceedings of the IEEE ICDM*. 667–676.
[41] Guoshuai Zhao, Hao Fu, Ruihua Song, Tetsuya Sakai, Xing Xie, and Xueming Qian. 2018. Why you should listen to this song: Reason generation for explainable recommendation. In *IEEE ICDM Workshops*. 1316–1322.
[42] Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. 2018. Explainable reasoning over knowledge graphs for recommendation. Retrieved from: *CoRR* abs/1811.04540 (2018).
[43] J. Ben Schafer, Joseph A. Konstan, and John Riedl. 1999. Recommender systems in e-commerce. In *Proceedings of the EC*. 158–166.

[44] Beidou Wang, Martin Ester, Jiajun Bu, and Deng Cai. 2014. Who also likes it? Generating the most persuasive social explanations in recommender systems. In *Proceedings of the AAAI*. 173–179.
[45] 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 *Proceedings of the ACM SIGIR*. 83–92.
[46] Wayne Xin Zhao, Yanwei Guo, Yulan He, Han Jiang, Yuexin Wu, and Xiaoming Li. 2014. We know what you want to buy: A demographic-based system for product recommendation on microblogs. In *Proceedings of the ACM SIGKDD*. 1935–1944.
[47] Yao Wu and Martin Ester. 2015. FLAME: A probabilistic model combining aspect based opinion mining and collaborative filtering. In *Proceedings of the WSDM*. 199–208.
[48] Yunfeng Hou, Ning Yang, Yi Wu, and Philip S. Yu. 2019. Explainable recommendation with fusion of aspect information. *World Wide Web* 22, 1 (Apr. 2019), 221–240.
[49] Xu Chen, Yongfeng Zhang, Hongteng Xu, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2018. Visually explainable recommendation. Retrieved from: *CoRR* abs/1801.10288 (2018).
[50] Piji Li, Zihao Wang, Zhaochun Ren, Lidong Bing, and Wai Lam. 2017. Neural rating regression with abstractive tips generation for recommendation. In *Proceedings of the ACM SIGIR*. 345–354.
[51] Felipe Costa, Sixun Ouyang, Peter Dolog, and Aonghus Lawlor. 2017. Automatic generation of natural language explanations. Retrieved from: *CoRR* abs/1707.01561 (2017).
[52] Steve J. Young, Milica Gasic, Blaise Thomson, and Jason D. Williams. 2013. POMDP-based statistical spoken dialog systems: A review. *Proc. IEEE* 101, 5 (2013), 1160–1179.
[53] Margaret Ann Boden. 2006. *Mind as Machine: A History of Cognitive Science*. Clarendon Press.
[54] Sina Jafarpour, Christopher J. C. Burges, and Alan Ritter. 2010. Filter, rank, and transfer the knowledge: Learning to chat. *Advances in Ranking* 10 (2010), 2329–9290.
[55] Hao Wang, Zhengdong Lu, Hang Li, and Enhong Chen. 2013. A dataset for research on short-text conversations. In *Proceedings of the EMNLP*. 935–945.
[56] Yu Wu, Wei Wu, Chen Xing, Ming Zhou, and Zhoujun Li. 2017. Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots. In *Proceedings of the ACL*. 496–505.
[57] Alan Ritter, Colin Cherry, and William B. Dolan. 2011. Data-driven response generation in social media. In *Proceedings of the EMNLP*. 583–593.
[58] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the NIPS*. 3111–3119.
[59] Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Trans. Sig. Proc.* 45, 11 (1997), 2673–2681.
[60] Xiaohui Yan, Jiafeng Guo, Yanyan Lan, and Xueqi Cheng. 2013. A biterm topic model for short texts. In *Proceedings of the WWW*. 1445–1456.
[61] Wanxiang Che, Zhenghua Li, and Ting Liu. 2010. LTP: A Chinese language technology platform. In *Proceedings of the COLING*. 13–16.
[62] Zhenghua Li, Min Zhang, Wanxiang Che, Ting Liu, Wenliang Chen, and Haizhou Li. 2011. Joint models for Chinese POS tagging and dependency parsing. In *Proceedings of the EMNLP*. 1180–1191.
[63] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.* 20, 4 (2002), 422–446.
[64] Steffen Rendle. 2012. Factorization machines with libFM. *ACM Trans. Intell. Syst. Technol.* 3, 3, Article 57 (May 2012), 22 pages.

Received October 2018; revised April 2019; accepted May 2019