Constructing a Transformer-Based Architecture for Explainable Conversational Recommendation

Stanford CS224N Custom Project

Brock Grassy
Department of Computer Science
Stanford University
bgrassy@stanford.edu

Abstract

Recent research on recommendation systems have emphasized producing item recommendations that are both relevant to users and backed by some sort of explanation as to why the item was selected. Parallel research has also been undertaken into producing conversational recommendation systems that allow for users to engage in dialogue with the system and provide iterative feedback on recommended items. This feedback is incorporated into model results, allowing users to request items to be recommended that match more closely with their explicit preferences. Despite there being numerous papers written on these topics individually, investigation into the intersection of these fields have been somewhat limited. In this paper I propose modification of an existing transformer-based explainable recommendation system architecture for use as a conversational explainable recommendation system. To achieve this goal, I also modify an existing explainable recommendation dataset to produce data suitable for use in the conversational objective. I found that this model architecture bettered existing results on explainable recommendation system benchmarks. Although the model did not improve recommendation performance when progressing through a simulated series of sentences in a conversation, a lack of proper conversational data means that we cannot rule out that this architecture would not perform well given better input. Furthermore, results point towards potential benefits of including synthetic conversational data as input to transformer-based explainable recommendation systems, which may improve implementations of the existing state of the art in that area.

1 Key Information to include

- Mentor: Hong Liu

2 Introduction

Recommendation systems are tools that provide recommendations for items (typically either in the form of a top item list or predicted ratings for an item) personalized to users. Traditionally, these recommendations are hard to interpret- these models tend to be black boxes that provide no insight into why certain items are selected. In standard enterprise uses of recommender system (for instance on retail sites such as Amazon), this feature may prove to be useful- users may want to know what reasons are given as to why a certain item is recommended. Research into *explainable recommender systems* aims to tackle this question. There are two main categories of explainable recommendation models. The first method involves incorporating explainability directly into the model through outputting it simultaneously with item ratings. This can be accomplished through construction of a two-headed model that handles each individual task as output from one of its heads. The other strategy commonly used for explainable recommendation is construction of a separate explanation model that can produce explanations independently of the recommendation model architecture. This

provides additional flexibility at the cost of potential loss of information that could be shared between the two tasks.

Explainable recommendation systems can also come in multiple flavors with respect to the type of explanations that they produce. Common examples include:

- Features/concepts the user tends to like that are shared by the item (action movies, folk albums, etc)
- Similar items to the recommended item that the user liked - for example, if the user has liked the movie Knives Out it may provide "user liked Knives Out" as the justification when recommending Clue, a similar murder-mystery styled movie.
- Free textual output ("This movie has great acting and a fast-moving plot", "This album is very pretty")

Constructing each of these explanation types has different trade-offs in terms of effectiveness of explanation and complexity. Computing concept or feature-based explanations may be simpler as there is a more limited set of possible explanations that can arise. Nearest-neighbor styled models are commonly seen in movie streaming websites (for instance in "Because you liked..." sections)- however, if the set of possible items is very large this may be very computationally expensive. Free text is the most flexible option, and is the one I will focus on in this paper.

Another downside to typical recommendation systems is that they lack the capacity for users to simply provide feedback on the recommendations they receive from the model. One potential way to address this issue is through research into *conversational recommendation*. As implied by the name, conversational recommendation systems mirror standard recommendation systems behavior with the additional capacity to receive conversational input from the user interacting with the system. For instance, let's examine an imaginary conversation with such a system.

User: I would like to watch a romance movie.
Model: What about Casablanca?
User: I'd like a more modern movie.
Model: I'd recommend When Harry Met Sally.
User: How about something more bittersweet?
Model: I recommend Eternal Sunshine of the Spotless Mind.

As can be seen, the model provides responses in the form of recommendations to the user's prompts, adjusting the recommendations based on the provided input. For such a system, we may want a conversational model to have the following characteristics:

- Adjusts to user responses and improves recommendations accordingly.
- Provides coherent responses (either by use of template to output sentence or some sort of generating language model).
- Does not sacrifice performance on other metrics, particularly recommendation quality.

There has been a small amount of research on unifying these two objectives into a single model (deemed an *explainable conversational recommendation system*). In this paper, I aim to construct a transformer-based architecture that manages both of these objectives jointly.

3 Related Work

There is a wide array of research relating to recommendation systems as a whole. For one particularly useful high-level review of standard strategies in this system one can refer to Isinkaye et al. (2015).

3.1 Explainable Recommendation: PETER

I examined two main explainable recommendation models, both written by the same set of authors. The first one I examined is PETER (PErsonalized Transformer for Explainable Recommendation)

presented in Li et al. (2021). The aim of this paper is to construct a simple transformer-based explainable recommendation system that provides personalized recommendations for users and items. In order to personalize the recommendation, the researchers propose a methodology for encoding user and item IDs into the model. For the model, we have three different types of token provided as input: user tokens, item tokens, and word tokens. The model constructs three different sets of randomly initialized embeddings for each sets of tokens and concatenates them together, giving the input sequence $[\mathbf{u}, \mathbf{i}, \mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e_n}]$ (where $\mathbf{u}$ is the user embedding, $\mathbf{i}$ is the item embeddings, and $\mathbf{e}_i$ is the embedding for the ith text input token).

To account for these new inputs, they produce a modified attention masking mechanism. The transformer architecture consists of L layers, and for each layer tokens are allowed to attend to all tokens to the left. In addition, the user and item tokens are allowed to attend to each other.

After passing the input data through the transformer, the output is put into a linear model to construct the logits for each word in the output vocabulary. This allows for generation of explanation text through greedy decoding and iterative selection of the word with the highest probability. They define a loss function $\mathcal{L}_e$ to penalize the difference between the predicted probability of the true word and the correct probability. The researchers also construct a mapping between user and item IDs and words and construct a loss term $\mathcal{L}_c$ to encourage the output explanation to be similar to concepts in the user and item embeddings.

Finally, they pass the output through one last multi-layer perceptron to compute a predicted rating for each user-item pair. MSE loss is computed between the predicted and true rating for all user-item pairs in the data, and is denoted by $\mathcal{L}_r$. The overall loss combines all these objectives into a single function as follows:

$$\mathcal{J} = \min_{\Theta}(\lambda_e \mathcal{L}_e + \lambda_c \mathcal{L}_e + \lambda_r \mathcal{L}_r).$$

The λ terms handle regularization and the relative weighting of each of the three objectives in the overall loss function, allowing people training the model to determine if they value explanation quality or recommendation quality more. This model provides good initial results- however, the transformer architecture is rather shallow. The next paper attempts to address what happens with a larger model.

3.2 Explainable Recommendation: PEPLER

The same authors produced a later model, PEPLER (PErsonalized Prompt Learning for Explainable Recommendation), that uses a pretrained GPT-2 transformer as the basis of their architecture (Li et al., 2023). The rough intention of the model is similar- they still want to produce a recommendation system that generates natural language explanations while retaining strong performance on current recommendation performance metrics. They propose a variety of variations of their model, but I would like to focus specifically on their "continuous prompting" model.

For this paper, the researchers use a similar strategy to construct user and item embeddings. These embeddings are initially randomly constructed in the same fashion as they were for PETER, and we construct a similar input sequence $[\mathbf{u}, \mathbf{i}, \mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e_n}]$. This input is fed through the GPT-2 model to generate output data and a corresponding explanation. Transformer loss is measured by the loss $\mathcal{L}_C$, defined in the same way as the previous model.

The researchers expand the recommendation rating generation model head to provide two options: the data is either fed through a matrix factorization (MF) method or a MLP (multi-layer perceptron). Either way, this recommendation model outputs a rating prediction for each user-item pair, with its loss penalized by MSE as a term $\mathcal{L}_R$. The researchers found that the MF head produces better text quality relative to MLP, while the MLP head produces better rating performance. This architecture is a flexible way to handle explainable recommendation that lends itself well to model tweaks, such as the one to include conversational input that I discuss later in the paper.

3.3 Explainable Conversational Recommendation: ECR

I found one main existing explainable conversational recommendation system (Chen et al., 2020). As discussed in the introduction, explainable conversational recommendation aims to marry the objective

of explainability with the capacity to have conversations with the user about the recommendations. This paper aimed to satisfy three main objectives: recommendation accuracy, explainability, and explanations that reflect the concepts provided in the user's messaging.

To accomplish these goals the authors leverage Microsoft Concept Graph, a system that defined a series of concept-level embeddings and an associated graph that shows connections between these concepts. As a first pre-processing step, they extract a set of concepts for each user and item. Without loss of generality, we can discuss how concepts are extracted for users. The authors find the set of all reviews that are submitted for each individual user. Using Microsoft Concept Graph, they find the set of concepts that appear the most frequently across these reviews. This same process is repeated for items.

After extracting user and item concepts, they proceed with extracting concepts from the conversational feedback provided by the user. These concepts are all fed into the model, which produces both explanation text and a predicted rating (as with the other models). In generating explanations, they use a GRU to constrain generation to use the most important concept provided in the input feedback. The loss is also penalized if the explanation is dissimilar to the other most relevant concepts with positive sentiment, or if it is similar to the most relevant concepts with negative feedback sentiment. This encourages the model to produce explanations that are similar to what the user wants.

When attempting to use this model as a baseline, I found that Microsoft has deprecated Microsoft Concept Graph at some point in the recent past. There were no associated codebases with this paper either. I invested a significant amount of time into attempting to implement the paper myself, but was unable to find a suitable reimplementation for Microsoft Concept Graph. As such, I am left without a suitable baseline directly for explainable conversational recommendation models and require additional work to construct baselines.

4 Approach

In order to construct my own explainable conversational recommendation system, I decided to modify the architecture provided in PEPLER to allow for conversational input- I use the codebase cited in their paper as a starting point for my implementation. Figure 1 shows a flow chart of the new model's architecture.

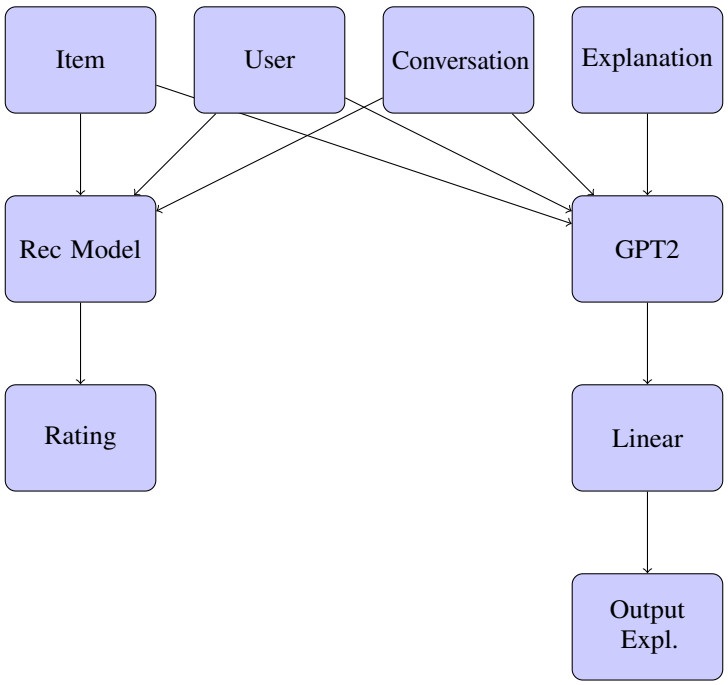

Figure 1: Proposed Transformer-Based Explainable Conversational Recommendation Architecture

As with PEPLER, my model constructs user and item embeddings $\mathbf{u}_a \in \mathbb{R}^d$, $\mathbf{i}_b \in \mathbb{R}^d$. In the original model, for a ground truth explanation sentence of padded length n, we concatenate this data to construct an input of dimension $\mathbb{R}^{d\times(n+2)}$ and feed it into GPT-2 to produce our predicted explanation. To add conversational input, I used the same tokenizer used to tokenize ground-truth explanation sentences to tokenize the input text. From this, the model's embeddings are computed for all tokens. In order to standardize the size of this embedding, I took the mean of all token embeddings in the input to produce a final input vector $\mathbf{c}_j \in \mathbb{R}^d$. The aim of this transformation is to incorporate all concepts in the sentence into a single unified vector.

Once we have all of our individual embeddings, we must now define how the explanation and recommendation heads of the model handle the new input. For the recommendation task, we simply concatenate the conversational embedding with the other embeddings like so: $[\mathbf{u}, \mathbf{i}, \mathbf{c}, \mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e_n}]$. Our new input is in $\mathbb{R}^{d\times(n+3)}$, and can be directly passed into the same GPT-2 model as is used in PEPLER.

To pass the conversational embedding into the recommendation model, we can instead construct an input by directly concatenating the user, item, and conversational embedding vectors together as a flat vector like so. The ensuing vector $[\mathbf{u}^T\mathbf{i}^T\mathbf{c}^T]$ is in $\mathbf{R}^{3d}$. I then proceeded to modify the MLP and MF methods used as recommendation models to take this additional input. This modification strategy is straightforward for the multi-layer perceptron, as simply upping the input dimension of the model to by $3d$ allows for the model to directly take all of that information as input. For the matrix factorization model, the output rating was originally determined by returning $\mathbf{u}^T\mathbf{i}$. I tweaked this by simply having it output $(\mathbf{u} \circ \mathbf{i})^T\mathbf{c}$ (where $\circ$ denotes the Hadamard product of two vectors).

No tweaks were made to the loss function or other internal parameters of the model. Ideally there would be an additional term in the loss function that, similarly to the ECR model presented before, directly rewards the model for outputting explanations and recommendations conceptually similar to positive concepts expressed in the conversational input and penalizes it for explanations and recommendations similar to negative or irrelevant concepts. I tried a few strategies of doing so (measuring similarity scores of word embeddings, etc) but was not able to come up with a satisfactory loss function to incorporate this. Despite this setback, it's still worth exploring if the conversational input leads to improvement without an explicit loss term.

5 Experiments

5.1 Data

I could not find any datasets that are custom-built for explainable conversational recommendation. As such, I modified an existing Amazon-based explainable recommendation dataset (provided in the PETER/PEPLER papers) to generate synthetic conversational data. The original form of the dataset is in the form of 450k (user, item, rating, explanation, key feature) movie review tuples. The explanation sentence is simply taken to be the first sentence of the full Amazon review associated with the user-item pair. The key feature is a single word that was determined to be most important that the researchers extracted from the explanation. I modified the dataset through the following process:

1. Use pretrained GloVe embeddings[1] to find 20 most similar words to the key feature.
2. Manually construct positive, negative, and mixed sentiment sentence templates that users may use in conversation (such as "I would like to watch movies related to <WORD>.", "I do not like movies about <WORD_1> but I enjoy <WORD_2>".)
3. Randomly choose positive and negative templates. For positive templates, fill with a random word from the most similar words. For negative templates, fill with a random word not in the most similar words.
4. Return the new (user, item, rating, explanation, sentences) tuple.

The result is now that for each explanation we put as input we include a series of sentences that the user could have conceivably provided in order to get an explanation tailored to these features. As one

[1] https://nlp.stanford.edu/projects/glove/

example, suppose the explanation in our original training data is "This movie is a good action movie with an enjoyable ending". Suppose that "action" is isolated as the top word, and that "action" and "energy" are two similar concepts chosen while "romance" and "slow" are two dissimilar concepts chosen. Our template-based system would produce input like the following:

"I like movies related to action. I do not like movies with romance. I am interested in movies with energy, but not movies that are slow."

This naturally is an imperfect solution- we have no guarantee that our chosen concepts through vector similarity are similar to the key feature in context, nor that the templates we construct are grammatically correct. If a dataset existed with both true conversational data collected from real people and corresponding valid item explanations, then we would be able to get around these data integrity issues. However, across the whole corpus we should tend to include concepts as input that are similar to what the user has given in their review, so it should somewhat simulate the sort of data we want to account for.

5.2 Evaluation method

We have four main categories of metrics that we want to measure: recommendation quality, output text quality (syntactically), output text explainability, and receptiveness to conversational input. For recommendation quality, we use RMSE and MAE to measure the error between the predicted rating and true rating across data in our training set. To measure the quality of our output text, we can use BLEU and ROUGE. FMR (a metric defined in the PETER/PEPLER papers) calculates explanability performance on our dataset by finding the proportion of data in our test set that have the key feature in their generated explanation. Finally, to measure conversational ability, I generated 3 conversational sentences for each test data item using the aforementioned methodology. For each set of 3 I then tested model RMSE performance given the first sentence, the first two, and all three sentences as input sequentially. If the model is properly incorporating the conversational input we would ideally see performance improve as more relevant sentences are provided.

5.3 Experimental details

As discussed in the model description, there is a regularization term that determins how much we weight the rating objective as opposed to the explainability objective. I tested values for that in the set $\lambda = \{0.01, 0.1, 1, 10\}$. As both PEPLER and my new model architecture take this parameter, I trained both models across these parameters. I also used both the matrix factorization and MLP rating model heads for both architectures, and added a PETER model with default parameters as one more baseline. All models were trained with initial learning rate 0.001 and Adam optimization for 100 epochs, with early stopping after 5 epochs without improvement.

5.4 Results

I have bolded the top three models for each metric across all trained models in Table 1. As can be seen, my model has at least one version of it appearing in the top three for all metrics except FMR. Similarly to the original PEPLER model architecture, the MLP version of the model performs better on the rating metrics while the MF version performs relatively better on explanation text quality. These results imply that including the synthetic conversational data into the model input improves rating performance compared to the model without conversational input. Even if the model doesn't perform well in the conversational metrics, this shows that incorporating synthetic conversational data into transformer-based method may directly improve explainable recommender system performance.

When testing the data on the aforementioned conversational metric, results across models all had the same behavior as in Table 2- there appeared to be no benefit to providing more than one conversational sentence with regards to RMSE. This could be due to poor quality in the synthetic data- before writing off this model as a bad option in this domain it would make sense to evaluate it using a better conversational dataset.

Model	RMSE	MAE	BLEU-4	ROUGE	FMR
My Model ($\lambda_R = 0.01$, MF)	0.9994	0.7686	**1.0796**	1.8542	0.1100
My Model ($\lambda_R = 0.1$, MF)	0.9745	0.7389	**1.0763**	**1.8932**	0.1111
My Model ($\lambda_R = 1$, MF)	1.0209	0.7922	0.9148	1.7875	0.1166
My Model ($\lambda_R = 10$, MF)	1.0562	0.8385	0.8330	1.4990	0.1025
My Model ($\lambda_R = 0.01$, MLP)	**0.9472**	0.7228	0.9373	1.7941	0.1126
My Model ($\lambda_R = 0.1$, MLP)	**0.9490**	**0.7083**	0.8554	1.6723	0.1031
My Model ($\lambda_R = 1$, MLP)	0.9536	0.7190	0.7172	1.4979	0.0977
My Model ($\lambda_R = 10$, MLP)	0.9511	0.7146	0.7596	1.4994	0.1026
PEPLER ($\lambda_R = 0.01$, MF)	1.1707	0.9341	0.9869	**1.9080**	**0.1273**
PEPLER ($\lambda_R = 0.1$, MF)	1.1436	0.9276	0.7912	1.7742	0.1195
PEPLER ($\lambda_R = 1$, MF)	1.1587	0.9413	0.9334	1.7115	**0.1248**
PEPLER ($\lambda_R = 10$, MF)	1.1549	0.9358	0.5179	1.5041	0.0928
PEPLER ($\lambda_R = 0.01$, MLP)	0.9513	0.7298	**1.0548**	**1.9187**	0.1073
PEPLER ($\lambda_R = 0.1$, MLP)	0.9525	**0.7016**	0.8242	1.7418	0.1159
PEPLER ($\lambda_R = 1$, MLP)	**0.9512**	0.7216	0.9231	1.6727	0.1136
PEPLER ($\lambda_R = 10$, MLP)	0.9549	**0.7092**	0.9428	1.8564	**0.1289**
PETER	0.9612	0.7268	0.9311	1.7102	0.0925

Table 1: New model and baseline model results

	1 Input	2 Inputs	3 Inputs
RMSE	0.7478	0.7456	0.7465

Table 2: Sample conversational results (MF, $\lambda_R = 0.1$)

6 Analysis

More care is needed when examining qualitative performance for these models. As there are rather dramatic differences in the architecture of my model and PEPLER with the PETER model, I will restrict this analysis to observation of specific text outputs. Let's examine the following true explanation and the generated explanations across models (restricting PEPLER and my model to use $\lambda_R = 0.1$ and matrix factorization):

Ground Truth: you realize just how different the characters and situations are from most thrillers My Model: the film is a thriller in the vein of the best of the genre PEPLER: the movie is a little too long and too long PETER: the characters are good

As can be seen, it seems like my model has captured features more specific to the recommended item itself, while both of the other models produce more generic explanations. However, this is not universally true. In investigation into other results, I found that some phrases consistently appeared in my model's generated text. For instance, every ten generated explanations or so had the phrase "this is a great movie" as the whole explanation or a subset of it. This points to a need for improvement in the objective function for our model- if we explicitly encourage that explanations with concepts similar to user feedback are returned, with either good feedback provided or strong synthetic data we should be able to construct more meaningful explanations than this.

In an initial investigation into the set of poor explanations, I have isolated a few typical other cases that tend to arise:

1. Poor quality initial explanations. If the initial explanation quality is protracted and gives no relevant information, the ensuring predicted explanation mirrors it accordingly. This could be mitigated with higher quality explanation data that is not inferred directly from item reviews.
2. Synthetic conversational data is not useful. If the randomly selected concepts are relevant, the ensuing explanations tend to be less relevant as well.
3. The generated explanation is occasionally opposite in sentiment from the true one. For example, in one case the true explanation is "i also did not like some very long shots and

kubrick 's choice of cruise and kidman" and the generated is "i am a big fan of tom cruise and i think he is a great actor". This is likely due to opaqueness in the key feature selection and similar word randomization when constructing synthetic sentences. There is no concept of sentiment involved in either of these processes. As such, a more robust mechanism of text construction may proper encapsulate these nuances and result in better downstream results.

7 Conclusion

Existing research on explainable conversational recommendation is very limited, and published architectures for this problem that involve transformers as their base are even scarcer. In this project I have examined and proposed a new transformer-based architecture that matches or bests state of the art performance of explainable recommendation systems on a dataset of Amazon reviews. Although the model does not improve in testing as more synthetic sentences are added as conversational input, the inclusion of these sentences contributes directly to improving performance over current non-conversational architectures. As these sentences can be generated and incorporated into transformer-based architectures without making the model fully conversational, the strategy I have described here has the potential to enhance the performance of existing state of the art explainable recommendation systems. Throughout my work on this project I have gained valuable experience working with transformer-based architectures and a wealth of domain-specific knowledge relating to recommendation systems. The following limitations of and potential associated avenues for future work may potentially provide ideas for myself or others to expand on the conclusions drawn from this paper:

1. There exists no current conversational explainable recommendation dataset. As such, it may be worth investing more time into either generating higher-quality synthetic data (through factoring in sentiment into generation, improved feature similarity metrics, less rigid synthetic templates, etc) or collecting data suitable for this task.
2. Similarly, it appears that current explainable datasets are somewhat limited by the quality of the explanations in the data. The Amazon model does not come with proper explanations for the recommendations in the training data, and instead infers it from the text in the associated review (which may not provide any sort of explanation at all). Running this model on a better dataset or acquiring better explanation data may also improve performance.
3. The model could benefit from explicit incorporation of similarity between the input feedback and output explanation into the loss metric to improve explanation quality. Due to time and training constraints, I was unable to find a suitable method to do so.
4. The embedding aggregation for the conversational input (taking the mean of embeddings across tokens in the input) may lose information and prove meaningless if there is a lot of input provided. As such, it may be worth exploring additional ways to pass the text input into the model. I briefly tried prepending this text to the ground-truth explanation passed into training, but found results to be unsatisfactory. Future efforts could explore either more robust embedding generation for multiple sentences or further tweaks to the architecture to better include full conversational history.
5. Current analysis of textual quality is not very formalized. It may be worth seeking out external human assistance to evaluate this model on output quality due to the limitations of purely computer-based evaluation systems.

8 Acknowledgements

I would like to thank my mentor, Hong Liu, for the comments provided on my proposal and milestone, as well as for his responses to the questions I had related to my project. I would also like to acknowledge AWS for stepping in to help with the class GPU situation and ensuring that myself and other students had access to the proper resources to make our projects possible. Finally, I would like to thank Professor Manning and the whole CS224 staff for the engaging content they have provided throughout the course of this class- it has been incredibly enjoyable throughout, and I'm very I had the chance to participate in this experience over the last few months.

References

Zhongxia Chen, Xiting Wang, Xing Xie, Mehul Parsana, Akshay Soni, Xiang Ao, and Enhong Chen. 2020. Towards explainable conversational recommendation. In *International Joint Conference on Artificial Intelligence*.

F.O. Isinkaye, Y.O Folajimi, and B.A. Ojokoh. 2015. Recommendation systems: Principles, methods and evaluation. In *Egyptian Informatics Journal*, Online.

Lei Li, Yongfeng Zhang, and Chen Li. 2021. Personalized transformer for explainable recommendation. In *Annual Meeting of the Association for Computational Linguistics*.

Lei Li, Yongfeng Zhang, and Chen Li. 2023. Personalized prompt learning for explainable recommendation.