Conversational Recommendation: Listen Before You Talk

ANONYMOUS AUTHOR(S)

Conversational recommender systems enable richer user-preference elicitation through dialogues. At its core, developing a conversational recommender system requires understanding the preferences of a user expressed in natural language and turning them into valuable item recommendations. In this paper, we focus on making recommendations (assigning scores to items) from dialogues. We study different ways to adapt recent language models to improve recommendation quality. We contribute four new model variations that combine transformers and that are inspired by classic recommender models such as matrix factorization. Using a conversational recommendation dataset we discuss advantages and drawbacks for each of them, tackling aspects like user cold-start, item cold-start, the portion of ranking that matters, time and computational complexity. We also observe that in this context hyperparameters can be tuned to favor certain metrics over others.

Additional Key Words and Phrases: Recommendation, Dialogue Systems, Transformers

ACM Reference Format:
Anonymous Author(s). 2020. Conversational Recommendation: Listen Before You Talk. In *RecSys '20: ACM Conference on Recommender Systems, September 22–26, 2020, Rio de Janeiro, Brazil.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Conversational recommendation encapsulates several machine learning sub-tasks including preference elicitation, item recommendation, and language generation. In this work we focus on item recommendation as the core task for a conversational recommender system and we study methods for the task of item (in our case, movie) recommendation from conversation.

This task comes with challenges, for example, a user may express preferences about items and their attributes (I enjoy action movies such as Wonder Woman). Further, the dialogue component introduces challenges beyond standard settings like recommendations from user reviews [7], since models have to understand dialogue dynamics. Fully interactive dialogue systems allow models to also alter dialogue dynamics; however, here we focus on the retrospective analysis of recorded dialogues to evaluate model performance in a controlled manner.

To explore this setting, we use, adapt and extend state of the art transformer based neural language modeling techniques to the task of recommendation from dialogue. We study the performance of different methods using the ReDial dataset [5], a conversational-recommendation dataset for movies. We also make use of a knowledge base of movies and measure their ability to improve performance for cold-start users, items, and/or both. We find that *pre-trained transformer models outperform baselines even if the baselines have access to the user preferences manually extracted from their utterances.*

2 TRANSFORMERS FOR LANGUAGE MODELLING

We provide a succinct review of transformer models focusing on the BERT model [2] which we use in this paper. The architecture of BERT is a multi-layer transformer encoder based on the original implementation described in Vaswani et al. [10]. It is pre-trained on two unsupervised tasks to obtain encoded representation of language. The first task is masked language modeling (MLM), where the goal is to predict tokens that were masked from the input sequence. The second task is next sentence prediction (NSP) where two sentences are input and the goal is to predict if the second sentence follows the first one. Pre-trained BERTs can than be used for downstream tasks, either in a feature-based manner or by fine-tuning, fine-tuning showing better results.

In this work we use two outputs from BERT: the [CLS] token and the average of last hidden layer (*avg*). There is a fully connected layer using as input the first token ([CLS] token) of BERT's last hidden layer. Its output is the same dimension as the input. The weights of this layer are trained during the NSP pre-training phase. We refer to this output as the pooler output (*pool*).

3 RECOMMENDATIONS FROM DIALOGUES

Problem definition. Our task is item recommendations for text-based conversational recommendation systems. A conversation consists of a sequence of utterances (a user utterance followed by a system utterance) indexed $\{1, \ldots, T\}$. The system can recommend an item i after each user u's utterance. The task then consists of predicting user preference or a rating R_u^{t+1} at time $t+1$ given a conversation $c_u^{1:t}$ up to utterance t. In a complete system this predicted rating would then be used downstream to compose the next utterance.

Data. We assume that we have access to the text of each conversation which we denote $c_u^{1:T}$. Each utterance is prefixed by a special token to denote the corresponding speaker's role ("S::" for user seeking the recommendation and "R::" for the recommender). We also assume access to a user's preferences or ratings vector $\mathbf{R}_u^{1:t} \in \mathbb{R}^{|\mathcal{V}|}$ extracted from the dialogues up to utterance t (for example, labelled by the users at the end of the dialog). In practice the data we study has binary labels (liked/disliked), but our models generalize beyond that.

In addition, to model cold-start items, some models will use item features. We assume access to a knowledge base of movies from which we can extract these features and encode them as text. We denote the resulting string as i_f. Further, in conversations, users can also express their preferences toward item attributes. Such preferences can be extracted. For example, in our target dataset users often mention movie genres they like. We encode the attributes a user mentions in a sparse vector $\mathbf{A}_u \in \mathbb{R}^{|\mathcal{V}|}$ (or $\mathbf{A}_u^{1:t}$ to denote available attributes following utterances 1 through t) where non-zero entries correspond to movies that have all attributes mentioned by user u. If $\mathcal{A}_u$ represents the set of attributes mentioned by user u and $\mathcal{A}_i$ the set of attributes associated with movie i in the knowledge base, we have that, for i $\in \mathcal{V}$, $A_{ui} = \mathbf{1}_{\mathcal{A}_u \subseteq \mathcal{A}_i}$, where $\mathbf{1}$ is the indicator function.

4 MODELING

We use and develop several models for the task of recommendations from dialogue utterances. We first discuss models which assume that user preferences (ratings) have been pre-extracted from the conversation and are available in a ratings-vector format ($\mathbf{R}_u^{1:t}$). We then introduce several variations of transformer models based on BERT that directly model the dialogue text. We also consider methods for modelling movie features to address the cold-start problem.

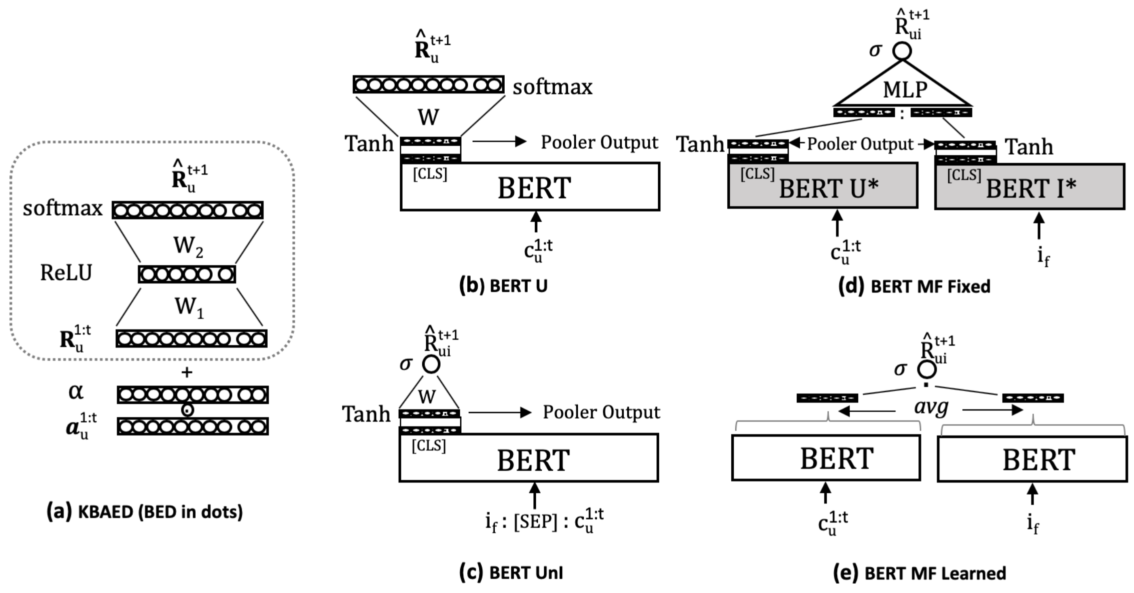

Fig. 1. We study different classes of models for item recommendation from conversation ($c_u^{1:t}$). (a) Encoder-decoders that rely on ratings $\mathbf{R}_u^{1:t}$ and attributes $\mathbf{a}_u^{1:t}$ having been extracted from the conversation. (b–e) Transformer-based models that directly model the (text) conversation. Models (b) and (c) are instances of BERT trained for recommendations. Models (d) and (e) are inspired by matrix factorization approaches and use two BERTs to learn a user and an item embedding. For (d) we reuse the BERTs trained using (b) (and denote them BERT U* and I*).

4.1 Baselines

Our *Basic Encoder Decoder BED (Figure 1a top)* is an instance of an auto-encoder for recommendations [9]. It use the vector of observed ratings $\mathbf{R}_u^{1:t}$ as input. Ratings must then be extracted from the dialogue utterances upstream. BED serves as a baseline method with which to compare models of conversation (§4.2). Ratings $\mathbf{R}_u^{t+1}$ are computed as $\text{softmax}(\mathbf{W}_2 \cdot \text{ReLU}(\mathbf{W}_1\mathbf{R}_u^{1:t} + \mathbf{b}_1) + \mathbf{b}_2)$ with parameters $\theta := \mathbf{W}_{\{1,2\}}, \mathbf{b}_{\{1,2\}}$. The parameters are optimized by minimizing a masked binary cross-entropy (MBCE) on the training data. It is masked since only the output dimensions that are observed are used in the loss. (It is not standard to combine a softmax with a BCE loss, but we found that it performs best in our study and is also supported by other studies [6].)

We also examine a *Knowledge-base augmented BED model (KBAED) (Figure 1a)*. In conversational recommender systems, users can express preferences toward attributes of movies. For example, the dataset we study in this paper contains many conversations in which movie genres (e.g., comedy, drama) are used to steer recommendations. KBAED extends BED to model such attribute preferences. The assumption we make is that a user expressing a preference for one or more attributes is expressing a positive preference toward movies that contains these attributes.

The input of KBAED consists of a vector of available attributes $\mathbf{A}_u^{1:t}$—which encodes the preferences of user u for item attributes—summed with the ratings vector $\mathbf{R}_u^{1:t}$. For genres, we found that limiting the attribute preferences to the top movies (we use 100) and weighting the level of the preference using movie popularity was beneficial: $\mathbf{a}_u^{1:t} = \| \text{top}_n(\mathbf{A}_u^{1:t} * \mathbf{p}) \|$, where top_n returns its arguments for the top n values and zero otherwise. We estimate a movie's popularity using its number of (training) ratings $\mathbf{p} = \sum_{u \in \mathcal{U}} [\mathbf{R}_{u,1}, \dots, \mathbf{R}_{u,|\mathcal{V}|}]$. Hence, $\mathbf{R}_u^{t+1} = \text{softmax}(\mathbf{W}_2 \cdot \text{ReLU}(\mathbf{W}_1(\mathbf{R}_u^{1:t} + \boldsymbol{\alpha}\mathbf{a}_u^{1:t}) + \mathbf{b}_1) + \mathbf{b}_2)$ with parameters $\theta := \{\mathbf{W}_{\{1,2\}}, \mathbf{b}_{\{1,2\}}, \boldsymbol{\alpha}\}$. KBAED trains with the MBCE loss.

4.2 Language models for recommendations from dialogue

Here we introduce a set of recommendation models based on BERT. Compared to the above baselines these models have the ability to make recommendations directly from the text of the conversations $c_u^{1:t}$. We explore different methods for transforming representations inferred through BERT into recommendations. We present two methods inspired by matrix-factorization (MF) modelling [8] that learn a separate user and item representation which is then combined into a rating (prediction). We also explore methods for modelling the item attributes i_f in the conversation (in addition to the conversation).

Bert U (Figure 1b): is a vanilla BERT which models the text in the conversation. Its pooler output (*pool*) is used as input to an affine transformation followed by a softmax over all movies $\mathbf{R}_u^{t+1} = \text{softmax}(\mathbf{W}(pool(\text{BERT}(c_u^{1:t})) + \mathbf{b})$ with parameters $\theta := \mathbf{W}, \mathbf{b}$ and BERT parameters. Training is done with MBCE. In a similar fashion, we can also instanciate BERT I which takes as input i_f and predicts ratings for all users.

BERT UnI (Figure 1c): One limitation of BERT U is that it cannot recommend new (cold-start) items. As a remedy we can use BERT to also model item attributes. The input of BERT UnI consists of the conversation $c_u^{1:t}$ concatenated with the item features i_f separated by a special token ([SEP]). An affine transformation is added on top of BERT's pooler output to predict the particular user-item rating $R_{ui}^{t+1} = \sigma(\mathbf{w} \cdot pool(\text{BERT}(c_u^{1:t} : [SEP] : i_f)) + b)$ with parameters $\theta := \mathbf{W}, \mathbf{b}$ and BERT parameters all learned with BCE loss.

BERT MF Fixed (Figure 1d): is inspired by the neural collaborative filtering model [4]. We concatenate the BERT representations of a user and an item and input them to a neural network which predicts their rating. Each representation is the pooler output of optimized BERT U and BERT I models. The predicted rating R_{ui}^{t+1} is given by $\sigma(\text{MLP}([pool(\text{BERT U}^*), pool(\text{BERT I}^*)]))$ where learning optimizes the MLP parameters only, using BCE loss.

BERT MF Learned (Figure 1e): is similar to BERT MF Fixed, but 1) it combines the user and item representations through a dot product, and 2) both BERTs are trained using the BCE loss. Preliminary results showed benefits in using the average of the last hidden layer (*avg* instead of *pool*) to obtain representations: $R_{ui}^{t+1} = \sigma(\text{avg}(\text{BERT}(c_u^{1:t})) \cdot avg(\text{BERT'}(i_f)))$.

5 STUDY

Dataset. We use the ReDial dataset which contains 11,348 conversations between a user seeking a recommendation and a user acting as a recommendation provider [5]. Binary ratings have been extracted at the utterance level. We split the conversations 80/10/10 in train/valid/test. We further segment conversations into multiple instances, one for each user utterance that was followed by a recommendation from the recommender. There are 6,924 movies (items) discussed in ReDial. Our test set has 299 target movies that were never seen in the training set. These will be considered as new movies for studying item cold-start and we use a movie knowledge base from IMDB[1] to create movie item embeddings.

Experimental Details: When considering the attributes $\mathbf{A}_u^{1:t}$ mentioned by a user we limit ourselves to the genre attribute, as it is the most common one in ReDial's conversations. We matched them using a list of 24 popular genres from IMDB. [2] We studied three different subsets of item information: i_f corresponds to a text representation of the full knowledge base we collected about item i, i_m is a shorter version with only title, genres, and actors, and i_s, is the same

[1]we accessed the data using the IMDb python package https://pypi.org/project/IMDb/
[2]"Popular TV Series by Genre" from https://www.imdb.com/feature/genre/?ref_=nv_ch_gr

Table 1. Model description and recommendation test results on ReDial. $\mathbf{W}, \mathbf{b}, \boldsymbol{\alpha}$ are trainable parameters as those in BERT models, f is the ReLU activation function, σ is sigmoid, $\cdot$ is dot product, : concatenation and * stands for optimized models. On the results side, numbers in parentheses imply that only targets that were not unseen in train set are used (equivalent to new items). Results for ReDial Model and KBRD copied from Chen et al. [1].

Model	Model Description	Recall@1	Recall@10	Recall@50	NDCG@100
ReDial model[5]	–	2.3	12.9	28.7	-
KBRD[1]	–	3.0	16.3	33.8	-
BED	$\text{softmax}(\mathbf{W}_2 f(\mathbf{W}_1 \mathbf{R}_u^{1:t} + \mathbf{b}_1) + \mathbf{b}_2)$	2.96	13.26	30.02	0.1262
KBAED	$\text{softmax}(\mathbf{W}_2 f(\mathbf{W}_1 (\mathbf{R}_u^{1:t} + \boldsymbol{\alpha} \mathbf{a}_u^{1:t}) + \mathbf{b}_1) + \mathbf{b}_2)$	3.05	15.33	34.16	0.1402
BERT U	$\text{softmax}(\mathbf{W}(pool(\text{BERT}(c_u^{1:t})) + \mathbf{b})$	**3.13** (0)	15.36 (0)	34.25 (0)	0.1365 (0)
BERT MF Fixed	$\sigma(\text{MLP}(\text{BERT U}^* : \text{BERT I}^*))$	2.93 (0)	13.33 (0)	32.54 (1.06)	0.1300 (0.0078)
BERT MF Learned	$\sigma(avg(\text{BERT}(c_u^{1:t}))) \cdot avg(\text{BERT'}(i_f))$	3.08 (0)	15.16 (0)	36.10 (0.71)	0.1432 (0.0048)
BERT UnI ($R^{-}20$)	$\sigma(\mathbf{w}(pool(\text{BERT}(i_s : \text{[SEP]} : c_u^{1:t}))) + b$	2.03 (0.24)	15.28 (4.33)	**40.59** (9.38)	0.1471 (0.0320)
BERT UnI ($R^{-}40$)	$\sigma(\mathbf{w}(pool(\text{BERT}(i_s : \text{[SEP]} : c_u^{1:t}))) + b$	3.02 (0.24)	**16.54** (4.33)	38.67 (6.73)	**0.1511** (0.0231)

as i_m but with only up to three (leading) actors. In Table 1 we report the exact attributes that were selected (based on cross-validation using ndcg@100 for early stopping) in each model.

We need to manage the rating imbalance in ReDial: 94% of ratings are 1s (liked) and only 6% are 0s. For models predicting a vector of ratings (BED, KBAED, and BERT U), it helps to use a softmax output forcing the model to keep a fixed probability mass over all items [6]. For the others, we used the common practice of imputing zero ratings to a random subset of unobserved items [3, 12]. We denote the number of random ratings as R^{-}.

Metrics. We use two standard metrics in our experiments: *Recall@k* and *NDCG@k*. Both evaluate the quality of the top-k ranked items by a model. Recall considers that items ranked below rank k have uniform importance, while NDCG exponentially reduces the importance of lower-ranked items.

5.1 Results

We first report overall results and then focus on different facets of the tasks and the models. Table 1 reports the average test recommendation performance of all models. In addition to the model we propose, we also compare with KBRD [1]. KBRD extracts entities from the dialogue and learn their embeddings using an external knowledge graph. Predictions are obtained through attention on these learned entities. The results of the ReDial model (the model accompanying the ReDial paper[5]) and KBRD were taken directly from Chen et al. [1]. To our knowledge, our setup is the same as theirs and the results comparable (up to a random seed).

The best recommendations are obtained by BERT UnI (only outperformed by BERT U in Recall@1). The fact that it infers a joint user-item distribution (as opposed to MF models that infer them independently) may explain this result. Note that we used i_s for BERT UnI for computational reasons since its input sequences are longer (concatenation of user and item text). Still, BERT UnI comes with a larger computational cost at prediction time since it needs to do a BERT forward pass for all combinations of users and items (Table 3). BERT MF provides good performance at reasonable cost since inferred user and item embeddings can be reused across predictions.

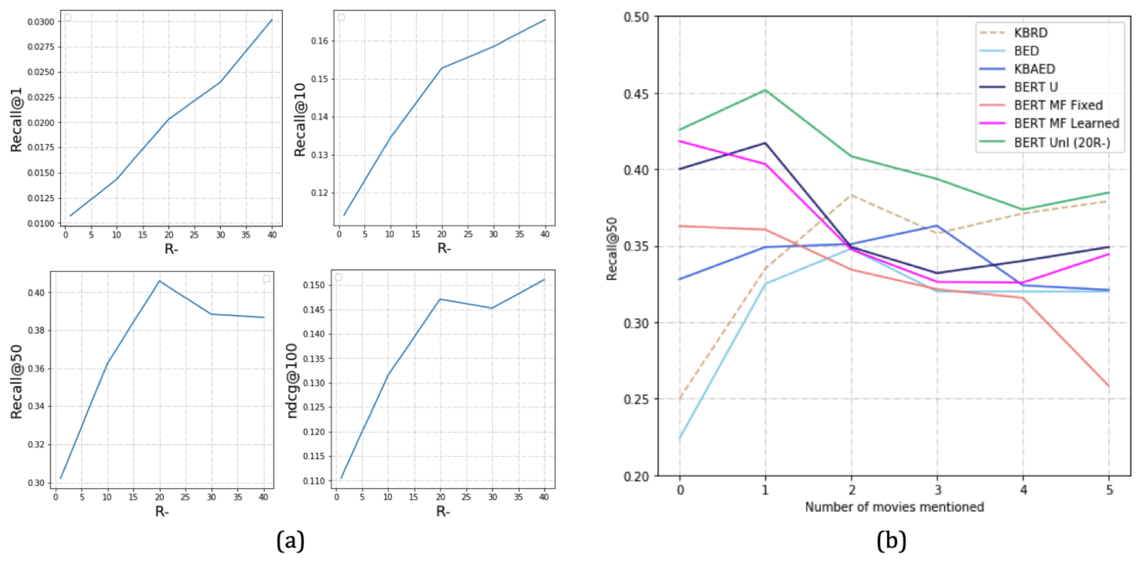

Fig. 2. (a) Impact in variation of R^{-} on four metrics. (b) Recall@50 for all models, KBRD[1] is previous state of the art.

5.1.1 Sequential and User Cold Start evaluation. In conversational recommendations the system gathers information throughout the dialogue. Figure 2b presents a temporal view of the experiment reported in Table 1. We group conversations in terms of how many movie preferences were available before making the recommendation. We note that overall the relative performance of models are similar in Table 1 with Bert UnI outperforming all approaches.

The trend that performance degrades over time is surprising. However, in ReDial, movies mentioned at beginning of conversations are, on average, very popular. But, as the conversation progresses recommendations are better informed and become more specific and hence harder to predict in absolute terms.

At the start of the conversation we can evaluate the performance of models in the user cold-start setting. Without surprise, BED's performance is the lowest since it only uses previous ratings. The results also show that KBRD is unable to use the little information available. The performance of KBAED is also low early in the conversation, but since it can use a genre input when available, it outperforms BED. Further, models using the complete text available through different implementations of BERT show even better performances as they can use all available information (e.g. all available attributes of movies and their associated sentiments) even including the nuances expressible in language.

We notice that two-movie mentions provides enough data for the encoder-encoder models (BED and KBAED) to match the performance of most text-based ones except for the performance of BERT UnI which is only matched by the end of the conversation by KBRD.

5.1.2 Item Cold Start. To analyze the performance of the models on cold-start items, we consider the ability of models to predict movies that were never seen in the train set, as discussed above. These results are presented next to the full results in Table 1 (in parentheses). We see that BERT U cannot rank unseen movies well, which is to be expected since it does not learn an explicit item representation. Results suggest that BERT UnI would provides the best prediction for unseen items given its joint representation of users and items. Further, the performance of BERT UnI on new items contributes to its overall superior performance (Table 1).

Table 2. BERT MF Learned output - Pooler VS Learned

Model	Output	R@1	R@10	R@50	NDCG@100
Learned	Pooler	0.88	6.66	15.52	0.0679
Learned	Average	2.10	12.02	28.79	0.1176

Table 3. Training time (20 R^-) and time complexity of BERT models to obtain predictions for $|U|$ users and $|V|$ items, where B_r is the complexity of the forward-pass of BERT.

Model	Train (hours)	Prediction
BERT U	19.2	$\lvert\mathcal{U}\rvert B_r$
BERT MF Fixed	38.7	$\lvert\mathcal{U}\rvert B_r + \lvert\mathcal{V}\rvert B_r$
BERT MF Learned	172	$\lvert\mathcal{U}\rvert B_r + \lvert\mathcal{V}\rvert B_r$
BERT UnI	44	$\lvert\mathcal{U}\rvert\lvert\mathcal{V}\rvert B_r$

5.1.3 Hyper-parameter exploration: Pooler Output VS Last Hidden Layer Average. Table 2 reports a comparison between two methods for obtaining representations from BERT: the pooler and averaging its last hidden layer. We compare these two strategies using BERT MF Learned models with 40 R^- and i_m. Results suggest that averaging the last hidden layer provides a better representation, as suggested in the documentation of the Hugging Face group [11]. Note that the difference between Bert MF Learned results in Tables 1 and 2 is caused by using i_f in the former and i_m in the latter, emphasizing the importance of using all available item features.

5.1.4 Hyper-parameter exploration: Impact of Random Ratings (R^-). We now explore the impact of the number of items that we randomly set to 0 for training. Figure 2a shows the performance of BERT UnI as a function of R^- (x-axis). We see a monotonic benefit in augmenting R^- for Recall@1 and Recall@10. However, the performance of Recall@50 actually diminishes when increasing R^- above 20. NDCG@100 which gives more importance to the top ranks shows mixed results. Overall, it seems like having more counter-examples (higher R^-) is beneficial to ensure the quality of short recommendation lists. This intuition is confirmed by the best results for Recall@1 obtained by BERT U, a model that uses softmax, which is similar to setting all unmentioned items at 0 (i.e. huge R^-).

5.1.5 Computational resources for training and serving recommendations. Even BERT fine-tuning uses significant computing resources. In Table 3 we report training time and resource consumption of the BERT-based models. For BERT MF and BERT UnI, the time depends on the number of Random Ratings (R^-). We present results for 20 R^-. BERT MF Learned requires learning two separate BERT models at the same time, one for users, one for items, which uses more memory. It forces us to reduce batch size and increases wall-clock training time.

We report the time complexity of making predictions for $|\mathcal{U}|$ user and $|\mathcal{V}|$ items in the last column of Table 3. Recall that BERT is composed of thousands of feed-forward networks and millions of dot products and so the extra time to computer the MLP and a small number of extra dot products is negligible. We denote the time complexity of a forward pass through BERT as B_r. For BERT U, a single forward-pass gives an embedding which can be decoded to obtain predicted ratings for all items, hence it costs $|\mathcal{U}|B_r$ to compute. For both BERT MF models, all users and all items have to go through their respective BERT once to get their representations and their combination is obtained through a dot product or MLP (both negligible time), hence $|\mathcal{U}|B_r + |\mathcal{V}|B_r$. For BERT UnI, since each combination of user and item has to go through BERT, we have $|\mathcal{U}||\mathcal{V}|B_r$.

6 CONCLUSION AND FUTURE WORK

We studied the task of recommendations from dialogue, a core component of conversational recommender systems. We show that new advances in language modeling provide strong performance in several settings including user and item cold-start at different training and serving costs.

REFERENCES

[1] Qibin Chen, Junyang Lin, Yichang Zhang, Ming Ding, Yukuo Cen, Hongxia Yang, and Jie Tang. 2019. Towards Knowledge-Based Recommender Dialog System. arXiv:cs.CL/1908.05391

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:cs.CL/1810.04805

[3] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. *CoRR* abs/1708.05031 (2017). arXiv:1708.05031 http://arxiv.org/abs/1708.05031

[4] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web* (Perth, Australia) *(WWW '17)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 173–182. https://doi.org/10.1145/3038912.3052569

[5] Raymond Li, Samira Ebrahimi Kahou, Hannes Schulz, Vincent Michalski, Laurent Charlin, and Chris Pal. 2018. Towards Deep Conversational Recommendations. In *Advances in Neural Information Processing Systems 31 (NIPS 2018)*.

[6] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. arXiv:stat.ML/1802.05814

[7] Julian McAuley Noveen Sachdeva. 2020. How useful are reviews for recommendation? A critical review and potential improvements. In *SIGIR*.

[8] Ruslan Salakhutdinov and Andriy Mnih. 2007. Probabilistic Matrix Factorization. In *Proceedings of the 20th International Conference on Neural Information Processing Systems* (Vancouver, British Columbia, Canada) *(NIPS'07)*. Curran Associates Inc., Red Hook, NY, USA, 1257–1264.

[9] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. AutoRec: Autoencoders Meet Collaborative Filtering. In *Proceedings of the 24th International Conference on World Wide Web* (Florence, Italy) *(WWW '15 Companion)*. Association for Computing Machinery, New York, NY, USA, 111–112. https://doi.org/10.1145/2740908.2742726

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. arXiv:cs.CL/1706.03762

[11] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. arXiv:cs.CL/1910.03771

[12] Hong-Jian Xue, Xin-Yu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. 2017. Deep Matrix Factorization Models for Recommender Systems. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence* (Melbourne, Australia) *(IJCAI'17)*. AAAI Press, 3203–3209.