Understanding the Effectiveness of Reviews in E-commerce Top-N Recommendation

Zhichao Xu
zhichao.xu@utah.edu
University of Utah

Hansi Zeng
hanszeng@cs.utah.edu
University of Utah

Qingyao Ai
aiqy@cs.utah.edu
University of Utah

Abstract

Modern E-commerce websites contain heterogeneous sources of information, such as numerical ratings, textual reviews and images. These information can be utilized to assist recommendation. Through textual reviews, a user explicitly express her affinity towards the item. Previous researchers found that by using the information extracted from these reviews, we can better profile the users' explicit preferences as well as the item features, leading to the improvement of recommendation performance. However, most of the previous algorithms were only utilizing the review information for explicit-feedback problem i.e. rating prediction, and when it comes to implicit-feedback ranking problem such as top-N recommendation, the usage of review information has not been fully explored. Seeing this gap, in this work, we investigate the effectiveness of textual review information for top-N recommendation under E-commerce settings. We adapt several SOTA review-based rating prediction models for top-N recommendation tasks and compare them to existing top-N recommendation models from both performance and efficiency. We find that models utilizing only review information can not achieve better performances than vanilla implicit-feedback matrix factorization method. When utilizing review information as a regularizer or auxiliary information, the performance of implicit-feedback matrix factorization method can be further improved. However, the optimal model structure to utilize textual reviews for E-commerce top-N recommendation is yet to be determined.

Keywords

Recommender System, Top-N Recommendation, Implicit Feedback, Reproducibility

ACM Reference Format:
Zhichao Xu, Hansi Zeng, and Qingyao Ai. 2021. Understanding the Effectiveness of Reviews in E-commerce Top-N Recommendation. In *Proceedings of the 2021 ACM SIGIR International Conference on the Theory of Information Retrieval (ICTIR '21), July 11, 2021, Virtual Event, Canada.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3471158.3472258

ICTIR '21, July 11, 2021, Virtual Event, Canada

ACM ISBN 978-1-4503-8611-1/21/07…$15.00
https://doi.org/10.1145/3471158.3472258

1 Introduction

There are two major lines of work in the study of recommender systems, rating prediction and top-N recommendation. In several recommendation applications, a user will give a numerical rating to the item after the interaction, such as 1-10 star ratings at the movie rating website IMDB. Through the numerical rating score, the user's level of affinity is expressed explicitly, so it is often formulated as an explicitly feedback problem. The task of rating prediction aims to accurately predict this numerical rating. Top-N recommendation problem, on the other hand, focus on predicting what items the user is likely to interact with. The user implicitly express her affinity towards the item by making the interact-or-not decision, so it is often formulated as an implicit feedback problem. Different from some rating applications, i.e. IMDB, in modern E-commerce websites, the user needs to purchase the item before leaving a rating accompanied with textual reviews or images. This first-purchase-then-review nature makes E-commerce top-N recommendation challenging since many other factors i.e. the price of the item, are taken into consideration when making the purchase decision.

A good news is that modern E-commerce websites contain heterogeneous sources of information, i.e. numerical ratings, textual reviews, images, which can be utilized to help with recommendation. Through textual reviews, a user explicitly express her affinity towards the item. In both rating prediction and top-N recommendation, review data has been widely identified to be useful in improving recommendation performance. Ganu et al. [10] found that text information can be used to assist the recommendation procedure. Later, researchers [2, 3, 5, 6, 8, 11, 12, 19, 24, 33–37] kept making progress in effectively incorporating review information into the task of recommendation. However, as pointed out by Peña et al. [25], the exploration to utilize textual reviews for recommendation are limited to rating prediction, and the effectiveness of review-based models for top-N recommendation is still underexplored. This gap serves as our motivation for our work.

In this work, we focus on the discussion of the better model designs to utilize textual reviews for E-commerce top-N recommendation. We replicate several state-of-the-art review-based rating prediction algorithms to make them suitable for top-N recommendation. Furthermore, we also include existing top-N recommendation models and draw comparison from both performance and inference speed. Our experiment results show that while reviews can be effective in the rating prediction task where users explicitly express their preferences, algorithms purely relying on features extracted from review data are outperformed by vanilla implicit feedback matrix factorization method. On the other hand, using review information as an additional regularizer or as auxiliary information can further improve matrix factorization's performance. Also,

review-based models with complex neural structures are of high time complexity and are not suitable for online top-N recommendation in practice. The optimal model structure to utilize textual reviews for E-commerce top-N recommendation is yet to be determined.

2 Related Work

In both item-rating applications i.e. IMDB and E-commerce websites i.e. Amazon, users explicitly express their affinities towards items by giving numerical scores and leaving textual reviews. Existing exploration in utilizing textual reviews for recommendation can be roughly classified into two categories: (1) text-as-feature: this line of research focused on utilizing textual information as the an information source to build user/item representations and conduct recommendations. Ganu [10] et al. propose to extract aspects from textual reviews to assist with rating prediction. Wang et al. [32] trained a topic model from content of scientific articles and combine it with the rating matrix to recommend. EFM [37] conducted aspect-level sentiment analysis to extract user's preference and product's quality on specific product features, then incorporate the results into matrix factorization framework to recommend. TriRank [12] modeled the user-aspect-item relations through a tripartite graph and cast the recommendation task to vertex ranking. DeepCoNN [39] first utilizes a deep neural network to learn user and item representations separately and calculate the recommendation score using a similarity function. Later, researchers [4, 5, 11, 34, 35] applied more complex neural architectures to learn information from textual reviews and further improve the recommendation performances. (2) text-as-regularizers: this line of research focused on utilizing textual information to regularize the user-item interactions and conduct recommendations. fLDA [1], extended from matrix factorization, regularizes both user and item factors in a supervised manner through explicit user features and the bag of words associated with each item. HFT [20] trains a topic model to regularize the latent factor model learned from the rating matrix. JRL utilizes the paragraph vector model [18] trained from textual reviews to regularizes the latent factor model trained from interaction matrix. To the best of our knowledge, it is the only model in this category that is originally designed for top-N recommendation task.

In this work, we focus on investigating the existing review-based text-as-feature models' performance in top-N recommendation. The closest work to ours is from Pena et al. [25]. They pointed out the exploration to utilize textual reviews for top-N recommendation is limited. Instead of proposing a new model to utilize textual reviews, we focus on studying existing works from both performance and efficiency. We hope to get insights from the results and inspire the community.

3 Methodology

3.1 Problem Formulation

We give a summary of notations used in this section in Table 1. Let $|\mathcal{U}|$ and $|\mathcal{I}|$ denote the number of users and items, respectively. We define the user-item rating matrix from users' explicit feedback as $\mathcal{R}^{|\mathcal{U}|\times|\mathcal{I}|}$ where each interaction tuple (u, i) is associated with an numerical rating r_u^i and a piece of textual review γ_u^i explaining why the user u gave such rating r_u^i to item i. Rating prediction task aims to minimize the pointwise MSE loss between $\hat{r}_u^i$ and r_u^i:

Table 1: A summary of notations

$u, \mathcal{U}$	user, user set
$i, \mathcal{I}$	item, item set
r_u^i	ground-truth rating score of user-item pair (u, i)
$\hat{r}_u^i$	predicted rating score of user-item pair (u, i)
γ_u^i	textual review γ given by u to i
Γ_u	reviews set given by user u
Γ_i	reviews set given to item i
β_i	item bias
$\overrightarrow{u}$	user latent factor
$\overrightarrow{i}$	item latent factor
$\mathcal{Y}$	positive user-item pairs set
$\mathcal{Y}^-$	negative user-item pairs set
y_u^i	predicted ranking score of user-item pair (u, i)

$$\frac{1}{\mathcal{Y}} \sum_{(u,i)\in\mathcal{Y}} ||\hat{r}_u^i - r_u^i||^2 \tag{1}$$

where $\mathcal{Y}$ is the set of (u, i) pairs that user u has rated item i.

In contrast, Top-N recommendation task aims to provide each user with a set of N items from a large set of items. Rendle et al. [26] proposed pairwise loss function (BPR) and has been widely used in top-N Recommendation from implicit feedback. Let $\mathcal{Y}^-$ be the set of negative (u, i) pairs (e.g., if we consider user purchase as ground truth, then $\mathcal{Y}^-$ is the unpurchased (u, i) pairs). Given $\mathcal{Y}$ and $\mathcal{Y}^-$, the pairwise binary cross-entropy loss of BPR can be formulated as:

$$\mathcal{L} = \sum_{(u,i)\in\mathcal{Y},(u,j)\sim\mathcal{Y}^-} P_u(i>j)\log(P_u(i>j)) + (1-P_u(i>j))\log(1-P_u(i>j)) \tag{2}$$

where (u, j) is randomly sampled from $\mathcal{Y}^-$ based on (u, i), and $P_u(i > j)$ is defined as

$$P_u(i>j) = \frac{1}{1+\exp(y_u^i - y_u^j)}$$

In our work, we modify the original algorithms designed for rating prediction by changing their loss functions from pointwise loss to pairwise loss for top-N recommendation. We also implement the corresponding negative sampling strategy for efficient training.

3.2 Recommendation Models

To investigate the effectiveness of explicit review information in the task of top-N recommendation, we include a variety of state-of-the-art review-based models. We also consider to compare the review-based models with some classical implicit-feedback models for better evaluation. The models are classified into three different categories: the implicit-feedback interaction-based models, models using text information as feature, and models using text information as regularizer. We make necessary adjustments to adapt these models for top-N recommendation. The list of models we used in experiments is as follows:

3.2.1 Interaction-based Models Interaction-based models simply model the historical interactions between users and items. They utilize a latent factor model, where each dimension of the latent factor is designed to represent a specific feature of the users and the items. Then a similarity function (mostly inner product) is applied to calculate the similarity between the user latent factor and the item latent factor and get the affinity score.

Bayesian Personalized Ranking Matrix Factorization (BPR-MF) [16, 26] BPR-MF follows the vanilla matrix factorization setup where each user & item is represented by a latent factor. We use the pairwise loss from implicit feedback to train the model, and the final affinity score given by user u to item i is predicted as

$$y_u^i = \beta_i + \overrightarrow{u} \cdot \overrightarrow{i} \tag{3}$$

and we optimize the parameters by maximize Equation 2.

BPR Generalized Matrix Factorization (BPR-GMF) [13] BPR-GMF adds a generalized function to model the complex interactions between user and item latent factors. Specifically, the affinity score is

$$y_u^i = \beta_i + F(\overrightarrow{u} \cdot \overrightarrow{i}) \tag{4}$$

where F is a deep neural network structure. In our implementation, we use a multi-layer densely-connected neural network (MLP) and we tune the number of MLP layers for best performance.

3.2.2 Text-as-regularizer Models
Text-as-regularizer models follows a traditional matrix factorization setup, and apply an additional objective function utilizing the representations learned from textual information to regularize the latent factors. The model can be trained offline and in the inference stage, the affinity score is computed by a simple inner product function. Thus they are considered an effective way for online recommendation.

Hidden Factors and Topics (BPR-HFT) [20] In addition to the BPR loss, BPR-HFT utilizes textual reviews to train an latent dirichlet allocation topic model and minimize the corpus likelihood to regularize the latent factors used in matrix factorization.

Joint Representation Learning (JRL) [36] Motivated by the paragraph vector model [18], JRL utilizes an additional generative loss built from textual reviews to regularize the latent vectors learned from implicit-feedback interaction matrix.

3.2.3 Text-as-feature Models Text-as-feature models utilize representations learned from textual reviews to build user/item feature vectors. A similarity function is then applied to calculate the affinity scores.

BERT-Rep [23]: For each user, all her reviews are aggregated to form a long document and input to BERT [7] to encode, and the output layer's [CLS] token vector is used as the user representation. The same procedure is applied to get the item representation. Then we apply dot product over the user/item representation to predict the corresponding affinity score as

$$BERT(Rep)(u,i) = \overrightarrow{u}_{cls}^{last} * \overrightarrow{i}_{cls}^{last} \tag{5}$$

where $*$ denotes inner product and *last* denotes the last layer of BERT's Transformer network.

Deep Co-operative Neural Network (DeepCoNN) [39]: Different from previous algorithms [3, 8, 20], DeepCoNN utilizes a CNN-based neural architecture to extract information from textual reviews. Specifically, all the reviews in Γ_u are concatenated as a document, then a TextCNN architecture [38] is applied to extract the latent feature factors from review documents to form the user feature factor. The same procedure is applied to get the item feature factor.

Neural Attentive Rating Regression (NARRE) [5]: Also based on TextCNN, NARRE additionaly learns a review-level attention weights distribution of each single piece of review in user/item document and it achieves better performance than DeepCoNN in terms of rating prediction.

Multi-Pointer Co-Attention Network (MPCN) [31]: MPCN selects informative reviews from Γ_u & Γ_i by review-level pointers using co-attention technique, and selects informative word-level representations by applying word-level pointers over selected reviews.

Asymmetrical Hierarchical Networks with Attentive Interactions (AHN) [9]: AHN treats user and item asymmetrically and builds representations hierarchically from sentence level and review level. It also dynamically models the interaction using co-attention mechanism.

Interpretable Convolutional Neural Networks with Dual Local and Global Attention (Dual-ATT) [30]: Dual-ATT applies a local and a global attentions to encode the user(item) documents separately. The final representation is learned by concatenating representations learned from both local and global attention modules.

A Zero-Attentional Relevance Matching Network for Review Modeling (ZARM) [35]: ZARM combines the concept of relevance matching and semantic matching, and uses an zero-attention schema to dynamically model user & item representations.

Attentive Aspect Modeling for Review-aware Recommendation (AARM) [11]: AARM is the state-of-the-art review-based model for top-N recommendation. It utilizes a Phrase-level Sentiment Analysis toolkit [37] to first extract aspect-opinion-sentiment triples from textual reviews. Then these triples are put into an attentive aspect-interaction module to learn the aspect-level interactions. The learned aspect-interaction vectors are concatenated with global-interaction latent factors learned from the implicit-feedback interaction matrix to compute the final affinity score. Note that AARM is intrinsically different from other text-as-feature models because it extracts information from review text only as additional features to be combined with a interaction-based model (e.g., a MF model).

4 Experiments

4.1 Dataset Description

We use three categories of data from Amazon Product Review Dataset [20][1]. Specifically, we want to investigate how text-based representation learning models perform in the cold-start scenario, so we include both 5-core and 0-core datasets. Here k-core means each user/item has at least k interactions in the dataset. A detailed statistics of the dataset is showed at Table 2. We use a randomized

[1] Amazon Product Review: http://jmcauley.ucsd.edu/data/amazon/links.html

user-level 7:3 split, namely, for each user, 70% of her total transactions are used for training and the rest for testing. We don't have a separate validation set since the interaction matrix is already very sparse in these datasets. To avoid the review information leaking problem mentioned by Catherine et al. [4], all the reviews in testset are not used in the training of the models.

Table 2: The basic statistics of the datasets

Dataset	#users	#items	#interactions	#density
Beauty-0core	1,210,271	249,274	2,023,070	6.71e-6
Beauty-5core	22,363	12,191	198,502	7.82e-4
Tools & Home-0core	1,212,047	260,657	1,926,047	6.09e-6
Tools & Home-5core	16,638	10,217	134,476	7.92e-4
Electronics-0core	4,201,696	476,001	7,824,482	6.91e-6
Electronics-5core	192,403	63,001	1,689,188	1.39e-4

4.2 Evaluation

We evaluate the ranking performances of all models using *Hit Rate* (HR) and *normalized Discounted Cumulative Gain* (nDCG). HR intuitively measures whether the test item is in the recommendation list and nDCG accounts for the position of the hit by assigning higher scores to hits at the top of the recommendation list. We report the HR@10 and nDCG@10 on both 0-core and 5-core datasets.

Previous implementations [13, 15, 22, 33] rank the ground truth items along with k randomly sampled negative items; According to Krichene and Rendle [17], this may lead to inconsistent results and may not be a fair comparison between algorithms. Yet, ranking all the items in the candidate items pool will lead to prohibitive computation cost for some review-based models. To reach a balance between consistency and efficiency, we use a two-stage retrieval & reranking strategy. In the retrieval stage, we use MF to retrieve top 1,000 items for each user, and in the reranking stage, for each user, we rerank these items along with the ground truth items. Through this two-stage strategy, we reduce the overall time complexity to ranking all items using complex text-based models. We also avoid the extreme case that randomly sampled negative test items are very irrelevant and leads to inconsistent recommendation performance in evaluation.

4.3 Implementation Details

4.3.1 Text Processing

We remove stopwords from the reviews and maintain a vocabulary of 50K most frequent words from the training corpus. To better catch the semantic information from textual reviews, we use Google's Word2Vec[2] 300-dimensional embeddings pretrained on 100 billion words from Google News [21]. For BERT-Rep method, top 512 tokens of each user & item document are used, and for other review-based methods except for AARM, top 1,000 tokens are used. For AARM, we use the Phrase-level Sentiment Analysis toolkit Sentires [3] to extract the aspect-opinion-sentiment triples from textual reviews. Specifically, we use the Word2Vec to convert the aspects into word/phrase embeddings.

[2]https://code.google.com/archive/p/word2vec/

[3]Sentires: https://github.com/evison/Sentires

4.3.2 Implementation

We implemented MF, GMF, DeepCoNN, MPCN, AHN, Dual-ATT, NARRE, ZARM using PyTorch [4]. For BPR-HFT and JRL, we used the implementation from the JRL Repo [5] and modified accordingly. For AARM, we used the implementation from AARM Repo [6] and modified accordingly. Our implementations will be available at https://github.com

4.3.3 Parameters & Hyperparameters

We train our models using Adam [14] and SGD [28]. We search the learning rate between 1e-1 and 1e-4, L2-regularization between 1e-1 and 1e-4, number of negative samples between 2 and 10, CNN window size between 3 and 10, dropout rate between 0.1 and 0.8, latent factor size between 16 and 128. In all the models we set the default latent factor size to 64 unless mentioned specifically.

5 Result & Analysis

5.1 Top-N Recommendation Performance

We show the ranking results in Table 3. We do not include the result of BERT model on 0-core dataset as the encoding takes too much time; performance-wise, we find:

Among interaction-based models, BPR-GMF achieves about the same performances as BPR-MF in all datasets; this is the same as what was reported by Rendle [27]. We argue that on sparse datasets like Amazon Product Reviews, BPR-GMF does not achieve significant improvement over BPR-MF because the deep MLP structure in equation 4 can not perfectly catch the complex interaction signals due to the lack of historical interactions for training.

Among text-as-feature models, we notice that AARM consistently outperform other models. For example, in 5-core Beauty dataset, AARM achieves 28.9% and 16.1% improvement in HR and nDCG respectively, compared with ZARM. We consider this performance boost is given by the global-interaction latent factors trained from the implicit-feedback interaction matrix. Aside from AARM, the performance difference between NARRE and ZARM is not statistically significant, so we consider they deliver about the same performance. Compared to complex structures such as attention mechanism used in MPCN, AHN and Dual-ATT, NARRE utilizes a simple review-level attention and proves to be both effective and computationally efficient.

In 5-core datasets, we notice AARM consistently outperforms interaction-based models. For example, in Electronics dataset, AARM achieves 5.1% and 9.1% improvement in HR and nDCG respectively, compared with BPR-GMF. This observation shows the aspect-interaction module can effectively capture the aspect-level features of the users and the items. Other text-as-feature models fail to deliver as good performance as interaction-based models. We also notice that text-as-regularizer models achieve the best performances overall compared with text-as-feature models and interaction-based models. For example, in Electronics dataset, JRL is 5.6% better in HR and 22.4% better in nDCG compared with AARM. Our findings are in accordance with Sachedeva's [29] argument that reviews are more effective as regularizer rather than as feature. Note that in

[4]PyTorch: https://pytorch.org/

[5]https://github.com/evison/JRL

[6]https://github.com/XinyuGuan01/Attentive-Aspect-based-Recommendation-Model

Table 3: The ranking performance (Hit Rate, nDCG) measured in %; We highlight the model with best performance. ♣ indicates its improvement over models in other two categories is statistically significant at 0.01 level with Paired t-test

Dataset		Tools & Home				Beauty				Electronics			
		0-core		5-core		0-core		5-core		0-core		5-core	
Metrics		Hit	nDCG	Hit	nDCG	Hit	nDCG	Hit	nDCG	Hit	nDCG	Hit	nDCG
interaction-based	BPR-MF	1.72	0.83	7.12	2.35	2.05	0.93	11.33	4.21	2.02	0.51	4.56	1.43
	BPR-GMF	1.74	0.89	7.04	2.38	2.01	0.94	11.01	4.37	2.00	0.52	4.71	1.49
text-as-regularizer	BPR-HFT	1.82	0.88	8.42	2.84	2.06	0.97	11.54	4.65	2.08	0.58	4.89	1.62
	JRL	1.93	0.97	**8.84**$^{\clubsuit}$	**3.01**$^{\clubsuit}$	2.22	1.04	**12.04**$^{\clubsuit}$	**5.03**$^{\clubsuit}$	**2.15**	**0.60**$^{\clubsuit}$	**5.23**$^{\clubsuit}$	**1.91**$^{\clubsuit}$
text-as-feature	BERT	-	-	5.35	1.69	-	-	8.87	4.05	-	-	3.67	1.05
	DeepCoNN	1.45	0.69	4.82	1.64	1.80	0.81	7.96	3.79	1.78	0.42	3.58	1.17
	MPCN	1.39	0.71	6.18	1.95	1.84	0.86	8.78	3.99	1.81	0.39	4.15	1.22
	AHN	1.45	0.73	6.13	1.92	1.86	0.86	8.92	4.05	1.79	0.41	4.21	1.25
	Dual-ATT	1.51	0.71	6.47	2.12	1.84	0.85	9.01	4.02	1.81	0.41	4.20	1.24
	NARRE	1.59	0.78	6.80	2.25	1.94	0.84	8.94	4.01	1.85	0.45	4.29	1.29
	ZARM	1.60	0.81	6.84	2.28	1.99	0.91	8.99	4.10	1.87	0.44	4.32	1.33
	AARM	**2.05**$^{\clubsuit}$	**1.01**	7.94	2.68	**2.31**	**1.09**	11.59	4.76	2.12	0.54	4.95	1.56

Table 4: The statistics of the reranking inference speed, measured in seconds per entry, with batch size 512

Model\Dataset	Tools & Home-5core	Beauty-5core	Electronics-5core
BPR-MF	0.004	0.004	0.004
BPR-GMF	0.012	0.012	0.012
BPR-HFT	-	-	-
JRL	0.005	0.005	0.005
DeepCoNN	0.079	0.082	0.081
AARM	0.203	0.162	0.312
NARRE	0.170	0.173	0.170
MPCN	0.255	0.257	0.255
AHN	0.330	0.332	0.330
Dual-ATT	0.276	0.278	0.275
ZARM	0.383	0.385	0.383

Sachdeva's experiments, NARRE outperforms interaction-based models in rating prediction while in our experiment, NARRE fails to outperform two interaction-based models. We argue this is because of the intrinsic difference between rating prediction and top-N recommendation. We further discuss this intrinsic difference in section 5.3.1.

Profiling cold-start users & items is a challenging task for E-commerce recommendation. Complex models suffer from overfitting problem and often don't perform well in cold-start scenario. We notice that in all three 0-core datasets, JRL achieves about the same performance as AARM, and significantly outperforms other text-as-feature models and interaction-based models. We consider AARM's good performance is mainly from pre-extracted aspects which are much more effective in cold-start scenario. Furthermore, we notice that other text-as-feature models don't deliver as good performance as interaction-based models. This indicates that without sufficient training data, complex text-as-feature models are not able to model the user preferences as well as item features. Among other text-as-feature models, we find DeepCoNN performs relatively better in 0-core datasets than in 5-core datasets. For example, DeepCoNN outperforms MPCN & AHN in Tools & Home 0-core dataset, and achieves about the same performance as MPCN, AHN & D-ATT in Electronics 0-core dataset. We argue that DeepCoNN's TextCNN structure gives it an edge over complex attention-based models and suffer less from overfitting.

5.2 Inference Speed

Real-time response is also necessary for online E-commerce recommendation. We report the comparison of inference speed at Table 4. We leave out the BPR-HFT here as its implementation is in C++ while all other models are implemented in Python. We find the inference speed of interaction-based and text-as-regularizer models are fast; and the inference speed of text-as-only-feature models are significantly slower. Such phenomenon can be expected since in the inference stage, text-as-regularizer models only need to find the corresponding user and item latent factors and compute the affinity score through a similarity function, and text-as-feature models are slow since complicated neural network structures such as TextCNN, attention are used. We also notice that AARM's inference speed will increase significantly when there are more aspects in the dataset. For example, there are in total 1,987 aspects in Tools & Home dataset and 690 aspects in Beauty dataset, and AARM's inference speed in Tools & Home is 25% more than in Beauty. Compared with the performance reported at Table 3, we conclude that AARM reaches the better balance between performance and computational complexity among all text-as-feature models. Overall, text-as-regularizer models are more computationally efficient.

5.3 Analysis

5.3.1 Rating prediction and top-N recommendation are intrinsically different tasks

We formulate the structure of text-as-feature model for rating prediction as:

$$Rep_u = f_u(\Gamma_u), Rep_i = f_i(\Gamma_i), \hat{r}_{u,i} = F(Rep_u, Rep_i) \tag{6}$$

Each dimension of the latent factor can be regarded as an aspect of the user/item profile, and the final function $F(\cdot,\cdot)$ can be regarded as calculating the similarity between the user factor and the item

factor; it varies from a simple inner product to a complicated deep neural network structure.

Explicit feedback problem i.e. problem aims to predict the level of affinity usually assume that the interaction has already happened, e.g. the user has watched the movie or has purchased the item. On the other hand, implicit feedback problem aims to predict whether the interaction will happen. Specifically, top-N recommendation aims to generate a short ranklist of items that the user may interact with to cover as many ground-truth items as possible. In modern E-commerce applications, this is more important as better top-N recommendation performance will increase the profit. Instead of fitting the absolute value of user-item ratings, user's preferences among different items are more important. We argue the intrinsic difference between two different types of tasks makes text-as-feature models designed specifically for explicit feedback problem e.g. rating prediction not suitable for implicit feedback problem, e.g. top-N recommendation, and existing model designs can not be migrated directly.

5.3.2 User/item latent representations built purely from textual reviews do not deliver good top-N recommendation performance

We observe that all text-as-feature models except for AARM fail to outperform vanilla matrix factorization model. We argue there may be four reasons that lead to text-as-feature models' underperformances: (1) the reviews are of low quality and do not include much useful information; (2) existing model structures are limited thus can not extract useful information from reviews; (3) in implicit feedback problem, there is a gap between user leaving a piece of textual review and her actual interaction decision, which means reviews do not necessarily reflect the user's actual preference. BERT is a state-of-the-art pretrained contextualized language model and is supposed to catch useful semantic information from texts, but our BERT-Rep baseline is still outperformed by both interaction-based and text-as-regularizer models by a large margin; (4) following previous points, we regard rating prediction as an explicit-feedback task and top-N recommendation as an implicit-feedback task. In textual reviews, users explicitly express their affinities to the items, so we argue that textual reviews are intrinsically more suitable to be used in rating prediction than in top-N recommendation. Many other factors such as the price of the item, the availability, etc need to be taken into consideration when making the interact-or-not decision.

So far we come to conclusion that purely relying on features extracted from textual reviews to build user/item representations is not a good approach for top-N recommendation task. JRL and AARM combines textual reviews with interactions data and achieves the best performance among all the models. This observation indicates that the combination of these two sources of information can deliver better performance. However, the optimum model structure is yet to be determined.

6 Conclusion & Future Work

In this work, we focus on the discussion of the better model structures to utilize textual reviews for E-commerce top-N recommendation. We adapt some existing text-as-features rating prediction models for top-N recommendation task. We also include existing top-N recommendation models and draw comparison from both performance and inference speed. We find that due to the intrinsic difference between two different tasks, existing text-as-feature rating prediction model designs are not suitable for E-commerce top-N recommendation. We further discuss the possible reasons behind text-as-feature models' underperformances. We come to conclusion that among existing review-based models, those only using textual reviews as features fail to outperform vanilla interaction-based models, and combining textual reviews with historical interactions data can deliver better performances. Overall, text-as-regularizer models seem to be better at utilizing textual review information, giving better performances without increasing inference time complexity. We also provide our implementation of multiple strong baselines, hoping to shed light to the reproducibility issue in current recommender system research community.

Our future work will be designing better and more efficient models to utilize textual reviews for implicit feedback top-N recommendation task.

7 Acknowledgement

This work was supported in part by the School of Computing, University of Utah and in part by NSF IIS-2007398. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

References

[1] Deepak Agarwal and Bee-Chung Chen. 2010. fLDA: matrix factorization through latent dirichlet allocation. In *Proceedings of the third ACM international conference on Web search and data mining*. 91–100.

[2] Qingyao Ai, Vahid Azizi, Xu Chen, and Yongfeng Zhang. 2018. Learning heterogeneous knowledge base embeddings for explainable recommendation. *Algorithms* 11, 9 (2018), 137.

[3] Yang Bao, Hui Fang, and Jie Zhang. 2014. Topicmf: simultaneously exploiting ratings and reviews for recommendation.. In *Aaai*, Vol. 14. Citeseer, 2–8.

[4] Rose Catherine and William Cohen. 2017. Transnets: Learning to transform for recommendation. In *Proceedings of the eleventh ACM conference on recommender systems*. 288–296.

[5] Chong Chen, Min Zhang, Yiqun Liu, and Shaoping Ma. 2018. Neural attentional rating regression with review-level explanations. In *Proceedings of the 2018 World Wide Web Conference*. 1583–1592.

[6] Xu Chen, Zheng Qin, Yongfeng Zhang, and Tao Xu. 2016. Learning to rank features for recommendation over multiple categories. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. 305–314.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[8] 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 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 193–202.

[9] Xin Dong, Jingchao Ni, Wei Cheng, Zhengzhang Chen, Bo Zong, Dongjin Song, Yanchi Liu, Haifeng Chen, and Gerard de Melo. 2020. Asymmetrical hierarchical networks with attentive interactions for interpretable review-based recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 7667–7674.

[10] Gayatree Ganu, Noemie Elhadad, and Amélie Marian. 2009. Beyond the stars: improving rating predictions using review text content.. In *WebDB*, Vol. 9. Citeseer, 1–6.

[11] Xinyu Guan, Zhiyong Cheng, Xiangnan He, Yongfeng Zhang, Zhibo Zhu, Qinke Peng, and Tat-Seng Chua. 2019. Attentive aspect modeling for review-aware recommendation. *ACM Transactions on Information Systems (TOIS)* 37, 3 (2019), 1–27.

[12] Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. 2015. Trirank: Review-aware explainable recommendation by modeling aspects. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. 1661–1670.

[13] 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*. 173–182.

[14] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[15] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. 426–434.

[16] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[17] Walid Krichene and Steffen Rendle. 2020. On sampled metrics for item recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1748–1757.

[18] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*. 1188–1196.

[19] Guang Ling, Michael R Lyu, and Irwin King. 2014. Ratings meet reviews, a combined approach to recommend. In *Proceedings of the 8th ACM Conference on Recommender systems*. 105–112.

[20] Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*. 165–172.

[21] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546* (2013).

[22] Xia Ning and George Karypis. 2011. Slim: Sparse linear methods for top-n recommender systems. In *2011 IEEE 11th International Conference on Data Mining*. IEEE, 497–506.

[23] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. *CoRR* abs/1901.04085 (2019). arXiv:1901.04085 http://arxiv.org/abs/1901.04085

[24] Zhimeng Pan, Wenzheng Tao, and Qingyao Ai. 2020. Review Regularized Neural Collaborative Filtering. *arXiv preprint arXiv:2008.13527* (2020).

[25] Francisco J Peña, Diarmuid O'Reilly-Morgan, Elias Z Tragos, Neil Hurley, Erika Duriakova, Barry Smyth, and Aonghus Lawlor. 2020. Combining Rating and Review Data by Initializing Latent Factor Models with Topic Models for Top-N Recommendation. In *Fourteenth ACM Conference on Recommender Systems*. 438–443.

[26] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618* (2012).

[27] Steffen Rendle, Walid Krichene, Li Zhang, and John Anderson. 2020. Neural collaborative filtering vs. matrix factorization revisited. In *Fourteenth ACM Conference on Recommender Systems*. 240–248.

[28] Herbert Robbins and Sutton Monro. 1951. A stochastic approximation method. *The annals of mathematical statistics* (1951), 400–407.

[29] Noveen Sachdeva and Julian McAuley. 2020. How Useful are Reviews for Recommendation? A Critical Review and Potential Improvements. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1845–1848.

[30] Sungyong Seo, Jing Huang, Hao Yang, and Yan Liu. 2017. Interpretable convolutional neural networks with dual local and global attention for review rating prediction. In *Proceedings of the eleventh ACM conference on recommender systems*. 297–305.

[31] Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2018. Multi-pointer co-attention networks for recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2309–2318.

[32] Chong Wang and David M Blei. 2011. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. 448–456.

[33] Zhichao Xu, Yi Han, Yongfeng Zhang, and Qingyao Ai. 2020. E-commerce Recommendation with Weighted Expected Utility. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 1695–1704.

[34] Hansi Zeng and Qingyao Ai. 2020. A Hierarchical Self-attentive Convolution Network for Review Modeling in Recommendation Systems. *arXiv preprint arXiv:2011.13436* (2020).

[35] Hansi Zeng, Zhichao Xu, and Qingyao Ai. 2021. A Zero Attentive Relevance Matching Networkfor Review Modeling in Recommendation System. *arXiv preprint arXiv:2101.06387* (2021).

[36] Yongfeng Zhang, Qingyao Ai, Xu Chen, and W Bruce Croft. 2017. Joint representation learning for top-n recommendation with heterogeneous information sources. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 1449–1458.

[37] 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 37th international ACM SIGIR conference on Research & development in information retrieval*. 83–92.

[38] Ye Zhang and Byron Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820* (2015).

[39] Lei Zheng, Vahid Noroozi, and Philip S Yu. 2017. Joint deep modeling of users and items using reviews for recommendation. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. 425–434.