Personalized Neural Embeddings for Collaborative Filtering with Unstructured Text

Guangneng Hu, Yu Zhang
Department of Computer Science and Engineering
Hong Kong University of Science and Technology
Hong Kong, China
{njuhgn,yu.zhang.ust}@gmail.com

Abstract

Collaborative filtering (CF) is the key technique for recommender systems. Pure CF approaches exploit the user-item relational data (e.g., clicks, likes, and views) only and hence suffer from the data sparsity issue. Luckily, items are usually associated with unstructured text such as article abstracts and product reviews. We develop a personalized neural embedding framework to exploit both user-item relational interactions and words in unstructured text seamlessly. We learn such embeddings of users, items, and words jointly, and predict the user preferences on items based on these learned representations. The proposed framework, Collaborative Memory networks (CoMem), estimates the probability that a user will like an item by two terms, behavior factors and semantic factors. On two real-world datasets, CoMem shows better performance than state-of-the-art.

1 Introduction

Recommender systems are widely used in various domains and e-commerce platforms, such as to help consumers buy products at Amazon, watch videos on Youtube, and read articles on Google News. They are useful to alleviate the information overload and improve user satisfaction. Given the history records of consumers such as the product transactions and movie watching, collaborative filtering (CF) is among the most effective approaches based on the simple intuition that if users rated items similarly in the past then they are likely to rate items similarly in the future [18].

The history records include both implicit (e.g., purchase and clicks) and explicit (e.g., likes/dislikes and ratings) feedback which can be represented as a user-item interaction matrix. Usually, the observed user-item interactions are partial with a large portion remaining not recorded. The goal of recommendation is to predict the user preferences on the missing item interactions. This setting requires to complete the partial observed rating matrix. Matrix factorization (MF) techniques which can learn the latent factors for users and items are the main cornerstone for CF [16, 13, 14]. It is effective and flexible to be integrated with additional data sources. Recently, neural networks like multilayer perceptron (MLP) are used to learn the interaction function from data [5, 4, 9] with the power of learning highly nonlinear relationships between users and items. MF and neural CF exploit the user-item behavior interactions only and hence suffer from the data sparsity and cold-start issues.

Items are usually associated with content information such as unstructured text, like the news articles and product reviews. These additional sources which provide independent and diverse information are essential for recommendation beyond user-item interaction data, and hence can alleviate the data sparsity issue [7]. For application domains like recommending research papers and news articles, the unstructured text associated with the item is its text content [21, 22, 1]. Other domains like recommending products, the unstructured text associated with the item is its user reviews which justify the rating behavior [15, 8, 10]. These methods adopt topic modelling techniques or neural networks to exploit the item content leading to performance improvement.

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

The typical way of exploiting text is to extract a feature vector from the text document by averaging the word embeddings pre-trained from a large corpus such as Wikipedia [10, 8]. These methods separate the extraction of text feature from the learning of user-item interaction and hence the two processes can not benefit from each other. Others learn a topic vector using topic modelling [21, 15, 2] by aligning behavior factors and topic factors with a link function such as softmax and offset. Recently, neural networks are used to learn a representation from the text using autoencoders [22, 23], recurrent networks [1], and convolutional networks [24, 3]. These methods treat different words in the document as equal importance and do not match word semantics with the specific user.

In this paper, we propose a novel neural framework to exploit interaction relational data and content information seamlessly. The proposed framework, Collaborative Memory networks (CoMem), fuses semantic representations learnt from unstructured text with behavior representations learnt from user-item interactions for effective estimation on user preferences' items. CoMem estimates the probability that a user will like an item by two factors. The *behavior factor* is to capture the personalized preference of the user to the given item. And the *semantic factor* is to capture the high-level representation attentively extracted from the unstructured text matching word semantics with user preferences. These two effective representations are used to learn user preferences on items.

To model the behavior factor, we adopt the same approach as neural CF, which learns the user-item nonlinear interaction relationships using a neural network (CFNet). To model the semantic factor, we adopt memory networks to match word semantics with the specific user via the attention mechanism inherent in the memory module (MemNet), determining which words are highly relevant to the user preferences. As far as we know, CoMem is the first neural embedding model that integrates relational interactions data with unstructured text by bridging neural CF and memory networks.

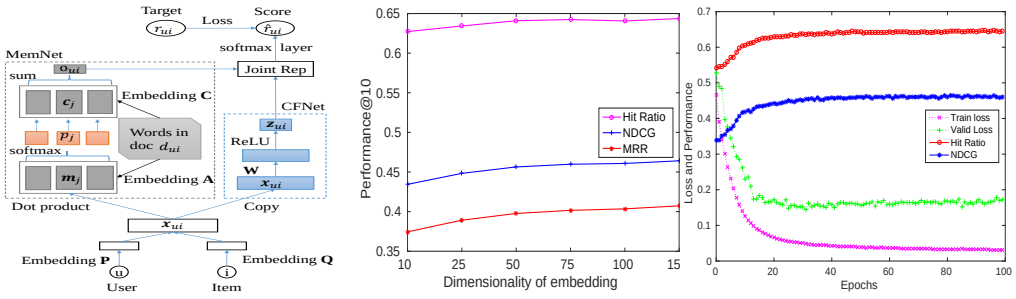
2 The CoMem Framework

Problem formulation For collaborative filtering with implicit feedback, there is a binary matrix $\boldsymbol{R} \in \mathbb{R}^{m \times n}$ to describe user-item interactions where each entry $r_{ui} \in \{0, 1\}$ is 1 (called observed entries) if user u has an interaction with item i and 0 (unobserved) otherwise. Denote the set of m-sized users by $\mathcal{U}$ and n items by $\mathcal{I}$. Usually the interaction matrix is very sparse since a user $u \in \mathcal{U}$ only consumed a very small subset of all items. Similarly for the task of item recommendation, each user is only interested in identifying top-K items. The items are ranked by their predicted scores: $\hat{r}_{ui} = f(u, i|\Theta)$, where f is the interaction function and Θ denotes model parameters.

For neural CF, neural networks (NNs) are used to parameterize f and learn it from interaction data: $f(\boldsymbol{x}_{ui}|\boldsymbol{P}, \boldsymbol{Q}, \theta_f) = \phi_o(\phi_1(\boldsymbol{x}_{ui}))$, where input $\boldsymbol{x}_{ui} = [\boldsymbol{P}^T \boldsymbol{x}_u, \boldsymbol{Q}^T \boldsymbol{x}_i] \in \mathbb{R}^d$ is concatenated from embeddings of user and item, which are projections of their one-hot encodings $\boldsymbol{x}_u \in \{0, 1\}^m$ and $\boldsymbol{x}_i \in \{0, 1\}^n$ by embedding matrices $\boldsymbol{P} \in \mathbb{R}^{m \times d/2}$ and $\boldsymbol{Q} \in \mathbb{R}^{n \times d/2}$, respectively ($d = d/2 + d/2$, $\Theta = \{\boldsymbol{P}, \boldsymbol{Q}, \theta_f\}$). Output and the hidden layer are computed by ϕ_o and ϕ_1 in a feedforward NN. Items are associated with unstructured text like reviews of products. For the document of item i by user u, denote the words in it as $d_{ui} = [w_j]_{j=1}^{l}$ where $l = |d_{ui}|$ and words come from a vocabulary $\mathcal{V}$. Neural CF can be extended to leverage text and the interaction function has the form of $f(u, i, d_{ui}|\Theta)$.

Architecture The architecture for the proposed CoMem model is illustrated in Figure 1a. Besides the layers of input, embedding, and output, CoMem consists of a CF network (CFNet) to learn nonlinear interaction relationships between users and items and of a memory network (MemNet) to learn text representations matching word semantics with specific users. The information flow in CoMem goes from the input (u, i) to the output $\hat{r}_{ui}$ through the following five layers: **Input:** $(u, i) \rightarrow \mathbb{1}_u, \mathbb{1}_i$ This module encodes user-item interaction indices. We adopt the one-hot encoding. It takes user u and item i, and maps them into one-hot encodings $\mathbb{1}_u \in \{0, 1\}^m$ and $\mathbb{1}_i \in \{0, 1\}^n$ where only the element corresponding to that index is 1 and all others are 0. **Embedding:** $\mathbb{1}_u, \mathbb{1}_i \rightarrow \boldsymbol{x}_{ui}$ This module embeds one-hot encodings into continuous representations $\boldsymbol{x}_u = \boldsymbol{P}^T \mathbb{1}_u$ and $\boldsymbol{x}_i = \boldsymbol{Q}^T \mathbb{1}_i$ by embedding matrices $\boldsymbol{P}$ and $\boldsymbol{Q}$ respectively, and then concatenates them as $\boldsymbol{x}_{ui} = [\boldsymbol{x}_u, \boldsymbol{x}_i]$, to be the input of following building blocks. **CFNet:** $\boldsymbol{x}_{ui} \rightsquigarrow \boldsymbol{z}_{ui}$ This module is a pure CF approach to exploit user-item interaction data. It takes the continuous representations from the embedding module and then transforms to a final *behavior factor* representation: $\boldsymbol{z}_{ui} = \text{ReLU}(\boldsymbol{W}\boldsymbol{x}_{ui} + \boldsymbol{b})$, where $\text{ReLU}(x) = \max(0, x)$ is the activation function and $\boldsymbol{W}$ and $\boldsymbol{b}$ are the weight and bias parameters. **MemNet:** $\boldsymbol{x}_{ui} \rightsquigarrow \boldsymbol{o}_{ui}$ This module is to model the item content with the guidance of interaction data. The item content is modelled by memories. It takes both representations from the embedding module and the text associated with the corresponding user-item to a final *semantic factor* representation:

$$\boldsymbol{o}_{ui} = \text{MemNet}(\boldsymbol{x}_{ui}, d_{ui}), \tag{1}$$



(a) Architecture. (b) Embedding. (c) Loss.

Figure 1: (a): Architecture of CoMem. (b): Embedding dimensionality. (c): Loss and performance

where MemNet denotes the computing function in the memory module (see Sec. 2.1). **Output:** $[\boldsymbol{z}_{ui}, \boldsymbol{o}_{ui}] \rightarrow \hat{r}_{ui}$ This module predicts the score $\hat{r}_{ui}$ for the given user-item pair based on the concatenated representation of behavior factor and semantic factor from the CFNet and MemNet: $\boldsymbol{y}_{ui} = [\boldsymbol{z}_{ui}, \boldsymbol{o}_{ui}]$. The output is the probability that the input pair is a positive interaction. This is achieved by a logistic layer: $\hat{r}_{ui} = 1/(1 + \exp(-\boldsymbol{h}^T \boldsymbol{y}_{ui}))$, where $\boldsymbol{h}$ is a parameter vector. We adopt the binary cross-entropy loss: $\mathcal{L} = -\sum_{(u,i)\in\mathcal{S}} r_{ui} \log \hat{r}_{ui} + (1 - r_{ui}) \log(1 - \hat{r}_{ui})$, where $\mathcal{S} = \boldsymbol{R}^+ \cup \boldsymbol{R}^-$ are the union of observed interaction matrix and randomly sampled negative examples. This objective can be optimized by stochastic gradient descent (SGD) and its variants.

2.1 MemNet: Integrating Unstructured Text

To model the unstructured text, we use a memory network (MemNet) to attentively extract useful content to match the word semantics with specific user where different words in the text document have different weights in the semantic factor. Memory networks [19] are proposed to address the question answering (QA) task where memories are a short story and the query is a question related to the text in which the answer can be reasoned by the network. We can think of the recommendation with text as a QA: the question to be answered is to ask how likely a user prefers an item. And the unstructured text is analogue to the story and the query is analogue to the user-item interaction.

Memory networks have been used in recommendation to model item content [11, 12], model users' neighborhood [6], and learn latent relationships [20]. We use memory networks to attentively extract important information from the text content via the attention mechanism which can match word semantics with the specific user and determine which words are highly relevant to the user preferences. Recall that a memory network has one internal memory matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}| \times d}$ where d is the dimension of each memory slot. And another external memory matrix $\mathbf{C}$ is the same dimensions as $\mathbf{A}$. Given a doc $d_{ui} = [w_j]_{j=1}^{l}$ corresponding to the user-item (u, i) interaction, we form the memory slots $\boldsymbol{m}_j = [\boldsymbol{m}_j^{(u)}, \boldsymbol{m}_j^{(i)}] \in \mathbb{R}^d$ by mapping each word w_j into an embedding vector with matrix $\mathbf{A}$. We get a preference vector $\boldsymbol{a}^{u,i} = [a_j^{u,i}]$ corresponding to the document d_{ui} and the user-item interaction (u, i), where each element encodes the relevance of the user u to the word $w_j \in d_{ui}$ given item i:

$$a_j^{u,i} = \boldsymbol{x}_u^T \boldsymbol{m}_j^{(u)} + \boldsymbol{x}_i^T \boldsymbol{m}_j^{(i)}, \quad j = 1, ..., l. \tag{2}$$

On the right hand of the above equation, the first term captures the matching between preferences of user u and word semantics. The second term computes the support of item i to the words. Together, the content-based addressing scheme can determine internal memories with highly relevance to the specific user u regarding the words d_{ui} given item i. We then normalize it by the softmax function: $p_j^{u,i} = \text{softmax}(a_j^{u,i})$, to produce a probability distribution over the words in d_{ui}. The neural attention mechanism allows the MemNet to focus on specific words while to place little importance on other words which may be less relevant. We construct the high-level semantic factor representations by interpolating the external memories with the attentive weights as the output of the memory module: $\boldsymbol{o}_{ui} = \sum_j p_j^{u,i} \boldsymbol{c}_j$, where external memory slot $\boldsymbol{c}_j \in \mathbb{R}^d$ is another embedding vector for word $w_j \in d_{ui}$ by mapping it with matrix $\mathbf{C}$.

3 Experiments

We evaluate on two real-world cross-domain datasets. The public **Amazon** dataset and the Cheetah **Mobile** dataset provided by an internet company (see Appendix 5.1 for details). We adopt the leave-

Table 1: Results on Amazon. We mark best baselines with asterisks(*) and best results boldfaced.

Amazon	Metric	Methods						Improve of CoMem vs.		
		BPRMF	HFT	TextBPR	MLP	LCMR	CoMem	MLP	TextBPR	LCMR
5	HR	.0810	.1077	.1517	.2100*	.2024	**.2352**	12.00%	55.04%	16.20%
	NDCG	.0583	.0815	.1208	.1486*	.1451	**.1646**	7.61%	28.87%	9.63%
	MRR	.0509	.0729	.1104	.1283*	.1263	**.1413**	6.19%	20.36%	7.41 %
10	HR	.1204	.1360	.1777	.2836*	.2836*	**.3186**	12.34%	79.29%	12.34%
	NDCG	.0710	.0907	.1291	.1697*	.1678	**.1915**	7.68%	35.11%	8.35%
	MRR	.0561	.0767	.1138	.1371*	.1356	**.1524**	5.39%	21.72%	5.92%
20	HR	.1821	.2782	.2268	.3820	.3951*	**.4221**	10.49%	86.11%	6.83%
	NDCG	.0864	.1252	.1414	.1899	.1918*	**.2175**	7.22%	33.55%	6.50%
	MRR	.0602	.0854	.1171	.1426*	.1420	**.1595**	4.42%	18.69%	4.42%

one-out (LOO) evaluation [9] and use three ranking metrics: hit ratio (HR), normalized discounted cumulative gain (NDCG), and mean reciprocal rank (MRR) (see Appendix 5.2). Our methods are implemented using TensorFlow (see Appendix 5.3). We compare with five baselines (see Appendix 5.4) as summarized in the following table:

Baselines	Shallow method	Deep method
Pure CF	BPRMF [17]	MLP [9]
Hybrid	HFT [15], TextBPR [8, 10]	LCMR [11], CoMem (ours)

3.1 Comparisons of Different Recommender Systems

Results are shown in Table 1 (and Table 3, see App. 5.6). We have some observations. First, CoMem outperforms the neural CF method MLP on two datasets in terms of three ranking metrics. On Amazon, CoMem obtains a large improvement in performance gain with relative 12.34% HitRatio@10, 7.68% NDCG@10, and 6.19% MRR@5. On Mobile, CoMem obtains a large improvement in performance gain with relative 4.98% HitRatio@5, and 4.16% NDCG@5, and 3.88% MRR@5. Since the CFNet component of CoMem is a neural CF method, results show the benefit of exploiting unstructured text to alleviate the data sparsity issue faced by the pure CF methods BPRMF and MLP.

Second, CoMem also outperforms the traditional hybrid methods HFT and TextBPR on the two datasets in terms of three ranking metrics. On the Amazon dataset, CoMem obtains a significantly large improvement in performance gain with relative 55.04% HitRatio@5, 28.87% NDCG@5, and 20.36%MRR@5. On the Mobile dataset, CoMem still obtains reasonably large improvements with relative 17.52% HitRatio@10, 1.81% NDCG@10, and 1.88%MRR@10. Compared with traditional hybrid methods which integrate the text using topic modelling or word embeddings, the results show the benefit of integrating text information through memory networks (and exploiting the interaction data through neural CF). In detail, the TextBPR extracts the text feature $\boldsymbol{f_i}$ by averaging the word embeddings $\boldsymbol{e}_w$ in the document: $\boldsymbol{f_i} = \frac{1}{|d_{ui}|}\sum_{w \in d_{ui}} \boldsymbol{e}_w$. We can see that it treats different words in the document as equal importance and does not match word semantics with the specific user.

Last, CoMem outperforms the neural hybrid method LCMR by a large margin on the Amazon dataset with relative improvements of 16.20% HitRatio@5, 9.63% NDCG, and 7.41%MRR@5. CoMem still obtains reasonabe improvements on the Mobile dataset with relative improvements of 3.14% HitRatio@5, 2.85% NDCG, and 2.72% MRR. As we have revealed the relationships between CoMem and LCMR in the theorem 1 (see Appendix 5.5), the design of CFNet of CoMem is more reasonable than that of centralized memory module of LCMR. The results show the effectiveness of CoMem to exploit unstructured text via the MemNet and the interaction data via the CFNet.

Analysis We first evaluate the effects of the dimensionality of the embedding space. The x-axis in Figure 1b is the dimension of user/item and hence the dimensionality of input to CFNet and MemNet is double since we adopt concatenation. It clearly indicates that the embedding should not be too small due to the possibility of information loss and the limits of expressiveness. We next show optimization curves of performance@10 and loss (averaged over all examples) against iterations on the Mobile in Figure 1c. The model learns quickly in the first 20 iterations and improves slowly until 50, though training losses continue to go down and valid losses stabilize. The average time per epoch of CoMem takes 68.1s and as a reference it is 34.5s for MLP using one NVIDIA TITAN Xp GPU.

4 Conclusion

It is shown that relational interactions can be effectively integrated with text content under a neural embedding architecture to help improve recommendation performance. The proposed CoMem model can attentively focus relevant words to match user preferences (semantic factor) and model nonlinear relationships between users and items (behavior factor). In practice, CoMem shows better performance than five baselines on two real-world datasets in terms of three ranking metrics.

References

[1] Trapit Bansal, David Belanger, and Andrew McCallum. Ask the gru: Multi-task learning for deep text recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 107–114. ACM, 2016.

[2] Yang Bao, Hui Fang, and Jie Zhang. Topicmf: Simultaneously exploiting ratings and reviews for recommendation. In *AAAI*, volume 14, pages 2–8, 2014.

[3] Rose Catherine and William Cohen. Transnets: Learning to transform for recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, pages 288–296. ACM, 2017.

[4] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pages 7–10. ACM, 2016.

[5] Gintare Karolina Dziugaite and Daniel M Roy. Neural network matrix factorization. *arXiv preprint arXiv:1511.06443*, 2015.

[6] Travis Ebesu, Bin Shen, and Yi Fang. Collaborative memory network for recommendation systems. *SIGIR*, 2018.

[7] Gayatree Ganu, Noemie Elhadad, and Amélie Marian. Beyond the stars: improving rating predictions using review text content. In *WebDB*, 2009.

[8] Ruining He and Julian McAuley. Vbpr: Visual bayesian personalized ranking from implicit feedback. In *AAAI*, pages 144–150, 2016.

[9] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, pages 173–182. International World Wide Web Conferences Steering Committee, 2017.

[10] Guang-Neng Hu and Xin-Yu Dai. Integrating reviews into personalized ranking for cold start recommendation. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 708–720. Springer, 2017.

[11] Guangneng Hu, Yu Zhang, and Qiang Yang. Lcmr: Local and centralized memories for collaborative filtering with unstructured text. *arXiv preprint arXiv:1804.06201*, 2018.

[12] Haoran Huang, Qi Zhang, Xuanjing Huang, et al. Mention recommendation for twitter with end-to-end memory network. In *Proc. IJCAI*, volume 17, pages 1872–1878, 2017.

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

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

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

[16] Andriy Mnih and Ruslan R Salakhutdinov. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pages 1257–1264, 2008.

[17] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 452–461. AUAI Press, 2009.

[18] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

[19] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448, 2015.

[20] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. Latent relational metric learning via memory-based attention for collaborative ranking. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 729–739. International World Wide Web Conferences Steering Committee, 2018.

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

[22] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1235–1244. ACM, 2015.

[23] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 353–362. ACM, 2016.

[24] Lei Zheng, Vahid Noroozi, and Philip S Yu. 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*, pages 425–434. ACM, 2017.

5 Appendix

5.1 Appendix: Dataset

We evaluate on two real-world cross-domain datasets. The public **Amazon** (`http://snap.stanford.edu/data/web-Amazon.html`) dataset has been widely used to evaluate the performance of collaborative filtering approaches [15, 8]. We use the category of Amazon Men. The original ratings are from 1 to 5 where five stars indicate that the user shows a positive preference on the item while the one stars are not. We convert the ratings of 4-5 as positive samples. The dataset we used contains 56K positive ratings on Amazon Men. There are 8.5K users and 28K Men products. We aim to improve the recommendation with the product reviews. The data sparsity is over 99.7%. We filter stop words and use tf-idf to choose the top 8,000 distinct words as the vocabulary [21]. The average number of words per review is 32.9. The second dataset, **Mobile**, is provided by a large internet company, i.e., Cheetah Mobile (`http://www.cmcm.com/en-us/`). The information contains logs of user reading news, the history of app installation, and some metadata such as news publisher and user gender collected in one month in the US. We removed users with fewer than 10 feedbacks. For each item, we use the news title as its text content. We filter stop words and use tf-idf to choose the top 8,000 distinct words as the vocabulary. This yields a corpus of 612K words. The average number of words per news is less than 10. The dataset we used contains 477K user-news reading records. There are 15.8K users and 84K news. We aim to improve the news recommendation by exploiting news titles. The data sparsity is over 99.6%. The statistics are summarized in Table 2.

5.2 Appendix: Evaluation protocols

For item recommendation task, the leave-one-out (LOO) evaluation is widely used and we follow the protocol in [9]. That is, we reserve one interaction as the test item for each user. We determine hyper-parameters by randomly sampling another interaction per user as the validation set. We follow the common strategy which randomly samples 99 (negative) items that are not interacted by the user and then evaluate how well the recommender can rank the test item against these negative ones.

Since we aim at top-K item recommendation, the typical evaluation metrics are hit ratio (HR), normalized discounted cumulative gain (NDCG), and mean reciprocal rank (MRR), where the ranked list is cut off at $topK = \{5, 10, 20\}$. HR intuitively measures whether the reserved test item is present on the top-K list, defined as: $HR = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \delta(p_u \leq topK)$, where p_u is the hit position for the test item of user u, and $\delta(\cdot)$ is the indicator function. NDCG and MRR also account for the rank of the hit position, respectively defined as: $NDCG = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{\log 2}{\log(p_u+1)}$, and $MRR = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{1}{p_u}$. NDCG provides a good discriminative power. A higher value with lower cutoff indicates better performance.

Table 2: Datasets and Statistics.

Dataset	Amazon	Mobile
#Users	8,514	15,890
#Items	28,262	84,802
#Feedback	56,050	477,685
#Words	1,845,387	612,839
Rating Density (%)	0.023%	0.035%
Avg. Words per Item	65.3	7.2

5.3 Appendix: Implementation details

For BPRMF, we use LightFM's implementation which is a popular CF library. For HFT and TextBPR, we use the code released by their authors. The word embeddings used in the TextBPR are pre-trained by GloVe. For latent factor models, we vary the number of factors from 10 to 100 with step size 10. For MLP, we use the code released by its authors. The LCMR model is similar to our CoMem model and thus implemented in company. Our methods are implemented using TensorFlow. Parameters are randomly initialized from Gaussian $\mathcal{N}(0, 0.01^2)$. The optimizer is Adam with initial learning rate 0.001. The size of mini batch is 128. The ratio of negative sampling is 1. The MLP follows a tower pattern, halving the layer size for each successive higher layer. Specifically, the configuration of hidden layers in the base MLP network is $[64 \rightarrow 32 \rightarrow 16 \rightarrow 8]$ as reference in the original paper. The dimension of embeddings of users and items is both default 75 (and hence $d = 150$).

5.4 Appendix: Baselines

BPRMF, Bayesian personalized ranking [17], is a state-of-the-art latent factor model based on matrix factorization and pair-wise loss. It learns on the target domain only. **HFT**, Hidden factors and hidden Topics [15], adopts topic distributions to learn latent factors from text reviews. HFT exploits text content through topic modelling techniques. It is a hybrid method. **TextBPR** extends the basic BPRMF model by integrating text content. TextBPR learns text factors from the text features which are pre-extracted using word embeddings. It has two implementations, the VBPR model [8][1] and the TBPR model [10] which are the same in essence in terms of modeling equations. **MLP**, multilayer perceptron [9], is a neural CF approach which learns the nonlinear interaction function using neural networks. It is a deep model learning on the target domain only. **LCMR**, Local and Centralized Memory Recommender [11], is a deep model for collaborative filtering with unstructured Text. Its local memory module is similar to our MemNet. Its centralized memory module is inferior to our CFNet as we have analyzed in Theorem 1 and will be demonstrated in the experiments. This is a deep hybrid method.

5.5 Appendix: Connections to Existing Approaches

We reveal the relations between the proposed model and two existing approaches including matrix factorization (MF) and a neural hybrid filtering method (LCMR).

We firstly show that the CFNet of CoMem generalizes matrix factorization.

Let $\boldsymbol{P}^T\boldsymbol{x}_u$ be that of latent user factors $\underline{\boldsymbol{x}_u}$ in MF, and $\boldsymbol{Q}^T\boldsymbol{x}_i$ be that of latent item factors $\underline{\boldsymbol{x}_i}$ in MF. MF computes the predicted score by $\hat{r}_{ui} = \underline{\boldsymbol{x}_u}^T\underline{\boldsymbol{x}_i}$. We replace the concatenation in the embedding module with element-wise multiplication:

$$\boldsymbol{x}_{ui} = (\boldsymbol{P}^T\boldsymbol{x}_u) \odot (\boldsymbol{Q}^T\boldsymbol{x}_i). \tag{3}$$

It requires that the dimensions of latent features of users and items are the same in this case (denoted as d). We have one fixed weight $\boldsymbol{W} \equiv \mathbf{1}$ which is a d-dimensional vector of all 1-s and the bias $\boldsymbol{b} \equiv \mathbf{0}$ is all 0-s. And we do not perform nonlinear ReLU activation and insted use the identity mapping. In this way, the prediction is the same whatever it is computed by MF or CFNet of CoMem.

This allows CoMem to have the nonlinear modelling power beyond the standard matrix factorization and mimic a class of factorization models, e.g., 2-way factorization machines. As we will see in the experiments, CoMem outperforms MF methods on real-world datasets.

We now show that the CoMem is a more reasonable architecture design than an existing neural hybrid filtering method.

Theorem 1 CoMem and LCMR [11] have the same function by replacing the activation ReLU with softmax in the CFNet.

Proof Since the input, embedding, MemNet, and output layers of both CoMem and LCMR are the same, we only need to show that the CFNet of CoMem have the same function with the centralized memory blocks in LCMR. Assume that there is only one hop centralized memory block in LCMR (If there are multi-hop, we can prove it by induction).

Denote the input as $\boldsymbol{x} \in \mathbb{R}^{d_1}$ and the output as $\boldsymbol{y} \in \mathbb{R}^{d_2}$. Our CFNet has two hidden layers $\mathcal{L}_1$ and $\mathcal{L}_2$, which are called a pair $\mathcal{P} = \{\mathcal{L}_1, \mathcal{L}_2\}$, and the corresponding connection matrices are $\boldsymbol{W}_1 \in \mathbb{R}^{N\times d_1}$ and $\boldsymbol{W}_2 \in \mathbb{R}^{d_2\times N}$ where N is the number of units in the first layer $\mathcal{L}_1$. Then the output of layer $\mathcal{L}_1$ is $\boldsymbol{o} = \boldsymbol{W}_1\boldsymbol{x}$. After a *softmax activation*, the input to layer $\mathcal{L}_2$ is $\boldsymbol{a} = \text{Softmax}(\boldsymbol{o})$ and the final output $\boldsymbol{y} = \boldsymbol{W}_2\boldsymbol{a}$.

For the LCMR, it has one building block $\mathcal{B} = \{\mathcal{A}, \mathcal{C}\}$ which consists of key and value memories $\mathcal{A}$ and $\mathcal{C}$, where dimensions $\boldsymbol{A} \in \mathbb{R}^{N\times d_1}$ and $\boldsymbol{C} \in \mathbb{R}^{N\times d_2}$ are compatibility with the input and output, respectively (where

[1] Replace the pre-extracted image features with the word embeddings as the input.

Table 3: Results on Mobile. Best baselines are marked with asterisks(*) and best results are boldfaced.

Mobile	Metric	Methods						Improve of CoMem vs.		
		BPRMF	HFT	TextBPR	MLP	LCMR	CoMem	MLP	TextBPR	LCMR
5	HR	.4380	.4966	.4948	.5380	.5476*	**.5648**	4.98%	14.14%	3.14%
	NDCG	.3971	.3617	.4298*	.4121	.4189	**.4345**	4.16%	0.95%	2.85%
	MRR	.3606	.3175	.3826*	.3702	.3762	**.3911**	3.88%	1.72%	2.72 %
10	HR	.4941	.5580	.5466	.6176	.6311*	**.6424**	4.02%	17.52%	1.79%
	NDCG	.4182	.4093	.4499*	.4381	.4460	**.4598**	3.51%	1.81%	2.19%
	MRR	.3694	.3365	.3913*	.3810	.3874	**.4016**	3.34%	1.88%	2.25%
20	HR	.5398	.6547	.6123	.6793	.6927*	**.6952**	2.34%	13.53%	0.36%
	NDCG	.4316	.4379	.4682*	.4529	.4619	**.4732**	2.99%	0.82%	1.63%
	MRR	.3730	.3445	.3958*	.3851	.3918	**.4053**	2.97%	1.55%	1.95%

the size of memories is N). The attention weights $\boldsymbol{a}$ are firstly computed by $\boldsymbol{o} = \boldsymbol{A}\boldsymbol{x}$ and then normalized by *softmax function* $\boldsymbol{a} = \text{Softmax}(\boldsymbol{o})$ to be a simplex vector. The output is a sum over the values weighted by attentive weights $\boldsymbol{y} = \boldsymbol{C}^T\boldsymbol{a}$. We can see that if we let $\boldsymbol{A} = \boldsymbol{W}_1$ and $\boldsymbol{C} = \boldsymbol{W}_2^T$, then the learning functions of CoMem and LCMR are the same. □

It seems unusual to use a softmax function mapping between two hidden layer connections. The values would be fairly small causing a vanishing gradient since it would be normalized to a probability distribution. A ReLU function would typically be used in this case. As a result, we can expect that the CoMem is a more reasonable architecture than LCMR, empirically demonstrated in the experiments.

5.6 Appendix: Results on Cheetah Mobile Dataset

Results on Cheetah Mobile dataset is shown in Table 3.