User Profiling based Deep Neural Network for Temporal News Recommendation

Vaibhav Kumar, Dhruv Khattar*, Shashank Gupta, Manish Gupta[1], Vasudeva Varma
Information Retrieval and Extraction Laboratory
International Institute of Information Technology Hyderabad
Hyderabad - 500032, Telangana, India
Email: {vaibhav.kumar, dhruv.khattar, shashank.gupta}@research.iiit.ac.in, {manish.gupta, vv}@iiit.ac.in

***Abstract*—One of the most important and challenging problems in recommendation systems is that of modeling temporal behavior. Typically, modeling temporal behavior increases the cost of parameter inference and estimation. Along with it, it also poses the constraint of requiring a large amount of data for reliably learning the parameters of the model. Therefore, it is often difficult to model temporal behavior in large-scale real-world recommendation systems.**

In this work, we propose a deep neural network architecture which is based on a two level approach. We first generate document embeddings for every news article. We then use these embeddings and the previously read articles by a user to come up with her user profile. We then use this profile along with adequate positive and negative samples in order to train our model. The resulting model is then applied to a real-world data set. We compare it with a set of established baselines and the experimental results show that our model outperforms the state-of-the-art. We also use the learned model to recommend articles to users who have had very little interaction with items, i.e., have read a very less amount of news articles. We then demonstrate the effectiveness of our model to solve the problem of item cold-start.

***Index Terms*—Deep Structured Semantic Model, User Profiling, News Recommendation**

I. Introduction

Recommendation systems are core components of many of the modern internet services including news, e-commerce, online movie sites and more. A major approach to the task of recommendation is called collaborative filtering [1] [2] [3] which uses the users past interaction with the item to predict the most relevant content. Another common approach is content-based recommendation, which uses features between items and/or users to recommend items to the users based on the similarity between its features. However, among the various approaches for collaborative filtering, matrix factorization [4] is the most popular one, which projects users and items into a shared latent space, using a vector of latent features to represent a user or an item. Thereafter, a users interaction with an item is modeled as the inner product of their latent vectors.

Collaborative filtering needs a considerable amount of previous history of interaction before it can provide high quality recommendations. This problem is typically known as the cold start problem. For a newly established news website, the problem would become even more severe since users have little or no history of interaction with the site. Traditional approaches fail to produce high quality recommendation in this case. However, in practice, it has been shown that content-based approach can handle cold start problem for new items well.

In the news domain, it becomes very crucial to account for the dynamic changes in users' interests as well as to come up with better recommendations. A solution for this could be the creation of a user profile which accounts for both the short term as well as the long term interests of the users. The content of the items plays a very important role in creating such a user profile, which many of the recent work do not account for.

Recently in [5], authors proposed a Neural Network architecture for collaborative filtering. They explore the use of deep neural network for learning the interaction function from the data. Their proposed method specifically aims to model the relationship between users and items.

In this work, we pose the recommendation problem as that of a binary classification problem. We use the user-item interaction and the content of the news in order to capture the similarity between users and items (news). We do not use any explicit information about the user in order to model her interests. We only focus on the implicit information provided by the user, i.e., whether a user read a given article or not.

The sequence in which the articles are read by the user encapsulates information about the interests of the user. In order to capture the preferences from the sequence and create a user profile from it, we need a mechanism which should be capable of handling long term dependencies. We use the following steps to achieve this.

1) First, we learn the doc2vec [22] embeddings for each news article.
2) We then choose a specific amount of reading history for all the users.
3) Finally we combine the doc2vec embeddings of each of the articles present in the user history using certain heuristics which preserves the temporal information encoded in the sequence of articles read by the user.

Then, in order to capture the similarity between users and items, we need to be able to project them to the same latent

*Corresponding Author
[1]The author is also an applied researcher at Microsoft.

space. We adapt Deep Structured Semantic Model (DSSM) [7] for this. DSSM was originally proposed for the task of web document ranking. Later, it was adapted for the task of recommendation in [8]. However, in [8] the features for the users are their search queries and features for items come from multiple domains (e.g Apps, Movies/TV etc.) which makes it difficult for a news website to directly adapt it as a lot of information outside the news domain is required. Then, for learning the parameters of the model we use a ranking based objective function. Finally, for recommending news articles to the users we use the computed inner product between user and item latent vectors.

To summarize, the contributions of our work are as follows.

1) We use doc2vec embeddings of each news article in order to come up with user profiles for each user which encapsulates information about the changing interests of the user over time.
2) We use a deep neural architecture for news recommendation in which we utilize the user-item interaction as well as the content of the news (items) to model the latent features of users and items.
3) We perform experiments to demonstrate the effectiveness of our model for the problem of news recommendation. We then perform experiments to show the effectiveness of our model when the user has had very little interaction with items.
4) We also address the effectiveness of our model in solving the item cold-start problem.

II. Related Work

In this section, we aim at reviewing a representative set of approaches that are related to our proposed approach.

A. Common Approaches for Recommendation

Recommendation systems in general can be divided into collaborative filtering based recommendation and content based recommendation. In a narrower sense, in collaborative filtering based recommendations, an item is recommended to a user if similar users liked that item. Collaborative filtering can be further divided into user collaborative filtering, item collaborative filtering or a hybrid of both user and item collaborative filtering. Examples of such techniques include Bayesian matrix factorization [9], matrix completion [2], Restricted Boltzmann Machine [3], nearest neighbor modeling [1] etc. Apart from this, in [28] [29] the authors have tried to come up with user profiling based techniques in order to provide better recommendations.

B. Neural Networks for News Recommendation

Early pioneer work which used neural network was done in [3], where a two-layer Restricted Boltzmann Machine (RBM) is used to model users' explicit ratings on items. The work has been later extended to model the ordinal nature of ratings [13]. Recently autoencoders have become a popular choice for building recommendation systems [10] [11] [12]. The idea of user-based AutoRec [11] is to learn hidden structures that can reconstruct a users ratings given her historical ratings as inputs. In terms of user personalization, this approach shares a similar spirit as the item-item model [15] [14] that represents a user using features related to her rated items. While previous work has lent support for addressing collaborative filtering, most of them have focused on observed ratings and modeled observed data only. As a result, they can easily fail to learn users' preferences accurately from the positive-only implicit data.

In [5], the authors present a general framework named NCF, short for Neural Collaborative Filtering that replaces the inner product (which calculates the similarity between a user and an item) with a neural architecture that can learn an arbitrary function from the given data. It uses a multi-layer perceptron to learn the user-item interaction function. NCF is able to express and generalize matrix factorization.

C. User-Item Projection

Since our work is based on user-item based collaborative filtering, we need to project users and items to a common latent space in order to capture their similarity and recommend items to users accordingly. One of the most effective approaches in projecting queries and documents into a common low-dimensional space has been shown in [7]. The model is named as Deep Semantic Structured Model (DSSM) [7] which is effective in calculating the relevance of the document given a query by computing the distance between them. Originally this model was meant for the purpose of ranking, but since the problem of ranking has very close associations with that of recommendation, DSSM was later extended to recommendation scenarios in [8]. In [8], the authors used DSSM for recommendation where the first neural network contains the users query history (and thus referred to as the user view) and the second neural network contains implicit feedback of items. The resulting model is named multi-view DNN (MV-DNN) since it can incorporate item information from more than one domain and then jointly optimize all of them using the same loss function as in DSSM. However, in [8], the features for the users were their search queries and features for items came from multiple sources (e.g Apps, Movies/TV etc.). This makes it less adaptable by a news website as it requires a lot of information outside the news domain.

III. Dataset

For this work we use the dataset published by CLEF NewsREEL 2017. CLEF NewsREEL provides an interaction platform to compare performance of different news recommender systems in an online as well as in an offline setting [16]. As a part of their evaluation for offline setting, CLEF shared a dataset which captures interactions between users and news stories. It includes interactions of eight different publishing sites in the month of February 2016. The recorded stream of events include 2 million notifications, 58 thousand item updates, and 168 million recommendation requests. The dataset also provides other information like the title and text of each news article, time of publication etc. Each user can be identified by a unique id. For our task, we needed to find out the sequence in which the articles were read by the users. Along with this we also find out the content of each of these read articles. Since we rely only on implicit feedback we only need to know whether the article was read by a user or not.

IV. Model Architecture

In this section we discuss briefly the components of our model. We first discuss user profiling, followed by the DSSM architecture. We then provide the training criteria for our model. Figure 1 illustrates the architecture of the proposed model.

A. User Profiling

We first define a set of notations useful in understanding the creation of user profile. We define the number of articles in the user reading history to be R. The doc2vec embeddings of each article in the history is represented by r_h where $1 \leq h \leq R$. Each vector is of size 300. The user profile for a user is denoted by U. We now discuss three kinds of operations using which we create the user profiles.

1) **Centroid**
   In this method, we find the centroid of the embeddings of the articles present in the reading history of the user. The centroid then represents the user profile.

$$U = \frac{1}{R} \sum_{h=1}^{R} r_h \tag{1}$$

2) **Discounting**
   In this we first discount each of the vectors present in the user reading history by a power of 2 such that an article read at time $t-1$ carries half the weight compared to an article read at time t. We then take an average of all the vectors.

$$U = \frac{1}{R} \sum_{h=1}^{R} \frac{r_h}{2^{R-h}} \tag{2}$$

3) **Exponential Discounting**
   In this we discount each of the vectors present in the user reading history by a power of e such that an article read at time $t-1$ carries $1/e$ the weight compared to an article read at time t. We then take an average of all the vectors.

$$U = \frac{1}{R} \sum_{h=1}^{R} \frac{r_h}{e^{R-h}} \tag{3}$$

Figure 2 plots the exponential discounting function as well as the function with a discounting factor of 2.

We experiment with different kinds of methods for creating a user profile in order to understand the temporal patterns present in the user reading history. The idea behind discounting and exponential discounting is to give more preference to the recently read articles and lesser preference to those read far away in the past.

B. Deep Structured Semantic Model

The Deep Semantic Structured Model (DSSM) [7] was proposed for the purpose of ranking. Essentially, DSSM can be viewed as a multi-view learning model that often composes of two or more neural networks for each individual view. In the original two-view DSSM model, the network on the left side was meant for query representation, whereas the networks on the right side were meant for representing the documents. The input to these networks could be of any arbitrary type like letter-tri-gram in the original paper or bag of unigrams used in [8]. Each input vector after that goes through a non-linear transformation in the feed-forward neural network to output an embedding vector, which is smaller in size than the input vector. The learning objective of the DSSM is to maximize the cosine similarity between the two output vectors. For the purpose of training, a set of positive examples and randomly sampled negative examples are generated in order to minimize the cosine similarity based loss.

C. Modified DSSM

In this work, we modify the DSSM in the following ways.

1) Instead of using letter-tri-gram, we use the doc2vec embeddings of each of the news articles as input.
2) The input to the left side of the model is the user reading history R. The doc2vec embeddings of each of the article present in the user reading history is passed as inputs. A user profile is then computed using these embeddings.
3) The input to the right side of the model consists of 1 positive instance (an article read by the user apart from the articles already present in the user history) and n negative articles. The n negative articles are randomly sampled.

D. Learning

Typically in matrix factorization, to learn the model parameters, existing pointwise methods [17] [18] perform regression with a squared loss. This is based on the assumption that observations are generated from a Gaussian distribution. However, in [8] it has been shown that such a method does not tally well when we have implicit data available to us. Also, in [19] it has been shown that a ranking based objective function is more suitable for the task of recommendation. Keeping these two aspects in mind, we adapt the loss function used in DSSM [20]. We first compute the posterior probability of a clicked news item given a user from the relevance score using a softmax function as follows.

$$P(item^+|u) = \frac{exp(R(u, item^+))}{\sum_{\forall item} exp(R(u, item))} \tag{4}$$

where $item^+$ denotes the item that was clicked by the user and $R()$ represents the inner product function. We then maximize the likelihood of the clicked news items given the user with the following loss function.

$$L(\Lambda) = -\log \prod_{u, item^+} P(item^+|u) \tag{5}$$

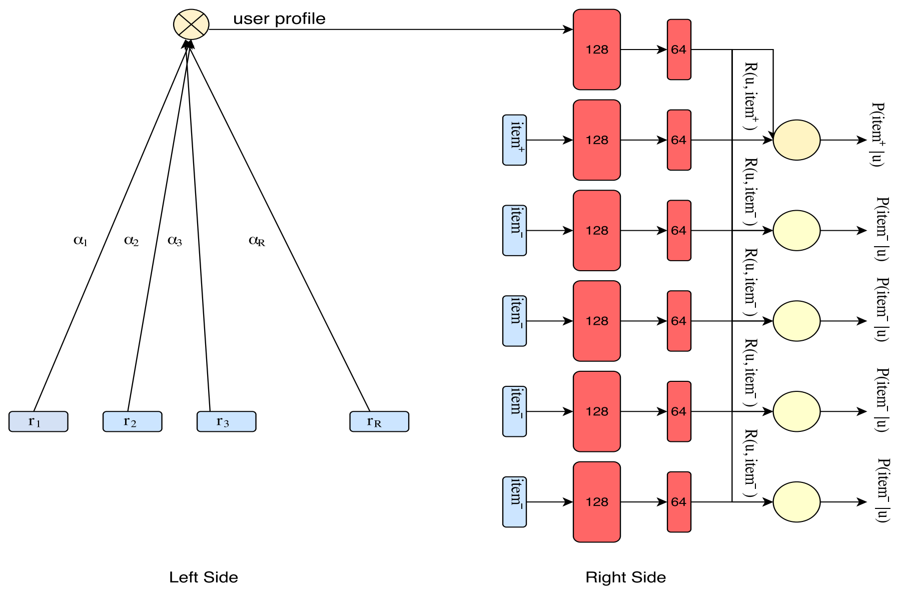

Fig. 1: Model Architecture for the Modified DSSM. $\otimes$ represents the operation using which the user profile is created. The left side of the model creates the user profile, while the right side of the model provides positive and negative instances for training the model parameters. $r_1 \dots r_R$ represent articles present in the user history. $item^+$ and $item^-$ represents the positive and negative instances respectively.

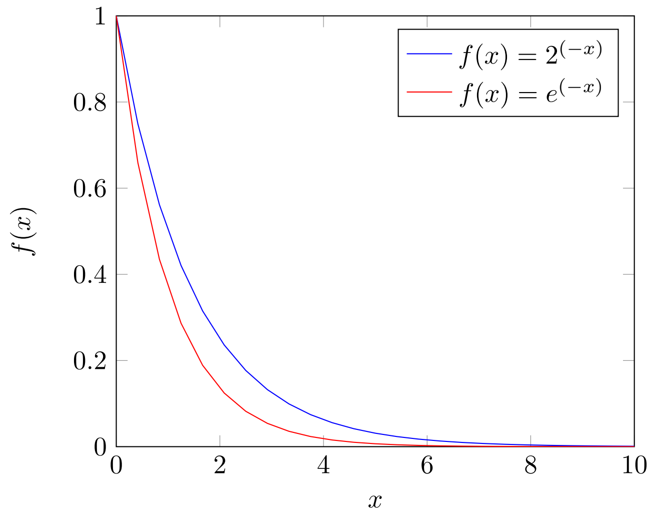

Fig. 2: Graph for the discounting functions used

where, Λ represents the parameters of our model.

V. Experiments

In this section we delineate our conducted experiments in order to answer the following questions.

1) How do the different profiling methods help in improving the overall recommendation?
2) How does our model perform against state-of-the-art methods?
3) How well does our model perform when the user has not had many interactions with the items, i.e., for users who have read very less number of articles?
4) How well does our model perform in recommending items which have not had any interaction by any user (item cold start problem)?

A. Experimental Settings

As mentioned earlier, we use the dataset provided by CLEF NewsREEL 2017. We extract the sequence in which the articles were read by the users. For each article we concatenate the body and the text and use gensim [21] to learn doc2vec [22] embeddings for those. The size of the embeddings is set to 300. In the given dataset, almost 77% of the users have read less than 3 articles. We choose users who have read in between 10–15 (both inclusive) articles for training and testing our model for item recommendation. The frequency of users who have read more than 15 articles varies extensively and hence we restrict ourselves to the upper bound of 15. We then choose users who have read 2–4 articles for testing our model for the case when the user has had very little interaction with the items (user cold start problem). For the item cold start problem, we test it on users who have read in between 10–15 articles.

1) **Evaluation Protocol** To evaluate the performance of the recommended item we use the leave-one-out evaluation strategy which has been widely adopted in literature [23] [19] [24]. For each user we held-out her latest interaction as the test set and utilized the remaining data for training. Since it is time consuming to rank

K	**Avg**		**Discounting**		**Exponential Discounting**	
	HR@K	NDCG@K	HR@K	NDCG@K	HR@K	NDCG@K
1	0.447	0.447	0.474	0.474	0.445	0.445
2	0.635	0.566	0.644	0.581	0.628	0.561
3	0.710	0.603	0.711	0.615	0.703	0.598
4	0.744	0.618	0.744	0.629	0.740	0.614
5	0.769	0.627	0.767	0.638	0.764	0.623
6	0.784	0.633	0.785	0.644	0.780	0.629
7	0.798	0.637	0.798	0.648	0.793	0.633
8	0.810	0.641	0.810	0.652	0.806	0.637
9	0.819	0.644	0.821	0.655	0.816	0.640
10	0.830	0.647	0.830	0.658	0.826	0.643

TABLE I: Performance of our model using different user profiling operations

all items for every user during evaluation, we followed the common strategy [8] [4] that randomly samples 100 items that the user has not interacted with, and then ranking the test item among the 100 items. The performance of a ranked list is judged by two metrics: Hit Ratio (HR) and Normalized Discounted Cumulative gain (NDCG) [20]. We truncated the rank list at 10 for both metrics. As such, the HR@k intuitively measures whether the test item is present in the top-k list, and the NDCG accounts for the position of the hit by assigning higher scores to hits at top ranks. We calculated both metrics for each test user and reported the average score.

2) **Baselines**
   - **BPR** [24]. This method optimizes the matrix factorization method with a pairwise ranking loss, which is tailored to learn from implicit feedback. We report the best performance obtained by varying the learning rate.
   - **eALS** [19]. This is a state-of-the-art matrix factorization method for item recommendation. It optimizes the squared loss (between actual item ratings and predicted ratings) and treats all unobserved interactions as negative instances and weighting them non-uniformly by item popularity.
   - **NeuMF** [5]. This is a state-of-the-art neural matrix factorization model. It treats the problem of generating recommendations using implicit feedback as a binary classification problem. Consequently it uses the binary cross-entropy loss to optimize its model parameters.

   Our proposed method is based on modeling user-item relationship, hence we mainly compare it with other user-item models only. We leave out the comparison with other models like SLIM [15] and CDAE [25] because these are item-item models and hence performance difference may be caused by the user models for personalization.

3) **Parameter Settings** We implemented our proposed method using Keras [27]. As mentioned earlier, for each user who had read in between 10-15 (both inclusive) articles we held out the last read article for our test set. We then construct our labeled set as follows.
   a) We first define the reading history that we want to use. We denote the reading history by R.
   b) For each user, we use R number of read articles as inputs to the left side of the model. Leaving the last read article out, the remaining articles are used as positive samples for the right view of the model.
   c) For each positive instance of a user, we randomly sample n negative instances (news items that the user has not interacted with) which are used as inputs for the item view of the model. We experimentally set the number of negative instances n to be 4.

   We then randomly divide the labeled set into training and validation set in a 4:1 ratio. This helps us to ensure that the two sets do not overlap. We tuned the hyperparameters of our model using the validation set. The model and all its variants are learned by optimizing the log loss of Equation 5. We initialize the fully connected network weights with the uniform distribution in the range between $-\sqrt{6/(fanin + fanout)}$ and $\sqrt{6/(fanin + fanout)}$ [17] . We used a batch size of 256 and used adadelta [26] as a gradient based optimizer for learning the parameters of the model.

B. *Performance Comparison*

From Table 1 we can see the performance of our model when using three different kinds of methods for user profiling. We observe that the Discounting based method for user profiling had better results in terms of HR as well as NDCG in most of the cases. One of the main reasons for this could be that, since discounting gives more weight to the recently read articles, it adapts to the user's temporal changes in interests in a better fashion.

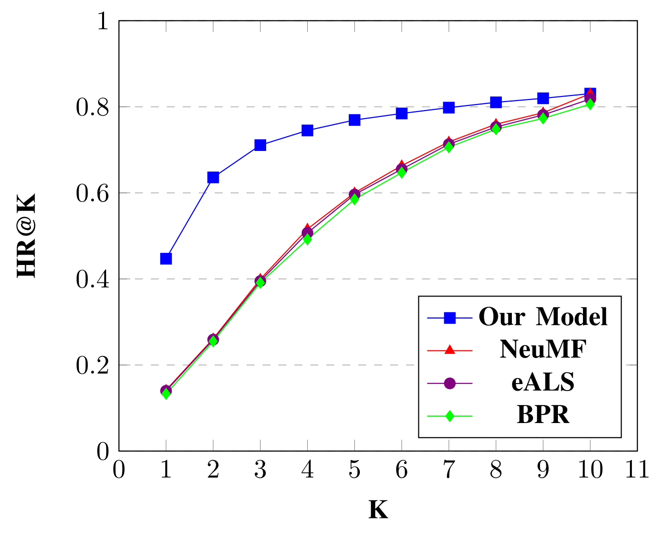

Fig. 3: HR@K performance of our model vs some state-of-the-art models

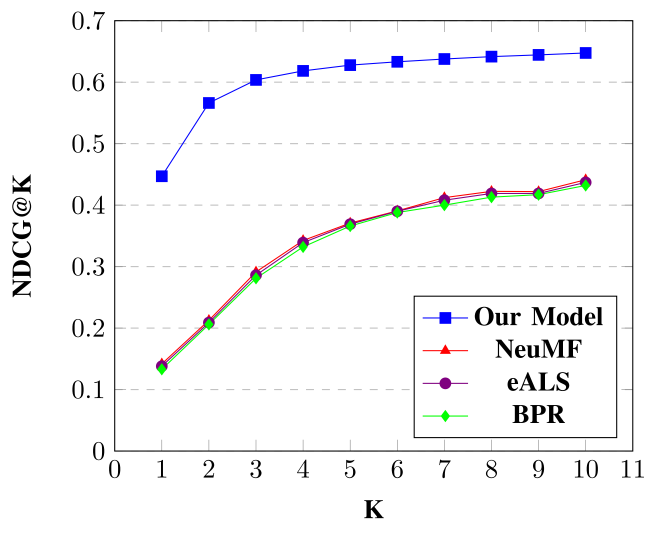

Fig. 4: NDCG@K performance of our model vs some state-of-the-art models

Figure 3 and Figure 4 show the performance of the Top-K recommended lists where the ranking position K ranges from 1 to 10. We leave out the variants of our own model here for comparison and only use the best performing model, i.e., the discounting based model. From the figures, it can be clearly seen that our model shows consistent improvements over the other methods across all positions. The reason for this can be attributed to the fact that apart from accounting for the user's general preferences we also account for the users changing interests and the extent of those interests which the baselines do not incorporate directly. Although, our model shows significant improvements in terms of HR when K varies from 1 to 5, slowly the performance of the baselines become similar to that of our proposed method when K is 10.

We observe major improvements in the NDCG scores of our model. There is an approximate 20% improvement over NeuMF. The reason for this is the loss function of Equation 5 used by our model. The loss function which is optimized for ranking, helps the model to recommend a better ranked list of items. There is however no significant difference between the performance of the baselines.

We then evaluated our model for the cold start cases. The results are shown in Figure 5 and Figure 6. For this task we segregated users who had read a new article in the end, i.e., they read articles which had never been seen before they read it. We found out that the number of such users were 74. Out of these 74 users, at an HR@10 we observe that around 33% of the time we were able to recommend that article. This promises us that our model is well suitable for handling the item cold-start problem. For user cold-start, we test our learned model over users who had read articles in between 2 to 4 (both inclusive). The HR@10 score was around 47%. We see a gradual increase in the hit rates as we increase the value of k. The results promise the efficiency of our model to handle the problem of user cold start as well.

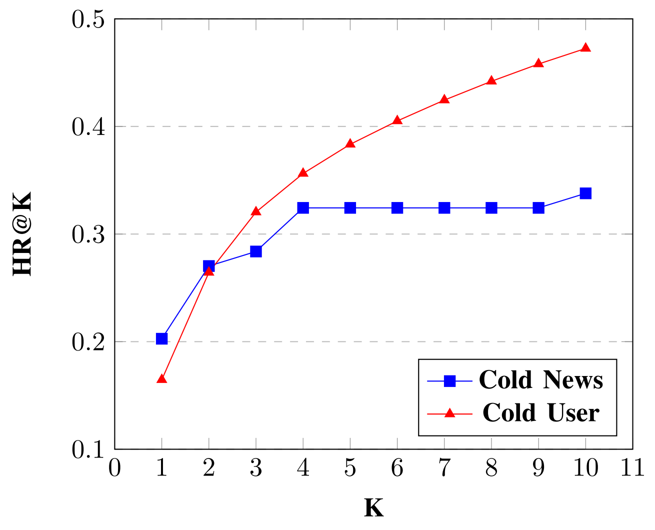

Fig. 5: HR@K of our model on Cold-Start cases

VI. Conclusion and Future Work

In this work, we tackle the problem of changing users' interests by first coming up with a user profile and then using it to learn the parameters of DSSM. This method can be considered similar to that of user-item collaborative filtering, the only difference being that, we account for the content of the items as well. The content-embeddings help us in generalizing the user preferences. We then also show the effectiveness of our model in recommending articles to users who have a very small reading history. The success of this prompts us towards the observation that content is very relevant in modeling user preferences even when the user has a very low reading history. Next, we also show that our model is very effective in solving the item cold-start problem as well.

For future work, we would like to experiment with neural

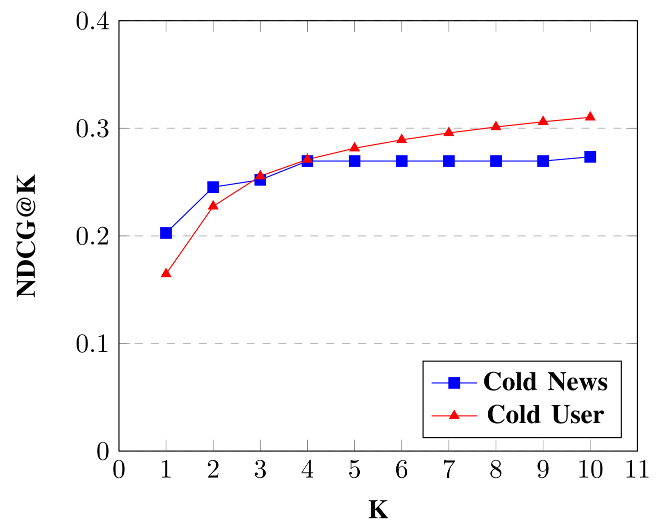

Fig. 6: NDCG@K of our model on Cold-Start cases

networks which are able to dynamically capture the temporal changes in user behaviour. This would probably help us to come up with better user profiles and help make the performance of the model even better.

References

[1] Bell, Robert M., and Yehuda Koren. "Improved neighborhood-based collaborative filtering." KDD cup and workshop at the 13th ACM SIGKDD international conference on knowledge discovery and data mining. sn, 2007.

[2] Rennie, Jasson DM, and Nathan Srebro. "Fast maximum margin matrix factorization for collaborative prediction." Proceedings of the 22nd international conference on Machine learning. ACM, 2005.

[3] Salakhutdinov, Ruslan, Andriy Mnih, and Geoffrey Hinton. "Restricted Boltzmann machines for collaborative filtering." Proceedings of the 24th international conference on Machine learning. ACM, 2007.

[4] Koren, Yehuda. "Factorization meets the neighborhood: a multifaceted collaborative filtering model." Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2008.

[5] He, Xiangnan, et al. "Neural collaborative filtering." Proceedings of the 26th International Conference on World Wide Web. International World Wide Web Conferences Steering Committee, 2017.

[6] Le, Quoc, and Tomas Mikolov. "Distributed representations of sentences and documents." Proceedings of the 31st International Conference on Machine Learning (ICML-14). 2014.

[7] Huang, Po-Sen, et al. "Learning deep structured semantic models for web search using clickthrough data." Proceedings of the 22nd ACM international conference on Conference on information & knowledge management. ACM, 2013.

[8] Elkahky, Ali Mamdouh, Yang Song, and Xiaodong He. "A multi-view deep learning approach for cross domain user modeling in recommendation systems." Proceedings of the 24th International Conference on World Wide Web. International World Wide Web Conferences Steering Committee, 2015.

[9] Salakhutdinov, Ruslan, and Andriy Mnih. "Bayesian probabilistic matrix factorization using Markov chain Monte Carlo." Proceedings of the 25th international conference on Machine learning. ACM, 2008.

[10] Chen, Minmin, et al. "Marginalized denoising autoencoders for domain adaptation." arXiv preprint arXiv:1206.4683 (2012).

[11] Sedhain, Suvash, et al. "Autorec: Autoencoders meet collaborative filtering." Proceedings of the 24th International Conference on World Wide Web. ACM, 2015.

[12] Strub, Florian, and Jrmie Mary. "Collaborative filtering with stacked denoising autoencoders and sparse inputs." NIPS workshop on machine learning for eCommerce. 2015.

[13] Phung, Dinh Q., and Svetha Venkatesh. "Ordinal Boltzmann machines for collaborative filtering." Proceedings of the Twenty-fifth Conference on Uncertainty in Artificial Intelligence. AUAI Press, 2009.

[14] Sarwar, Badrul, et al. "Item-based collaborative filtering recommendation algorithms." Proceedings of the 10th international conference on World Wide Web. ACM, 2001.

[15] Ning, Xia, and George Karypis. "Slim: Sparse linear methods for top-n recommender systems." Data Mining (ICDM), 2011 IEEE 11th International Conference on. IEEE, 2011.

[16] Hopfgartner, Frank, et al. "Benchmarking news recommendations: The clef newsreel use case." ACM SIGIR Forum. Vol. 49. No. 2. ACM, 2016.

[17] Glorot, Xavier, and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. 2010.

[18] Liu, Jiahui, Peter Dolan, and Elin Rnby Pedersen. "Personalized news recommendation based on click behavior." Proceedings of the 15th international conference on Intelligent user interfaces. ACM, 2010.

[19] He, Xiangnan, et al. "Fast matrix factorization for online recommendation with implicit feedback." Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval. ACM, 2016.

[20] He, Xiangnan, et al. "Trirank: Review-aware explainable recommendation by modeling aspects." Proceedings of the 24th ACM International on Conference on Information and Knowledge Management. ACM, 2015.

[21] Rehurek, Radim, and Petr Sojka. "Software framework for topic modelling with large corpora." In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks. 2010.

[22] Le, Quoc, and Tomas Mikolov. "Distributed representations of sentences and documents." Proceedings of the 31st International Conference on Machine Learning (ICML-14). 2014.

[23] Bayer, Immanuel, et al. "A generic coordinate descent framework for learning from implicit feedback." Proceedings of the 26th International Conference on World Wide Web. International World Wide Web Conferences Steering Committee, 2017.

[24] Rendle, Steffen, et al. "BPR: Bayesian personalized ranking from implicit feedback." Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence. AUAI Press, 2009.

[25] Wu, Yao, et al. "Collaborative denoising auto-encoders for top-n recommender systems." Proceedings of the Ninth ACM International Conference on Web Search and Data Mining. ACM, 2016.

[26] Zeiler, Matthew D. "ADADELTA: an adaptive learning rate method." arXiv preprint arXiv:1212.5701 (2012).

[27] Chollet, Franois. "Keras." (2015).

[28] Saia, Roberto, Ludovico Boratto, and Salvatore Carta. "Semantic Coherence-based User Profile Modeling in the Recommender Systems Context." KDIR. 2014.

[29] Saia, Roberto, Ludovico Boratto, and Salvatore Carta. "A semantic approach to remove incoherent items from a user profile and improve the accuracy of a recommender system." Journal of Intelligent Information Systems 47.1 (2016): 111-134.