Deep Neural Architecture for News Recommendation

Vaibhav Kumar, Dhruv Khattar*, Shashank Gupta, Manish Gupta, and Vasudeva Varma

International Institute of Information Technology Hyderabad, Gachibowli, Telangana - 500032, India
{vaibhav.kumar,dhruv.khattar,shashank.gupta}@research.iiit.ac.in
{manish.gupta,vv}@iiit.ac.in
https://www.iiit.ac.in

Abstract. Deep neural networks have yielded immense success in speech recognition, computer vision and natural language processing. However, the exploration of deep neural networks for recommender systems has received a relatively little introspection. Also, different recommendation scenarios have their own issues which creates the need for different approaches for recommendation. Specifically in news recommendation a major problem is that of varying user interests. In this work, we use deep neural networks with attention to tackle the problem of news recommendation.

The key factor in user-item based collaborative filtering is to identify the interaction between user and item features. Matrix factorization is one of the most common approaches for identifying this interaction. It maps both the users and the items into a joint latent factor space such that user-item interactions in that space can be modeled as inner products in that space. Some recent work has used deep neural networks with the motive to learn an arbitrary function instead of the inner product that is used for capturing the user-item interaction. However, directly adapting it for the news domain does not seem to be very suitable. This is because of the dynamic nature of news readership where the interests of the users keep changing with time. Hence, it becomes challenging for recommendation systems to model both user preferences as well as account for the interests which keep changing over time.

We present a deep neural model, where a non-linear mapping of users and item features are learnt first. For learning a non-linear mapping for the users we use an attention-based recurrent layer in combination with fully connected layers. For learning the mappings for the items we use only fully connected layers. We then use a ranking based objective function to learn the parameters of the network. We also use the content of the news articles as features for our model. Extensive experiments on a real-world dataset show a significant improvement of our proposed model over the state-of-the-art by 4.7% (Hit Ratio@10). Along with this, we also show the effectiveness of our model to handle the user cold-start and item cold-start problems.

* Vaibhav Kumar and Dhruv Khattar are the corresponding authors

Keywords: Deep Neural Networks, News Recommendation

1 INTRODUCTION

The web provides instant access to a wide variety of online news. Hence, it becomes desirable to have a recommender system that would point a user to the most relevant items and thus would maximize the user engagement with the site and minimize the time for finding relevant content. With the advent of deep learning, although recommender systems have been used with good success for products like movies and books, but have surprisingly found very little attention to the problem of news recommendation.

A major approach to the task of recommendation is called collaborative filtering [5][3][4] which uses the user's 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 new items to the users based on the similarity between features. However, amongst the various approaches for collaborative filtering, matrix factorization [11] 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 user's 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 recommendation. This problem is known as the typical 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.

Each recommendation scenario has its own issues which creates the need for different approaches for building recommendation systems. For example, news recommendation may put more focus on the freshness of the content while other systems like that of movie recommendation may emphasize more on content relatedness. Adding to this, specifically in the case of news, user interests keep evolving/changing over time. It might be possible that a user who reads news articles only pertaining to politics may suddenly develop interest in sports due to various reasons. Hence, it becomes very crucial to account for the dynamic changes in interests as well as come up with better recommendation. While many existing techniques assume the user interest to be static, this assumption seems a bit unrealistic. This suggests the need to handle temporal changes in the interests of the users.

Recently in [1], 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 come up with a hybrid approach that uses user-item interactions and the content of the news to capture the similarity between users and items (news). We only focus on implicit feedback (clicks and impressions) provided by the users, i.e whether they have read a given article or not and in what sequence were those articles read by them.

The sequence in which the articles are read by the user encapsulates information about the interests of the user. Capturing the interests of the user from the sequence of read articles requires a component which should be capable of learning long-term dependencies. LSTMs in general have shown to be suitable for this particular task [29][30]. To capture both static and dynamic interests which the user has developed over time, we use bidirectional LSTMs [31]. We chose a specific amount of reading history of each user as input to the LSTMs. Once these interests are captured, we then need to know the extent of each of the user's interests. We incorporate a neural attention mechanism [25] for this purpose. Then, in order to capture the similarity between users and items, we need to be able to project them to the same latent space. We adapt Deep Structured Semantic Model (DSSM) [8] for this. DSSM was originally used for the task of web document ranking. Later, it was adapted for the task of recommendation in [9]. However, in [9] 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 in this work are as follows.

1. We present 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.
2. In order to address the changing interests of the users and the granularity/extent of these interests over time, we incorporate attentional bidirectional LSTMs which in turn helps to model the latent features of the user.
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 to solve the problems of user and item cold-start respectively.

The rest of the paper is organized as follows, first we review major approaches in recommender systems followed by a discussion on works which are directly related to ours. In Section 3, we give a brief description of the dataset used. After that, in Section 4 we provide the architecture of our model and also depict its similarity to matrix factorization. We then present a comprehensive empirical study to support our claims in Section 5. Finally, we conclude and suggest future work.

2 RELATED WORK

There has been extensive study on recommendation systems with a myriad of publications. However, the exploration of deep neural networks on recommender systems has received relatively less scrutiny. In this section, we aim at reviewing a representative set of approaches that are related to our proposed approach.

2.1 Common Approaches for Recommendation

Recommendation systems in general can be divided into collaborative 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 technique include Bayesian matrix factorization [2], matrix completion [3], Restricted Boltzmann Machine [4], nearest neighbour modelling [5] etc. In user collaborative methods such as [5], the algorithm first computes similarity between different users based on the items liked by them. Then, the scores of user-item pairs are computed by combining scores of this item given by similar users. Item based collaborative filtering [6], computes similarity between items based on the users who like both items. It then recommends items to the user based on the items she has previously liked. Finally, in user-item based collaborative filtering, both the users and the items are projected into a common vector space based on the user-item matrix and then the item and user representation are combined to find a recommendation. Matrix factorization based approaches like [3] and [2] are examples of such a technique. One of the major drawbacks of collaborative filtering is its inability to handle new users and new items, a problem which is often referred as the cold-start issue.

Another common approach for recommendation is content-based recommendation. In this approach, features from user's profile and/or item's are extracted and are used for recommending items to users based on these features. The underlying assumption is that the users tend to like items similar to those they already like. In [7], each user is modeled by a distribution over news topics that is constructed from articles she liked with a prior distribution of topic preference computed using all users who share the same location. A major advantage of using content-based recommendation is that it can handle the problem of item cold-start as it uses item features for recommendation. For user cold-start, a variety of other features like age, location, popularity aspects could be used. In the following we discuss recommendation works which use neural networks.

2.2 Neural Network based Recommendation

Early pioneer work which used neural network was done in [4], where a two-layer Restricted Boltzmann Machine (RBM) was used to model users explicit ratings on items. The work has been later extended to model the ordinal nature

of ratings [22]. Recently autoencoders have become a popular choice for building recommendation systems [24][23][19]. The idea of user-based AutoRec [23] is to learn hidden structures that can reconstruct a user's ratings given her historical ratings as inputs. In terms of user personalization, this approach shares a similar spirit as the item-item model [21][6] that represent a user as her rated item features. While previous work has lent support for addressing collaborative filtering, most of them have focused on observed ratings and modeled the observed data only. As a result, they can easily fail to learn users preference from the positive-only implicit data.

The work that is most relevant to our work is [18] and [1]. In [18] a collaborative denoising autoencoder (CDAE) for CF with implicit feedback is presented. In contrast to the DAE-based CF [19], CDAE additionally plugs a user node to the input of autoencoders for reconstructing the user's ratings. As shown by the authors, CDAE is equivalent to the SVD++ model [11] when the identity function is applied to activate the hidden layers of CDAE. Although CDAE is a collaborative filtering model, it is solely based on item-item interaction whereas the work which we present here is based on user-item interaction. On the other hand in [1], authors have explored deep neural networks for recommender systems. They present a general framework named NCF, short for Neural Collaborative Filtering that replaces the inner product 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. They then combine the linearity of matrix factorization and non-linearity of deep neural networks for modelling user-item latent structures. They call this model as NeuMF, short for Neural Matrix Factorization.

2.3 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 [8]. The model is named as Deep Semantic Structured Model (DSSM) [8] 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 [9]. In [9], the authors used DSSM for recommendation where the first neural network contains user's query history (and thus referred to as 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 in DSSM. However, in [9], 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.

For many of the approaches in recommendation systems, the objective is to minimize the root mean squared error on the user-item matrix reconstruction. However, in [10] it has been shown that ranking based objective function is more effective in generating relevant recommendations.

3 MODEL ARCHITECTURE

We first briefly review DSSM and then we provide the description of our model. We then try to show the relationship between matrix factorization and our approach.

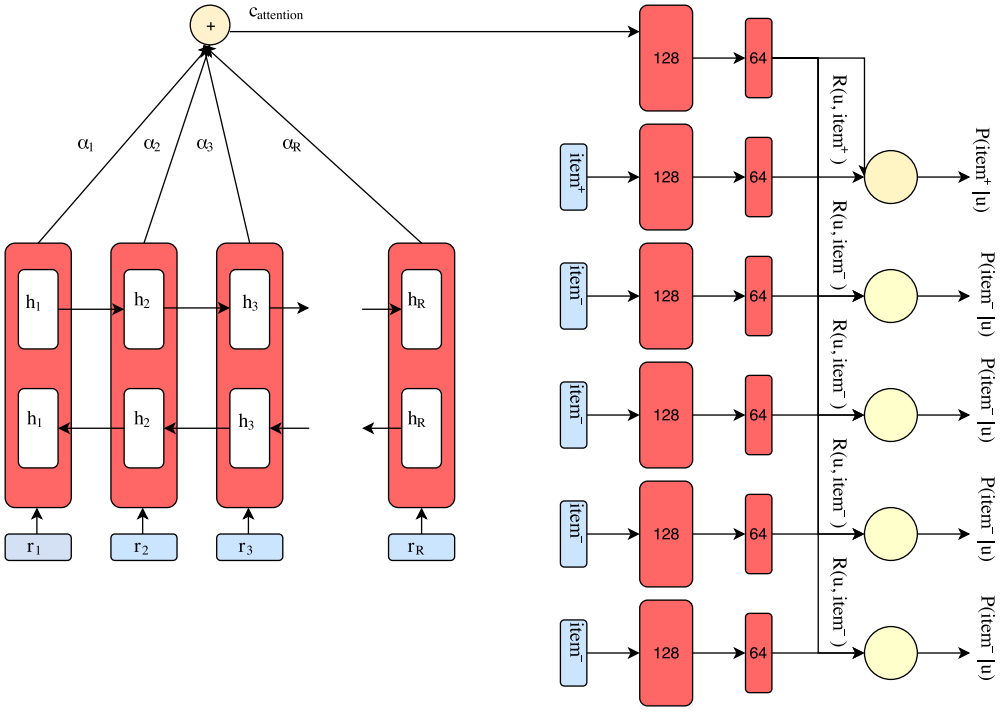

Fig. 1: Recurrent Attention DSSM Model Architecture

3.1 Recurrent Attention DSSM (RA-DSSM)

In the MV-DNN, the input to the user view was merely the query history of users. In this work, we modify the way in which inputs are sent to the user view in order to adapt it specifically for the case of news recommendation. One

of the major issues in news recommendation is that of changing user interests. Interests of users can be classified into short term as well as long term interests. Hence, it becomes crucial for a news recommender to identify these interests and recommend accordingly.

3.2 Deep Semantic Structured Model

The Deep Semantic Structured Model (DSSM) [8] 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 [9]. After that, each input vector goes through a non-linear transformation in the feedforward 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 loss on positive examples. In [9], authors used DSSM for recommendation where the first neural network contained the query history of users and the second neural network contained the implicit feedback of items (e.g News Clicks, App Downloads). The resulting model is named as multi-view DNN (MV-DNN) since it can incorporate item information from more than one domain and jointly optimize them using the same loss function in DSSM.

3.3 Recurrent Attention DSSM (RA-DSSM)

In the MV-DNN, the input to the user view was merely the query history of users. In this work, we modify the way in which inputs are sent to the user view in order to adapt it specifically for the case of news recommendation. One of the major issues in news recommendation is that of changing user interests. Interests of users can be classified into short term as well as long term interests. Hence, it becomes crucial for a news recommender to identify these interests and recommend accordingly.

LSTMs have shown to be capable of learning long-term dependencies [29][30]. Bidirectional LSTMs on the other hand can capture past and future information effectively. Users interests keep changing over time and at the time of recommendation we need to know the current interest and the long term user interest. Using Bidirectional LSTMs as an encoder helps us to identify interests which the user has taken up recently (short term) as well the long term interests of the user. For each user, we have the sequence in which news articles were read by her. We then choose the first R read articles for each user and use it as inputs to our bidirectional LSTMs. The forward state updates of the LSTM satisfy the following equations

$$[\overrightarrow{f_t}, \overrightarrow{i_t}, \overrightarrow{o_t}] = \sigma[\overrightarrow{W}[\overrightarrow{h}_{t-1}, \overrightarrow{r_t}] + \overrightarrow{b}] \tag{1}$$

$$\overrightarrow{l_t} = \tanh[\overrightarrow{V}[\overrightarrow{h}_{t-1}, \overrightarrow{r_t}] + \overrightarrow{d}] \tag{2}$$

$$\overrightarrow{c_t} = \overrightarrow{f_t} \cdot \overrightarrow{c}_{t-1} + \overrightarrow{i_t} \cdot \overrightarrow{l_t} \tag{3}$$

$$\overrightarrow{h_t} = \overrightarrow{o_t} \cdot \tanh(\overrightarrow{c_t}) \tag{4}$$

here σ is the logistic sigmoid function, $\overrightarrow{f_t}$, $\overrightarrow{i_t}$, $\overrightarrow{o_t}$ represent the forget, input and output gates respectively. $\overrightarrow{r_t}$ denotes the input at time t and $\overrightarrow{h_t}$ denotes the latent state, $\overrightarrow{b_t}$ and $\overrightarrow{d_t}$ represent the bias terms. The forget, input and output gates control the flow of information throughout the sequence. $\overrightarrow{W}$ and $\overrightarrow{V}$ are matrices which represent the weights associated with the connections. The backward states ($\overleftarrow{h}_1, \overleftarrow{h}_2, \ldots, \overleftarrow{h}_R$) are computed in a similar manner as above. The amount of reading history used as inputs to the bidirectional LSTM is denoted by R. We then concatenate the forward and backward states to obtain the annotations $(h_1, h_2, \ldots, h_R)$, where

$$h_i = \begin{bmatrix} \overrightarrow{h}_i \\ \overleftarrow{h}_i \end{bmatrix} \tag{5}$$

We then need to identify the extent/granularity of each interests. Recently in [25], the effectiveness of attention mechanisms has been shown for the task of neural machine translation. The goal of the attention mechanism in such tasks is to derive a context vector that captures relevant source side information to help predict the current target word. In our case, we want to use the sequence of annotations generated by the encoder to come up with a context vector that captures the extent of the user's interests. Though, in a typical RNN encoder-decoder framework [25], a context vector is generated at each time step to predict the target word, in our case, we only need to calculate the context vector for a single time step.

$$c_{attention} = \sum_{j=1}^{R} \alpha_j h_j \tag{6}$$

where, h_1,...,h_R represents the sequence of annotations to which the encoder maps the sequence of read news articles and each α_j represents the respective weight corresponding to each annotation h_j. The user view (left view) of the model can be seen in Figure 1. The input to this is a selected amount of reading history of each user. Each r_i in the figure is a news embedding of dimension 300.

However, the right view of the DSSM remains the same as can be seen in Figure 1. For inputs to the right view of the DSSM, we select one positive sample i.e an article that has been read by the user (apart from those that were used as input to the user view) and n randomly selected negative samples (articles that have not been read by the user). Each $item^+$, $item^-$ used as inputs to the item view is also an embedding of size 300.

3.4 Learning

Typically in matrix factorization, to learn the model parameters, existing pointwise methods [13][16] perform regression with a squared loss. This is based on the assumption that observations are generated from a Gaussian distribution. However, in [1] it has been shown that such a method does not tally well when we have implicit data available to us. Also, in [10] 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 [8]. We first compute the posterior probability of a clicked news item given a user from the relevance score using a softmax function

$$P(item^+|u) = \frac{exp(R(u, item^+))}{\sum_{\forall item} exp(R(u, item))} \tag{7}$$

where u denotes the user, $item^+$ 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

$$L(\Lambda) = -\log \prod_{u,item^+} P(item^+|u) \tag{8}$$

where, Λ represents the parameters of our model.

3.5 Relation with Matrix Factorization

We now show how we could interpret our model as a special case of matrix factorization, which is one of the most popular model for recommendation and has been investigated extensively in literature.

Matrix factorization models map both users and items to a joint latent factor space of dimensionality f, such that user-item interactions are modeled as inner products in that space. Accordingly, each item i is associated with a vector $q_i \in \mathbb{R}^f$ and each user is associated with a vector $p_u \in \mathbb{R}^f$. For a given item i, the elements of q_i measure the extent to which the item possesses those factors, positive or negative. For a given user u, the elements of p_u measure the extent of interest the user has in items that are high on the corresponding factors, again, positive or negative. The resulting dot product of the two vectors captures the interaction between the user u and item i. This approximates the user u's rating for the item i, denoted by r_{ui}, leading to the estimate

$$\hat{r_{ui}} = q_i^T p_u \tag{9}$$

The major challenge in this is to compute $q_i, p_u \in \mathbb{R}^f$. We solve this problem by using deep neural networks. The deep neural architecture allows us to learn a non-linear mapping for the users and the items to the same latent space. For computing the mapping for the users, we first use a recurrent network followed by an attention layer. Fully connected layers are then used for bringing in the

user and the items to the same latent space. In the final layer of the DSSM, we compute the similarity between the user and the item using the dot product of the non-linear mappings of the input vectors. The user can then be represented as $\Phi(u)$ and the item can be represented as $\Phi(i)$ (here Φ represents the learnt non-linear mapping). Finally we estimate the rating as,

$$\hat{r_{ui}} = \Phi(i)^T \Phi(u) \tag{10}$$

Although in [1], to compute this similarity the authors resorted to learn any arbitrary function, we learn a non-linear transformation and then utilise the dot product for computing the similarity.

4 EXPERIMENTS

We conduct experiments to answer the following questions:

1. Does our proposed model outperform the state-of-the-art implicit collaborative methods? Also, how do the different variations of our model perform for the given task.
2. How does our proposed model work for solving the item cold start problem?
3. How does our proposed model work for solving the user cold start problem?

4.1 DATASET

For this work we use the dataset published by CLEF NewsREEL 2017. CLEF NewsREEL provides an interaction platform to compare different news recommender systems performance in an online as well as offline setting [32]. 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 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.

4.2 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 [12] to learn doc2vec [27] 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 (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 user cold start problem. For the item cold start problem, we again test it on users who have read in between 10-15 articles. We ensure that the chronology of the data is kept intact.

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 [26][13][14]. 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 all items for every user during evaluation, we followed the common strategy [9][11] that randomly samples 100 items that are not interacted by the user, ranking the test item among the 100 items. The performance of a ranked list is judged by Hit Ratio (HR) and Normalized Discounted Cumulative gain (NDCG) [15]. Without special mention, 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.

Baselines : We compare our proposed approach with the following methods:

- **ItemPop**. News articles are ranked by their popularity judged by their number of interactions. This is a non-personalized method to benchmark the recommendation performance [14].
- **BPR** [14]. 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 fixing and varying the learning rate.
- **eALS** [13]. 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** [1]. This is a state-of-the-art neural matrix factorization model. It treats the problem of generating recommendation using implicit feedback as a binary classification problem. Consequently it uses the binary cross-entropy loss to optimize its model parameters.

For all the above methods we choose that number of predictive factors which maximize the performance over our given dataset.

Our proposed method is based on modelling user-item relationship, hence we mainly compare it with other user-item models. We leave out the comparison with other models like SLIM [21] and CDAE [18] because these are item-item models and hence performance difference may be caused by the user models for personalization.

Parameter Settings For the all the conducted experiments we use an Intel i7-6700 CPU @ 3.40GHz which has a RAM of 32GB and a Tesla K40c GPU. We ran all our experiments on the GPU. We implemented our proposed method using Keras [20]. As mentioned earlier, for each user who had read in between

10-15 (inclusive) articles we held out the last read article for our test set. We then construct our training set as follows:

1. We first define the reading history that we want to use. We denote the reading history by Rh.
2. For each user, we use Rh number of read articles as inputs to the user view. Leaving the last read article out, the remaining articles are used as positive samples for the item view (right view) of the model.
3. 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 training 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 hyper-parameters of our model using the validation set. All the model and its variants are learnt by optimizing the log loss of Equation 8. We initialise the fully connected network weights with the uniform distribution in the range between $-\sqrt{6/(fanin + fanout)}$ and $\sqrt{6/(fanin + fanout)}$ [28] . We used a batch size of 256 and used adadelta [17] as a gradient based optimizer for learning the parameters of the model. Also, it is worth noticing that, just in the case of NeuMF [1], where the size of the last layer of the deep network determines the number of predictive factors, we can also treat the size of the last layer of our network (just before computing the similarity) as the number of used predictive factors.

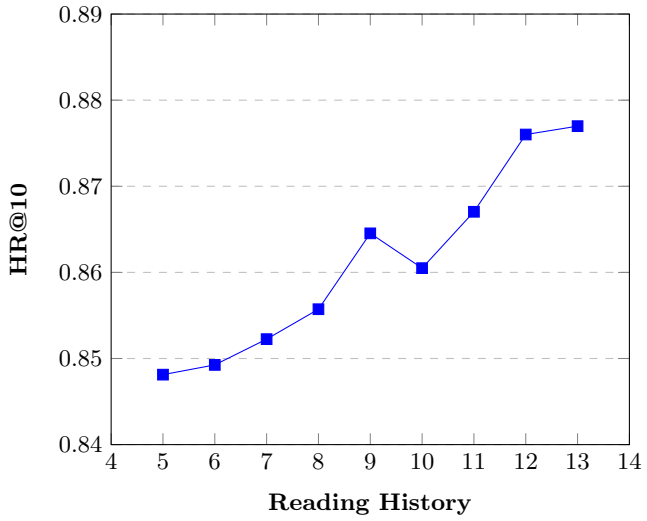

Fig. 2: HR@10 of RA-DSSM w.r.t. the User's Reading History

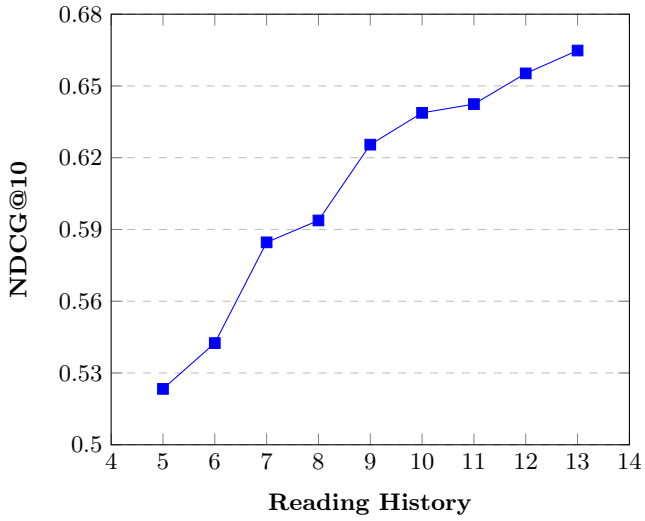

Fig. 3: NDCG@10 of RA-DSSM w.r.t. the User's Reading History

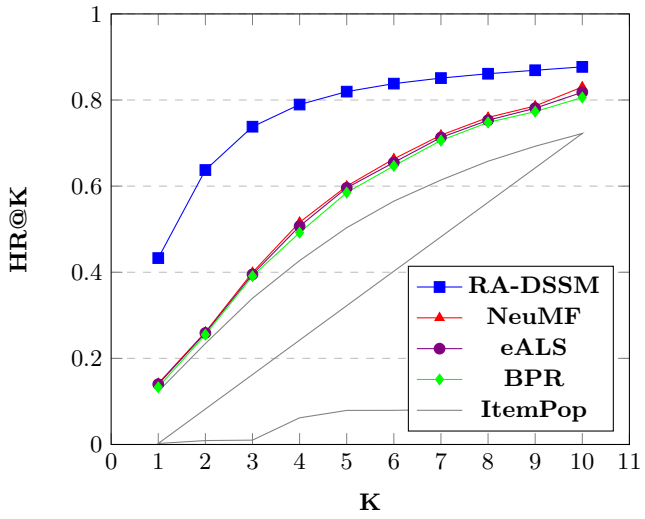

Fig. 4: HR@K performance of our model vs some state-of-the-art models

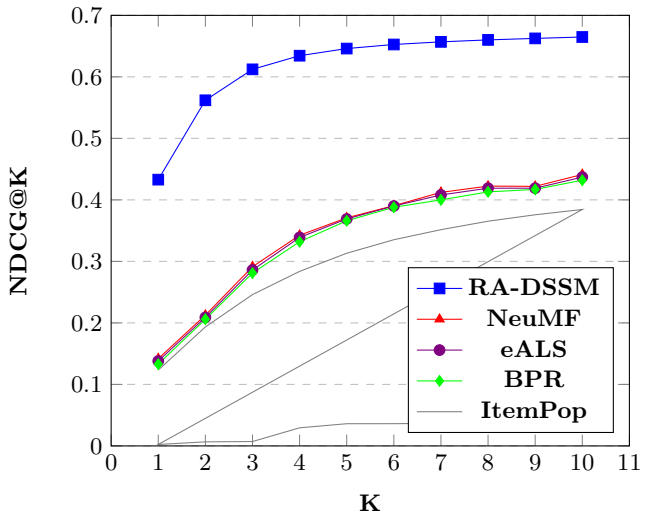

Fig. 5: NDCG@K performance of our model vs some state-of-the-art models

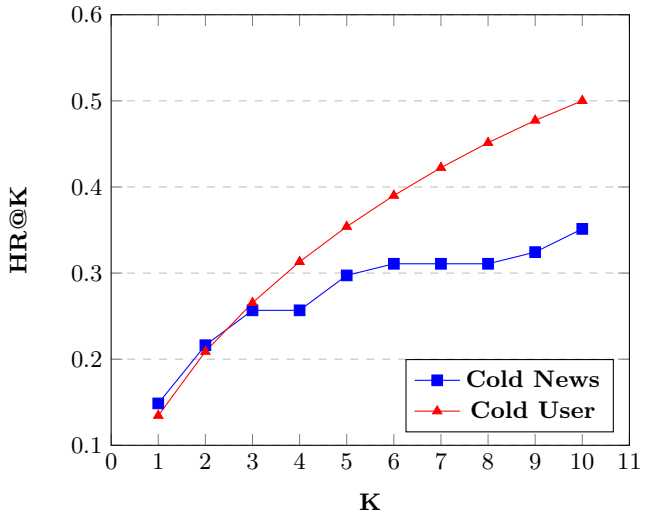

Fig. 6: HR@K of RA-DSSM on Cold-Start cases

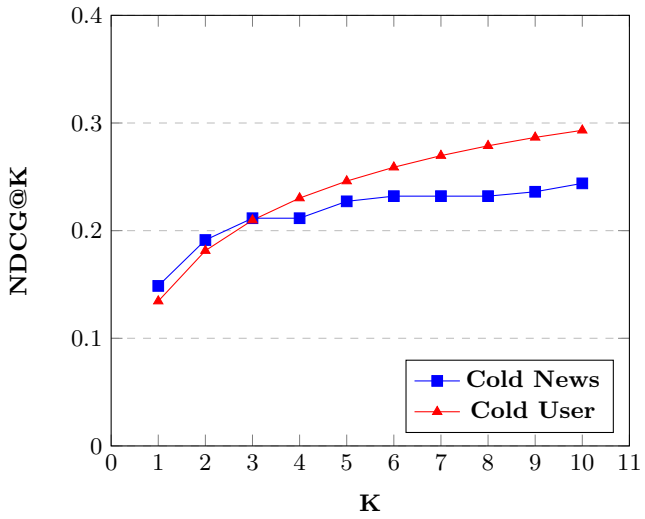

Fig. 7: NDCG@K of RA-DSSM on Cold-Start cases

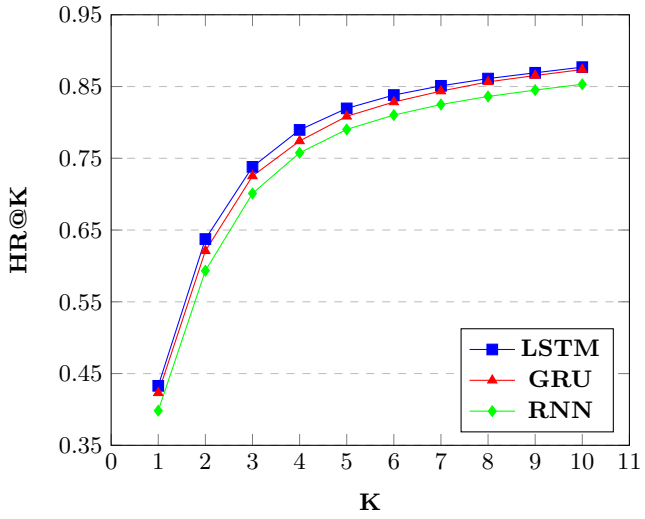

Fig. 8: HR@K for different recurrent units

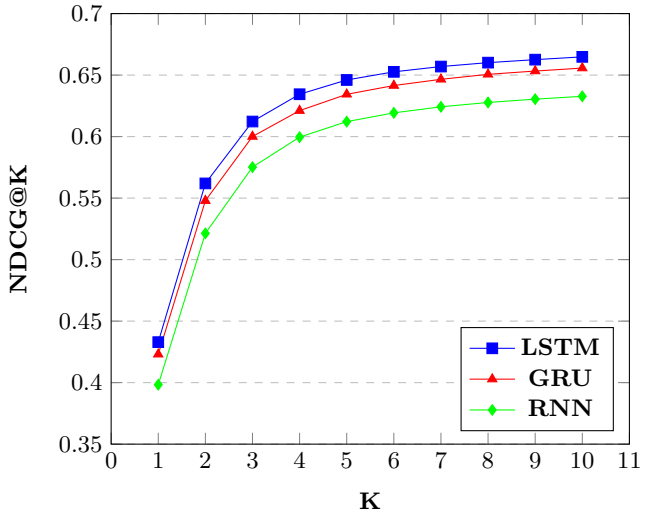

Fig. 9: NDCG@K for different recurrent units

4.3 Performance Comparison

Figure 2 and Figure 3 shows the performance of our model by varying the amount of reading history used as inputs for the user side of RA-DSSM. Overall we see that as we increase the amount of reading history used, the performance also increases. This shows that a user has multiple interests which slowly get captured as the number of articles used for the user view of RA-DSSM is increased.

Interests of the user develop and vary with time and hence we also experimented by concatenating the time at which the articles were read by each user along with the article embeddings and used these as inputs to the model. It was observed that there was no significant change in the performance. One of the prime reasons for this could be that the model is able to encode the aspect of time into itself given its sequential nature.

Figure 4 and Figure 5 shows 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 using RA-DSSM. As from the figure, 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. We observe major improvements in the NDCG scores of our model. There is an approximate 22% improvement over NeuMF. The reason for this is the loss function of Equation 8 used by our model. The loss function which is optimized for ranking, helps the model to recommend a better ranked list of items. For baseline methods we see that eALS outperforms BPR with a margin of 2%. We

also note that ItemPop performs worst which indicates the need for modelling user's personalized preferences.

We then evaluated our model for the cold start cases as can be seen in Figure 6 and Figure 7. 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 35% 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 (inclusive). The HR@10 score was around 50%. 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.

We then note the effects on our model by varying the kind of recurrent network used. We tested our model by using LSTMs, GRUs (Gated Recurrent Units) [33] and Vanilla RNN. From Figure 8 and Figure 9, the trend in the performance can be observed as follows: LSTM >GRU >RNN. One of the reasons for this could be the fact that an LSTM or a GRU is better able to encode the interests of the user. In Table 1, we note the performance by adding bidirectional units and attention layer to the LSTM. We note that Attention BiLSTM >BiLSTM >LSTM. We also note that the attention does indeed enable us to capture the extent of interests as it performs slightly better than the bidirectional LSTM.

5 CONCLUSION AND FUTURE WORK

In this work we used deep neural networks for news recommendation. We combined user-item collaborative filtering with the content of the read news articles to come up with our model. We tackled the problem of changing and diverse reading interests of users using a recurrent network combined with neural attention. We also show the effectiveness of our model in solving the problem of user cold-start and item cold-start as well. We then also show the effectiveness of our model when using one-hot item encodings. This shows the adaptability of our model for other recommendation scenarios which purely rely on implicit feedback.

In future, we would like to note the effects of learning an arbitrary function instead of using the inner product to calculate the similarity between the user and the item. We would also like to evaluate our model over different recommendation scenarios. Apart from this, we would also like to explore the idea of reinforcement based for news recommendation. Implicit feedback provided by the users could be used to model their interests and recommend articles to them.

References

1. He, Xiangnan and Liao, Lizi and Zhang, Hanwang and Nie, Liqiang and Hu, Xia and Chua, Tat-Seng: Neural Collaborative Filtering. In Proceedings of the 26th International Conference on World Wide Web, WWW '17

2. Ruslan Salakhutdinov and Andriy Mnih. 2008. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In Proceedings of the 25th inter- national conference on Machine learning. ACM, 880–887.
3. Jasson DM Rennie and Nathan Srebro. 2005. Fast maximum margin matrix factorization for collaborative prediction. In Proceedings of the 22nd international conference on Machine learning. ACM, 713–719.
4. Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. 2007. Restricted Boltzmann machines for collaborative ltering. In Proceedings of the 24th international conference on Machine learning. ACM, 791–798.
5. Robert M Bell and Yehuda Koren. 2007. Improved neighborhood-based collaborative ltering. In KDD cup and workshop at the 13th ACM SIGKDD international conference on knowledge discovery and data mining. Citeseer, 7–14.
6. Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filltering recommendation algorithms. In Proceedings of the 10th international conference on World Wide Web. ACM, 285–295.
7. Jiahui Liu, Peter Dolan, and Elin Rønby Pedersen. 2010. Personalized news recommendation based on click behavior. In Proceedings of the 15th international conference on Intelligent user interfaces. ACM, 31–40.
8. Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In Proceedings of the 22nd ACM international conference on Conference on information & knowledge management. ACM, 2333–2338.
9. Ali Mamdouh Elkahky, Yang Song, and Xiaodong He. 2015. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In Proceedings of the 24th International Conference on World Wide Web. ACM, 278–288.
10. Joonseok Lee, Samy Bengio, Seungyeon Kim, Guy Lebanon, and Yoram Singer. 2014. Local collaborative ranking. In Proceedings of the 23rd international conference on World wide web. ACM, 85–96.
11. 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. ACM, 426–434.
12. Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks. ELRA, Valletta, Malta, 45–50. http://is.muni.cz/publication/884893/en.
13. Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast matrix factorization for online recommendation with implicit feedback. In Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval. ACM, 549–558.
14. Ste en Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt- Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence. AUAI Press, 452–461.
15. 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. ACM, 1661–1670.
16. Ruslan Salakhutdinov and Andriy Mnih. 2007. Probabilistic Matrix Factorization.. In Nips, Vol. 1. 2–1.
17. Matthew D Zeiler. 2012. ADADELTA: an adaptive learning rate method. arXiv preprint arXiv:1212.5701 (2012).

18. Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In Proceedings of the Ninth ACM International Conference on Web Search and Data Mining. ACM, 153–162.
19. Florian Strub and Jeremie Mary. 2015. Collaborative Filtering with Stacked Denoising AutoEncoders and Sparse Inputs. In NIPS Workshop on Machine Learning for eCommerce.
20. François Chollet and others. 2015. Keras. `https://github.com/fchollet/keras`. (2015).
21. Xia Ning and George Karypis. 2011. Slim: Sparse linear methods for top-n recommender systems. In Data Mining (ICDM), 2011 IEEE 11th International Conference on. IEEE, 497–506.
22. Dinh Q Phung, Svetha Venkatesh, and others. 2009. Ordinal Boltzmann ma- chines for collaborative ltering. In Proceedings of the Twenty- fth Conference on Uncertainty in Arti cial Intelligence. AUAI Press, 548–556.
23. Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. Autorec: Autoencoders meet collaborative ltering. In Proceedings of the 24th International Conference on World Wide Web. ACM, 111–112.
24. Minmin Chen, Zhixiang Xu, Fei Sha, and Kilian Q Weinberger. 2012. Marginalized Denoising Autoencoders for Domain Adaptation. In Proceedings of the 29th International Conference on Machine Learning (ICML-12). 767–774.
25. Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural ma- chine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014).
26. Immanuel Bayer, Xiangnan He, Bhargav Kanagal, and Ste en Rendle. 2017. A Generic Coordinate Descent Framework for Learning from Implicit Feedback. In Proceedings of the 26th International Conference on World Wide Web (WWW '17).
27. Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In Proceedings of the 31st International Conference on Machine Learning (ICML-14). 1188–1196.
28. Xavier Glorot and Yoshua Bengio. 2010. Understanding the di culty of training deep feedforward neural networks.. In Aistats, Vol. 9. 249–256.
29. Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. Neural computation 9, 8 (1997), 1735–1780.
30. Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Advances in neural information processing systems. 3104–3112.
31. Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. IEEE Transactions on Signal Processing 45, 11 (1997), 2673–2681.
32. Frank Hopfgartner, Torben Brodt, Jonas Seiler, Benjamin Kille, Andreas Lommatzsch, Martha Larson, Roberto Turrin, and András Serény. 2016. Benchmarking news recommendations: The clef newsreel use case. In ACM SIGIR Forum, Vol. 49. ACM, 129–136.
33. Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555 (2014).