HyperNews: Simultaneous News Recommendation and Active-Time Prediction via a Double-Task Deep Neural Network

Rui Liu1, Huilin Peng1, Yong Chen2* and Dell Zhang3,4

1School of Computer Science and Engineering, Beihang University, Beijing, China
2School of Electronics Engineering and Computer Science, Peking University, Beijing, China
3Birkbeck, University of London, London, United Kingdom
4Blue Prism AI Labs, London, United Kingdom
{lr, penghuilin}@buaa.edu.cn, alphawolf.chen@gmail.com, dell.z@ieee.org

Abstract

Personalized news recommendation can help users stay on top of the current affairs without being overwhelmed by the endless torrents of online news. However, the freshness or timeliness of news has been largely ignored by current news recommendation systems. In this paper, we propose a novel approach dubbed HyperNews which explicitly models the effect of timeliness on news recommendation. Furthermore, we introduce an auxiliary task of predicting the so-called “active-time” that users spend on each news article. Our key finding is that it is beneficial to address the problem of news recommendation together with the related problem of active-time prediction in a multi-task learning framework. Specifically, we train a double-task deep neural network (with a built-in timeliness module) to carry out news recommendation and active-time prediction simultaneously. To the best of our knowledge, such a “kill-two-birds-with-one-stone” solution has seldom been tried in the field of news recommendation before. Our extensive experiments on real-life news datasets have not only confirmed the mutual reinforcement of news recommendation and active-time prediction but also demonstrated significant performance improvements over state-of-the-art news recommendation techniques.

1 Introduction

Nowadays, with massive news aggregated from various channels every minute by online news platforms such as Google News and Toutiao, it is impossible for users to read through all of them [Phelan et al., 2011; Morales et al., 2012; Okura et al., 2017; Lian et al., 2018; Wu et al., 2019c]. Therefore, personalized news recommendation, which can help users cope with such information overload, has recently become a hot research topic [Cheng et al., 2016; Gulla et al., 2017; An et al., 2019; Zhu et al., 2019].

Basically, the core problem in news recommendation is to learn news and user representations. The early studies in this area typically encode news data with static features such as the title, content and categories of each news article. For example, to employ the acclaimed LibFM [Rendle, 2012] that combines the generality of feature engineering with the superiority of factorization models for news recommendation, we could simply concatenate the news features and the user features as inputs, and then get the corresponding click probabilities as outputs. Specifically, the news features could consist of the TF-IDF vectors representing the given news article’s title and content as well the one-hot vector of its categories, while the user features could be extracted from the past news articles read by that particular user in the same fashion. Moreover, both DeepFM [Guo et al., 2017] and Wide&Deep [Cheng et al., 2016] try to further capture the higher-order interactions in addition to the linear and pairwise ones between features. Thus they share the same inputs as LibFM and also generate click probabilities. Similarly, DSSM [Huang et al., 2013] projects queries and documents into a common low-dimensional space where the relevance of a document to the given query could be computed by their cosine similarity score. It could also be utilized to generate news and user representations for news recommendations, as in LibFM. To sum up, the above methods are general-purpose recommender systems mainly using static features; they probably would not be able to handle the dynamics between news and users very well.

To deal with the possible drift of users’ interests, the latest solutions for personalized news recommendation utilize not only the above-mentioned static features but also the features extracted from each user’s recently read news articles to approximate his/her current reading interests. For example, DKN [Wang et al., 2018] utilizes a multi-channel word-entity-aligned knowledge-aware CNN which fuses semantic and knowledge-level representations to construct a news encoder, and designs an attention module to dynamically aggregate a user’s history w.r.t. current candidate news articles. NAML [Wu et al., 2019a] adaptively selects informative representations for news and users by designing a multi-view attentive mechanism. NPA [Wu et al., 2019b] proposes a personalized attention network which exploits user-ID embeddings as queries for the word- and news-level attention networks to realize the differentiated dynamics. LSTUR [An et al., 2019] aims to learn long-term user representations from the embeddings of their IDs and short-term user representa-
tions from their recently browsed news articles via a GRU network to combine the long-term preferences and short-term interests for better personalized recommendation. In summary, the above relatively new approaches to news recommendation attempt to capture users’ current interests implicitly through their recently read news articles. However, as we will show later, it would be even better to exploit the temporal attributes in the news dataset and model the timeliness of news explicitly.

Furthermore, we reckon that the so-called ‘active-time’ (i.e., the time interval from the user’s click opening the news article page to the user’s click closing it) attribute measures the time spent by the user to read that particular news article, and thus can clearly reflect that user’s reading interests: ceteris paribus, the longer the ‘active-time’ (reading time), the more the user is interested in the article. The problem of active-time prediction is itself meaningful for some practical applications like display advertising and network traffic control. It also needs to model the interaction between news articles and news consumers. Due to the apparent correlation between the active-time prediction task and the news recommendation task, we conjecture that it would be promising to address them together.

Contributions. Motivated by the above two considerations, we put forward a new approach to personalized news recommendation, named HyperNews, which introduces an explicit timeliness module to refine news representation, and also an auxiliary task of active-time prediction to reinforce news recommendation in the multi-task learning framework. Our experiments on real-world news datasets show that HyperNews outperforms state-of-the-art news recommendation techniques. Our analysis of the experimental results confirm that the timeliness module and the active-time prediction task both contribute to the performance improvements.

2 Problem Statement

In the problem of personalized news recommendation, we try to predict how likely a user will click on a previously unseen news article based on his or her clicking history. The recently available event-based news data (describing which user clicked on which news article at what time and closed it at what time) provide us with opportunities to carry out news recommendation in a richer context. In particular, we would argue that a user’s active-time on a news article could be, and probably should be, predicted at the same time. Here, we propose a unified model to undertake these two tasks simultaneously.

Suppose a news dataset for training consists of \( n \) instances \((x, y_p, y_t)\), where \( x \) is the composite feature vector representing a pair of user and news article, \( y_p \) is the binary label indicating whether the user clicked on that news article \((y_p = 1)\) or not \((y_p = 0)\), and \( y_t \) records the corresponding active-time. Intuitively, the news recommendation task (i.e., estimating \( y_p \) based on \( x \)) and the active-time prediction task (i.e., estimating \( y_t \) based on \( x \)) should be correlated with each other. Therefore, it would make perfect sense to exploit the commonalities across these two tasks by utilizing the multi-task learning framework: \( (\hat{y}_p, \hat{y}_t) = \text{DoubleTaskModel}(x) \) where \( \hat{y}_p \) is the predicted click-probability and \( \hat{y}_t \) is the predicted active-time. Obviously, the click-probability output can be used to determine how highly the given news article should be recommended to the respective user. To make the active-time prediction task easier, we discretize the continuous active-time values into discrete time interval groups (e.g., “15-25 seconds”), and thus convert the regression problem into a classification problem. It turns out that such an aggressive simplification worked really well in our experiments, suggesting that a crude estimate of active-time would be good enough for our purposes.

3 The Method: HyperNews

Fig. 1a shows the overall framework of our proposed method, HyperNews, which can simultaneously conduct news recommendation and active-time prediction in the manner of multi-task learning. These two tasks would share the same inputs, i.e., the attributes of news and users including ‘news-title’, ‘news-content’, ‘news-category’, ‘publish-time’, ‘click-time’, and ‘active-time’. However, the news encoder for these two tasks would use two different sets of learnable weights to...
combine explicit embedding and implicit embedding as they are likely to have different impacts upon different tasks (see Section 3.1). It is worth mentioning that for news recommendation, we have designed and included a timeliness module (see Section 3.3) in order to emphasize the impact of news freshness (i.e., the elapsed time between the ‘publish-time’ of the news article and the ‘click-time’ when the user clicked to open it for reading).

3.1 News Encoder

The news encoder (Fig. 1b) is used to learn news embedding from various news attributes (i.e., title, content and categories). We divide these attributes into two types: explicit information and implicit information. The explicit information includes the news article’s title and categories that users can see before making their decisions to click, while the implicit information refers to the news article’s content that a user can see only after clicking. The reason why we differentiate them is that they should have different effects on click-probability and active-time. Intuitively, the explicit information would have a greater impact on click-probability than the implicit information. Once the embeddings of these two parts have been obtained, an attention unit would be utilized to adaptively learn their combinations for different tasks.

Explicit Embedding

The explicit embedding unit contains two components.

The first is to embed news titles using a typical CNN-based model with a sequence of words as inputs. Specifically, a news title is represented as a word embedding based matrix $w_{1:n} = \{w_1, w_2, ..., w_n\} \in \mathbb{R}^{d \times l}$, where $w_i \in \mathbb{R}^{d \times l}$ is the $i$-th word of the news title, $d = 100$ denotes the dimension of word embedding, and $n$ represents the number of words. Then the matrix $w_{1:n}$ would be applied with a convolution operator via a filter $H \in \mathbb{R}^{d \times l}$, where $l = 3 (l < n)$ means the window size. For each sub-matrix $w_{i:i+l-1}$, we can get a feature $c_i$ by:

$$c_i = f(H \ast w_{i:i+l-1} + b),$$

where $\ast$ is the convolution operator, $b$ is a bias term, and $f$ is the ReLU activation function. With all the possible positions of the matrix $w_{1:n}$ crossed, we get a feature map:

$$c = [c_1, c_2, ..., c_{n-l+1}]^T,$$

where $c_i$ marks the $i$-th result of the filter $H$. Then we utilize a max-pooling operation to identify its most significant feature:

$$\bar{c} = \max\{c\} = \max\{c_1, c_2, \cdots, c_{n-l+1}\}.$$  

Next, we take $m = 200$ filters with the same window sizes (i.e., $l = 3$) to conduct the same operation as above, and each filter will generate a feature. As a result, the $m$ features are concatenated together to finally represent the news title as:

$$e_h = [\bar{c}_1, \bar{c}_2, \cdots, \bar{c}_m]^T.$$  

The second component is to embed news categories. The reason why we take them into consideration is that users usually only care about the news articles within those categories of their interests and would barely click on any news article outside. Regarding this part, the input for each category is a unique id, which is randomly initialized with a low-dimensional (e.g., 100D) vector $e_c$. After that, a dense layer is employed to learn the category’s hidden representation:

$$e'_c = \tanh(W_c \times e_c + b_c),$$

where $W_c$ and $b_c$ are learnable parameters. Since each news article may have a different number of categories and usually different categories have different influences, we choose the top three most important categories and then use an attention unit to calculate a weight for each one. The final representation $\bar{e}_c$ for news categories is the weighted sum of their hidden representations. Note that if a news article has fewer than three categories, we would pad the input with zero vectors. Denote the weight of the $i$-th category as $\alpha^c_i$, then there holds:

$$\alpha^c_i = q^c_i \times \tanh(V_c \times e'_c + b'_c), (i = 1, 2, 3),$$

$$\bar{\alpha}^c_i = \frac{\exp(\alpha^c_i)}{\sum_j \exp(\alpha^c_j)}, (j = 1, 2, 3),$$

and

$$\bar{e}_c = \sum_i \bar{\alpha}^c_i e'_c, (i = 1, 2, 3),$$

where $q^c$, $V_c$ and $b'_c$ are the parameters of the attention unit.

In the end, the representations of news-title and news-categories would be concatenated together, and then fed into a one-layer fully connected neural network (FNN) to get a low-dimensional embedding as the representation of the explicit information:

$$e_{N_e} = \text{FNN}([e_h; \bar{e}_c]).$$

Implicit Embedding

The implicit embedding unit is to embed news content.

First, we train the LDA [Blei et al., 2003] topic model on the whole Adressa\textsuperscript{1} news dataset, based on which we could infer a probabilistic representation for any news article’s content. Let $D$ denote the news article’s content, then the topic representation (#dimensions=100), regarded as part of the input, is given by:

$$e_d = \text{LDA}(D).$$

Second, we use doc2vec [Le and Mikolov, 2014], also trained on Adressa, to construct another document representation (#dimensions=100) as:

$$e_d = \text{doc2vec}(D).$$

Through the utilization of doc2vec, the semantic representation of news content would be further enriched. Note that the above two steps can both be processed in advance and therefore would not incur extra computational costs for our model. Third, the length of news content is obviously a big factor affecting users’ active-time on that piece of news, so we include it as useful feature for our model. Specifically, the raw lengths of news content would be divided into a series of bins each with an interval of 50 words, and then similar to word embedding, we represent each news-length-bin feature as a low-dimensional (e.g., 100D) vector $e_l$. It would be randomly initialized and then go through a fully-connected layer to become the final representation:

$$e'_l = \tanh(W_l \times e_l + b_l),$$

where $W_l$ and $b_l$ are learnable parameters. Since each news article may have a different number of length-bins and usually different length-bins have different influences, we choose the top three most important length-bins and then use an attention unit to calculate a weight for each one. The final representation $\bar{e}_l$ for news-lengths would be the weighted sum of their hidden representations.
where $e_t$, $W_f$ and $b_f$ are to-be-learned parameters.

Finally, we concatenate these three vectors and feed them to a one-layer FNN to represent the implicit information:

$$e_{N_i} = \text{FNN}( [e_t; e_d; e'_t] ).$$

\[ (13) \]

**Attention Unit**

Intuitively, the explicit information and the implicit information would exert different influences on click-probability and active-time. Therefore we introduce an attention unit to automatically learn the weights for these two different parts. Denote the two weights as $\alpha_1$ and $\alpha_2$, we have:

$$\alpha_1 = q_n^T \times \text{tanh}(V_n \times e_{N_i} + b_n),$$

$$\alpha_2 = q_e^T \times \text{tanh}(V_e \times e_{N_i} + b_e),$$

and

$$\tilde{\alpha}_n = \frac{\exp(\alpha_n)}{\sum_i \exp(\alpha_n)},$$

\[ (16) \]

where $q_n$, $V_n$, and $b_n$ are learnable parameters. Afterwards, the news embedding is calculated as:

$$e = \tilde{\alpha}_1 e_{N_e} + \tilde{\alpha}_2 e_{N_l}.$$  

\[ (17) \]

Note that in the HyperNews model, we actually would have three different sets of values for those parameters ($q_n$, $V_n$, $b_n$) which would lead to three separate news encodes $e^n$, $e^e$ and $e^u$ for click-probability, active-time, and user representation respectively (see Fig. 1a).

**3.2 User Encoder**

The user encoder is to learn a user’s feature representation from that user’s click history. Today, there are two ways which have been proved effective. To be specific, recurrent networks (like GRU and LSTM) are skilled at encoding sequential data, which would lead to three separate news encodes $e^n$, $e^e$ and $e^u$ for click-probability, active-time, and user representation respectively (see Fig. 1a).

Firstly, we divide the continuous time into variable-length time interval bins $[1h, 2h, \cdots, 1d, 2d, \cdots, 1m, 2m, \cdots]$, where $h$, $d$ and $m$ represent hour, day and month respectively. For example, if it is 75-minutes, we would assign it to the $2h$ bin and set its id as 2. For each interval id, a low-dimensional (e.g., 200D) vector $e_f$ could be randomly initialized and then forwarded to a dense layer:

$$e'_f = \text{tanh}(W_f \times e_f + b_f),$$

\[ (21) \]

where $e_f$, $W_f$ and $b_f$ are parameters to be learned. In what follows, a new time-enriched news embedding for news recommendation can be reached via:

$$e_N = e^n \circ e'_f,$$

\[ (22) \]

where $\circ$ is the Hadamard (element-wise) multiplication operator. Finally, the click-probability (how likely a user clicks on a news article) is calculated by the sigmoid function of the dot product between user embedding and news embedding, i.e.,

$$\hat{y}_p = \text{sigmoid}(e_U^T \times e_N),$$

\[ (23) \]

based on which our news recommender system can recommend to each user the corresponding top-$k$ (e.g., $k=20$) news articles with the highest scores.

**3.4 Active-Time Prediction**

Making very precise estimations of active-time is challenging, but actually it may not be really necessary. For example, whether a user has spent 50 seconds or 60 seconds on a news article probably would not make much difference. Therefore we convert the continuous time regression task into a discrete time interval classification task. In what follows, we plot “the number of events (#events) vs. active-time” of the whole Adressa dataset in Fig. 2a and find that it is a long-tailed distribution; then we partition the active-time range $[5, 205]$ equally into 20 time interval bins (Fig. 2b). The reasons why we conduct this division are twofold: (1) the events of active-time in the range of $[5, 205]$ account for about 90% of the cases; (2) the active-time values fewer than 5 seconds are probably anomalies. Note that the 20 time intervals are numbered $1, 2, \cdots, 20$ as class labels.

In the end, we combine the news embedding and user embedding as $e_T = [e_U; e_T]$, and then utilize a softmax classifier to calculate the probabilities over the 20 time intervals:

$$\hat{y}_t = \text{softmax}(W_t \times e_T + b_t),$$

\[ (24) \]

where $W_t$ and $b_t$ are the classifier’s parameters. Afterwards, we conduct active-time prediction by selecting the most likely time interval corresponding to the largest element of $\hat{y}_t$. 

![Figure 2: The distribution of events in the Adressa news dataset.](image-url)
3.5 Model Training

Similar to Ref. [Zhu et al., 2019], we also regard the observed click events as positive samples and unobserved ones as negative samples. In what follows, we denote a training sample as $X = \{x_1, \cdots, x_r, x_{r+1}, y_p, y_t\}$, where $\{x_1, \cdots, x_r\}$ represents the $r$ news articles recently read by the user, $x_{r+1}$ is the candidate news article, $y_p$ is the click label (1 for clicked and 0 otherwise), and $y_t$ is the active-time label (a one-hot vector for the time interval id; invalid for negative samples). The output of our double-task HyperNews model would include both $\hat{y}_p$ and $\hat{y}_t$: their respective loss functions are as follows:

$$L_p = -\left\{ \sum_{x \in S^+} y_p \log(\hat{y}_p) + \sum_{x \in S^-} (1 - y_p) \log((1 - \hat{y}_p)) \right\},$$

and

$$L_t = -\sum_{x \in S^+} y_t \log(\hat{y}_t),$$

where $S^+$ and $S^-$ are the set of positive samples and the set of negative samples respectively, while $S^+$ is the set of positive samples with the active-time attribute.

As Fig. 2b clearly says, the active-time prediction task is probably susceptible to severe class imbalance. Therefore we adjust each class’s weight based on its effective size [Cui et al., 2019]. Specifically, the effective size of a class is defined as:

$$E_m = (1 - \beta^m)/(1 - \beta),$$

where $m$ is the actual number of samples in the class, and $\beta$ is a hyperparameter whose value is usually close to 1 (e.g., 0.99, 0.999, etc.). Accordingly, the class-balanced variant of $L_t$ can be written as:

$$L'_t = -\frac{1}{E_{m_y}} \sum_{x \in S^+} y_t \log(\hat{y}_t),$$

where $m_y$ is the actual number of samples in class $y_t$.

Taking both $L_p$ (Eq. (25)) and $L'_t$ (Eq. (28)) into consideration, the overall loss function to minimize is:

$$L = L_p + \lambda L'_t,$$

where $\lambda$ is a non-negative parameter to balance the importance of news recommendation and active-time prediction.

During training, we apply two regularization techniques, dropout (rate=0.5) and batch normalization, on the fully-connected layers to avoid overfitting. The popular Adam optimizer (with learning rate=1e-6) is employed. The batch size has been set to 400, while the number of epochs for convergence has been set to 20.

4 Experiments

In this section, we examine the effectiveness of HyperNews by comparing it with some competitive baselines.

4.1 Datasets

Adressa\textsuperscript{1} [Gulla et al., 2017] is an event-based real-life news data collection that contains a large number of Norwegian news articles in conjunction with their readers. In our experiments, we represent each news reading event by its most important attributes: ‘user-id’, ‘news-id’, ‘news-title’, ‘news-content’, ‘news-category’, ‘publish-time’, ‘click-time’, and ‘active-time’. Among them, the three temporal attributes — ‘publish-time’, ‘click-time’, and ‘active-time’ — have not been considered in previous news recommendation studies, but we have found that they could be exploited for more accurate modeling of the interaction between users and news articles.

Two editions of Adressa dataset, dubbed Adressa-1\textit{week} and Adressa-4\textit{week}, have been constructed, which correspond to the first 1 week’s and the first 4 weeks’ news consumption event-logs, respectively. For Adressa-1\textit{week}, we take the first six days’ events to train predictive models and the last day’s events to test their performance; while for Adressa-4\textit{week}, we take the first three weeks’ events as the training set and the last week’s events as the test set. Table 1 shows some characteristics of these two datasets.

![Table 1: The descriptive statistics of our datasets.](image)

### Table 1: The descriptive statistics of our datasets.

<table>
<thead>
<tr>
<th>Number</th>
<th>Adressa-1\textit{week}</th>
<th>Adressa-4\textit{week}</th>
</tr>
</thead>
<tbody>
<tr>
<td>#users</td>
<td>601,215</td>
<td>1,540,168</td>
</tr>
<tr>
<td>#news-articles</td>
<td>17,692</td>
<td>37,067</td>
</tr>
<tr>
<td>#words-per-title</td>
<td>6,63</td>
<td>6,50</td>
</tr>
<tr>
<td>#words-per-content</td>
<td>552,15</td>
<td>530,86</td>
</tr>
<tr>
<td>categories</td>
<td>245</td>
<td>252</td>
</tr>
<tr>
<td>events</td>
<td>3,123,261</td>
<td>11,584,797</td>
</tr>
<tr>
<td>events-with-‘active-time’</td>
<td>1,062,793</td>
<td>3,951,288</td>
</tr>
</tbody>
</table>

4.2 Competitors and Metrics

To evaluate the performance of our HyperNews, several state-of-the-art competitors have been invited for comparison, including LibFM\textsuperscript{2} [Rendle, 2012], DeepFM\textsuperscript{3} [Guo et al., 2017], Wide&Deep\textsuperscript{4} [Cheng et al., 2016], DSSM\textsuperscript{5} [Huang et al., 2013], LSTUR\textsuperscript{6} [An et al., 2019], and DAN\textsuperscript{7} [Zhu et al., 2019]. Moreover, to verify the usefulness of multi-task learning in our application, we construct two single-task editions of HyperNews by removing some parts of the model: (1) HyperNews\textsuperscript{8\textsc{NoPred}} which is a single-task news recommendation model without the active-time prediction component, and (2) HyperNews\textsuperscript{8\textsc{NoRecom}} which is a single-task active-time prediction model without the news recommendation component.

We have made the source code of HyperNews and also the processed datasets for our experiments available online\textsuperscript{8}.

Regarding the performance measures, we employ the popular AUC [Fawcett, 2006] and $F_1$ [Li et al., 2008] metrics for news recommendation, and just $F_1$ for active-time prediction.

---

\textsuperscript{2}https://github.com/srendle/libfm
\textsuperscript{3}https://github.com/ChenglongChen/tensorflow-DeepFM
\textsuperscript{4}https://github.com/kaitolucifer/wide-and-deep-learning-keras
\textsuperscript{5}https://github.com/ChenglongChen/tensorflow-DeepFM
\textsuperscript{6}https://github.com/InsaneLife/dssm
\textsuperscript{7}The source code is kindly provided by the authors.
\textsuperscript{8}https://github.com/zhuqiannan/dan-for-news-recommendation
\textsuperscript{9}https://github.com/penghll/Hypernews

---

\textsuperscript{1}http://reclab.idi.ntnu.no/dataset/
Table 2: News recommendation performance ($AUC$ and $F_1$).

<table>
<thead>
<tr>
<th>Method</th>
<th>Adressa-1week</th>
<th>Adressa-4week</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$AUC$</td>
<td>$F_1$</td>
</tr>
<tr>
<td>LibFM</td>
<td>0.7025</td>
<td>0.6953</td>
</tr>
<tr>
<td>DeepFM</td>
<td>0.7358</td>
<td>0.7288</td>
</tr>
<tr>
<td>Wide&amp;Deep</td>
<td>0.7415</td>
<td>0.7244</td>
</tr>
<tr>
<td>DSSM</td>
<td>0.7511</td>
<td>0.7232</td>
</tr>
<tr>
<td>LSTUR</td>
<td>0.8077</td>
<td>0.7558</td>
</tr>
<tr>
<td>DAN</td>
<td>0.8234</td>
<td>0.7676</td>
</tr>
<tr>
<td>HyperNews$_{(\text{NoPred})}$</td>
<td>0.8377</td>
<td>0.7798</td>
</tr>
<tr>
<td>HyperNews</td>
<td><strong>0.8661</strong></td>
<td><strong>0.8027</strong></td>
</tr>
</tbody>
</table>

Table 3: Active-time prediction performance ($F_1$).

<table>
<thead>
<tr>
<th>Method</th>
<th>Adressa-1week</th>
<th>Adressa-4week</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$F_1$</td>
<td>$F_1$</td>
</tr>
<tr>
<td>HyperNews$_{(\text{NoRecom})}$</td>
<td>0.8398</td>
<td>0.7843</td>
</tr>
<tr>
<td>HyperNews</td>
<td><strong>0.8946</strong></td>
<td><strong>0.8590</strong></td>
</tr>
</tbody>
</table>

4.3 Settings

There are two groups of competitors: general-purpose recommender systems (LibFM, DeepFM, DSSM, Wide&Deep) and news-oriented recommender systems (LSTUR, DAN). In accordance with the widely-adopted settings in Refs. [An et al., 2019; Zhu et al., 2019], for the former group, we just take the concatenation of news title, content and categories as input features; whereas for the latter group, we adopt the configurations as described in each method’s corresponding paper. The baseline methods’ parameters have been tuned on Adressa-1week to get competitive performances and then applied to both datasets. In the same way, the parameters of HyperNews are configured as $\lambda = 0.8$ and $\beta = 0.99999$ (see Section 4.5).

4.4 Results

Table 2 shows the $AUC$ and $F_1$ scores of different news recommendation methods. Clearly, our multi-task learning approach — the complete HyperNews system — defeats all the other competitors, which confirms its high effectiveness. It can be observed that the news-oriented models (LSTUR, DAN, and HyperNews) can usually achieve higher scores than the general-purpose models (LibFM, DeepFM, Wide&Deep, and DSSM), probably because the former only resort to basic features (like title, content, categories) but ignore task-specific information utilized in the latter. Moreover, the double-task HyperNews model exhibits a substantial performance gain over the single-task HyperNews$_{(\text{NoPred})}$ model, which tells us that the active-time prediction task does help the news recommendation task through their joint learning of the neural network.

In addition, the results of active-time prediction are shown in Table 3. The $F_1$ scores of the double-task HyperNews model are significantly higher than those of the single-task HyperNews$_{(\text{NoRecom})}$ model, which suggests that the news recommendation task could reinforce the active-time prediction task as well.

4.5 Hyperparameter Analysis

HyperNews contains two hyperparameters: $\lambda$ that controls the balance between news recommendation and active-time prediction, and $\beta$ that is used to calculate each class’s effective size. Keeping $\beta = 0.99999$ fixed, we vary $\lambda$ from 0 to 5 and plot the experimental results in Fig. 3a. It can be seen that on Adressa-1week, no matter what task it is and which metric is adopted, $\lambda = 0.8$ always leads to the best performance. Similarly, keeping $\lambda = 0.8$ fixed, Fig. 3b reveals that when $\beta = 0.99999$, HyperNews yields the best results for both tasks in terms of both metrics. Note that when $\beta = 0$, Eq. (28) would be equal to Eq. (26).

4.6 Ablation Study

Fig. 4 compares the complete HyperNews system with the ablated version of HyperNews without the timeliness module. It is clear that on both datasets, for both tasks, under both performance measures, the complete HyperNews outperforms the ablated HyperNews consistently and substantially. This confirms the critical contribution of the timeliness module.

5 Conclusion

This paper investigates the usefulness of temporal attributes to news recommendation, which has been neglected before. Specifically, we propose a novel deep neural network model named HyperNews which includes a timeliness module and utilizes a multi-task learning framework (to conduct news recommendation and active-time prediction simultaneously). Our extensive experiments on real-life news datasets have confirmed that the explicit timeliness module and the auxiliary active-time prediction task do benefit news recommendation greatly and thus enable HyperNews to beat a number of state-of-the-art news recommendation techniques.
References


