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

Anonymous

Abstract

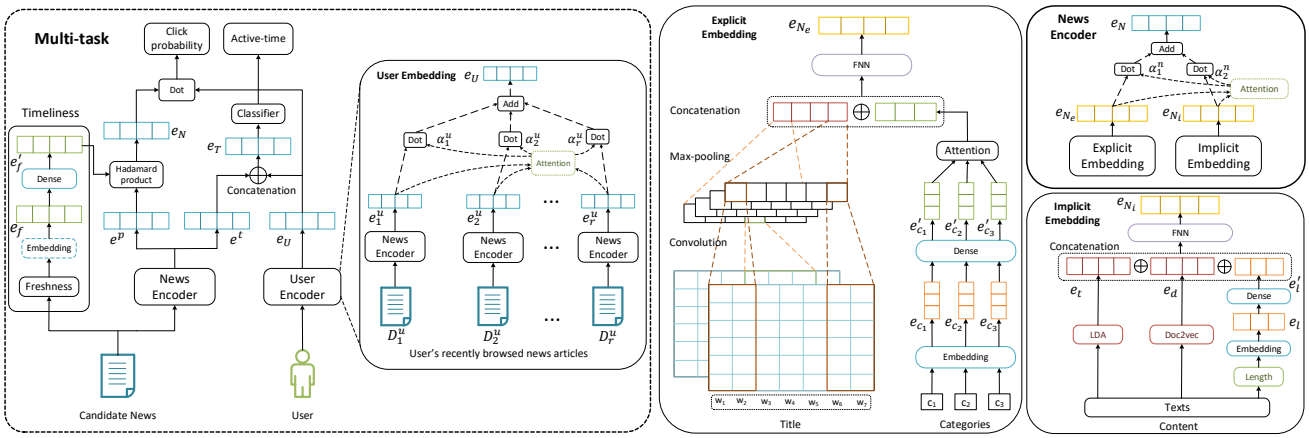
Personalized news recommendation can help users stay on top of the current affairs without being overwhelmed by the endless torrents of online news. However, the freshness or timeliness of news has been largely ignored by current news recommendation systems. In this paper, we propose a novel approach 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 really 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 not 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 everyday 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 overcome the 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, LibFM [Rendle, 2012] utilizes factorized interactions between variables for prediction tasks, hence it could concatenate the user features and the candidate news' features as inputs and then output a click-probability for deciding whether to recommend or not. Specifically, the user features could be the concatenation of TF-IDF features extracted from the browsed news titles and contents, and the normalized count features from the categories of the browsed news; while the news features could be the concatenation of TF-IDF features from its title, contents and one-hot vectors of its categories. Likewise, approaches like DeepFM [Guo *et al.*, 2017] and Wide&Deep [Cheng *et al.*, 2016] both 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 all generate click probabilities. As a comparison, DSSM [Huang *et al.*, 2013] projects queries and documents into a common low-dimensional space where the relevance of a document given a query could be computed by their cosine values. Therefore, the similar news and user representations as LibFM could be applied for news recommendations. 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 of 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 currect reading interests. For example, DKN [Wang *et al.*, 2018] features a multi-channel and word-entity-aligned knowledge-aware CNN which fuses semantic- and knowledge-level representations to construct news encoder, and designs an attention module to dynamically aggregate a user's history w.r.t. current candidate news. 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 the embeddings of user ID as the queries of 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 representations from their recently browsed news via GRU network to achieve the long-term preferences and short-term interests for



(a) General Framework. (b) News Encoder.

Figure 1: Our proposed HyperNews model.

better personalized recommendations. 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 to open the news article page to the user' click to close it) attribute measures the time spent by the user to read that particular news article, and thus can partially reflect that user's reading interests: ceteris paribus, the longer the 'active-time' (reading time), the more interested the user. 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 readers. 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 *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

Recently, the availability of event-based news data (describing some one clicked some news at sometime and closed it at sometime) , the recommendation task, which should predict whether a user will click an unseen news or not based on his or her click history, is especially preferred by both researchers and practitioners. In addition, we argue that a user's active-time on a news article can be, and probably should be, predicted simultaneously. Hence, we build a unified model to solve these two tasks.

Suppose a dataset for training consists of n instances $(\mathbf{x}, y, t)$, where $\mathbf{x}$ records a pair of user and news, y (equals 0 or 1) is the associated label indicating user click behaviors (i.e., y=1 means the user clicked the news, and y=0 otherwise), and t writes down the active-time. To be specific, $\mathbf{x}$ contains two parts: $user$ and $news$, where the relevant info, such as user's click history, news titles, contents, categories, and the related time attributes, are involved. It is worth pointing out that the news recommendation and active-time prediction both need to model users and news, therefore a multi-task learning framework, $[click\_probability, time\_interval] = Double\_Task\_Model(\mathbf{x})$, which exploits the commonalities across different tasks, could be constructed. Note that the higher the click-probability, the more likely the news recommendation. Here the continuous value of active-time would be discretized into time-intervals (e.g., 100-110 seconds), thus the prediction of active-time could be considered as a time-interval classification problem. It turns out that such an aggressive simplification works well in our experiments, suggesting that a crude estimation of active-time is usually good enough.

3 The Method: HyperNews

Fig. 1a shows the general 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 consisting of 'news-title', 'news-content', 'news-category', 'publish-time', 'click-time', and 'active-time'. For news recommendation, we design a timeliness module on the news embedding to emphasize the impact of 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). For active-time prediction, we employ a softmax classifier to estimate a

time-interval. Note that it would be necessary to use two different encoders with separate learn-able attention networks for explicit and implicit embeddings, because they have different impacts to our tasks.

3.1 News Encoder

News encoder (Fig. 1b) is used to learn news embedding from various news attributes (i.e., title, content and categories). We further divide these attributes into two parts: explicit and implicit info. The explicit corresponds to news title and its categories that users can easily obtain before clicking, and the implicit represents news contents that users would read after clicking. The reason why we differentiate them is that they should have different effects on click-probability and active-time. Intuitively, the explicit info has a greater impact on click-probability and the implicit info otherwise. Once the embeddings of such two parts are achieved, an attention unit could be further applied to adaptively learn their combinations for different tasks.

Explicit Embedding. The explicit embedding unit contains two components. The first is to embed news title. Specifically, we adopt a typical CNN-based model with a sequence of words as inputs, which are represented as a word embedding based matrix $\mathbf{w}_{1:n} = [\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_n] \in \mathbb{R}^{d \times n}$, where $\mathbf{w}_i \in \mathbb{R}^{d \times 1}$ is the i-th word of the news title, $d = 100$ denotes the dimension of word embedding, and n represents the number of words in news title. Then the matrix $\mathbf{w}_{1:n}$ will be applied with a convolution operator via a filter $\mathbf{H} \in \mathbb{R}^{d \times l}$, where $l = 3$ ($l < n$) means the window size. In regard to each sub-matrix $\mathbf{w}_{i:i+l-1}$, we can get a feature c_i by:

$$c_i = f(\mathbf{H} * \mathbf{w}_{i:i+l-1} + b), \tag{1}$$

where $*$ is the convolution operator, b is a bias term, and f is the Relu activation function. With all the possible positions of the matrix $\mathbf{w}_{1:n}$ crossed, there yields a feature map:

$$\boldsymbol{c} = [c_1, c_2, ..., c_{n-l+1}]^T, \tag{2}$$

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

$$\widetilde{c} = \max\{\boldsymbol{c}\} = \max\{c_1, c_2, \cdots, c_{n-l+1}\}. \tag{3}$$

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

$$\mathbf{e}_h = [\widetilde{c}_1, \widetilde{c}_2, \cdots, \widetilde{c}_m]^T. \tag{4}$$

With respect to the second component, it is to embed news categories. The reason why we take them into considerations is that users usually care about their interested categories and barely click news with categories they are not interested in. Regarding this part, the input for each category is a unique id, which is randomly initialized with a low-dimensional (i.e., 100) vector $\mathbf{e}_c$. After that, a dense layer is employed to learn category's hidden representation formulated with:

$$\mathbf{e}'_c = \tanh(\mathbf{W}_c \times \mathbf{e}_c + \mathbf{b}_c), \tag{5}$$

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

$$\alpha_i^c = \mathbf{q}_c^T \times \tanh(\mathbf{V}_c \times \mathbf{e}'_{c_i} + \mathbf{b}'_c), (i = 1, 2, 3), \tag{6}$$

$$\widetilde{\alpha}_i^c = \frac{\exp(\alpha_i^c)}{\sum_j \exp(\alpha_j^c)}, (j = 1, 2, 3), \tag{7}$$

and

$$\widetilde{\mathbf{e}}_c = \sum\nolimits_i \widetilde{\alpha}_i^c \mathbf{e}'_{c_i}, (i = 1, 2, 3), \tag{8}$$

where $\mathbf{q}_c$, $\mathbf{V}_c$ and $\mathbf{b}'_c$ are parameters of the attention unit.

To the end, representations of news title and its categories are concatenated together, which are fed into a one-layer fully connected neural network (FNN) for a lower dimensional embedding as the representation of the explicit info:

$$\mathbf{e}_{N_e} = \mathrm{FNN}([\mathbf{e}_h; \widetilde{\mathbf{e}}_c]). \tag{9}$$

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*[1] news dataset, based on which we could infer a probabilistic representation for any news content. Mark news content as D, then the topic representation (dimensions=100), treated as a part of the input, is listed:

$$\mathbf{e}_t = \mathrm{LDA}(D). \tag{10}$$

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

$$\mathbf{e}_d = \mathrm{Doc2Vec}(D). \tag{11}$$

Note that by means of doc2vec, the semantic representation of news content could be further enriched. Besides, the above two steps can both be processed in advance and wouldn't increase model's computations.

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 an important feature of 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 (i.e., 100) vector $\mathbf{e}_l$. It would be randomly initialized and then go through a fully-connected layer to become the final representation:

$$\mathbf{e}'_l = \tanh(\mathbf{W}_l \times \mathbf{e}_l + \mathbf{b}_l), \tag{12}$$

where $\mathbf{e}_l$, $\mathbf{W}_l$ and $\mathbf{b}_l$ are to-be-learned parameters.

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

$$\mathbf{e}_{N_i} = \mathrm{FNN}([\mathbf{e}_t; \mathbf{e}_d; \mathbf{e}'_l]). \tag{13}$$

Attention Unit. Intuitively, explicit and implicit infos 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 α_1^n and α_2^n, we have:

$$\alpha_1^n = \mathbf{q}_n^T \times \tanh(\mathbf{V}_n \times \mathbf{e}_{N_e} + \mathbf{b}_n), \tag{14}$$

$$\alpha_2^n = \mathbf{q}_n^T \times \tanh(\mathbf{V}_n \times \mathbf{e}_{N_i} + \mathbf{b}_n), \tag{15}$$

and

$$\widetilde{\alpha}_i^n = \frac{\exp(\alpha_i^n)}{\exp(\alpha_1^n) + \exp(\alpha_2^n)}, (i = 1, 2), \tag{16}$$

where $\mathbf{q}_n$, $\mathbf{V}_n$ and $\mathbf{b}_n$ are learnable parameters. Afterwards, the news embedding is calculated as:

$$\mathbf{e} = \widetilde{\alpha}_1^n \mathbf{e}_{N_e} + \widetilde{\alpha}_2^n \mathbf{e}_{N_i}. \tag{17}$$

Note that in the HyperNews model, we actually would have three different sets of values for those parameters ($\mathbf{q}_n$, $\mathbf{V}_n$, $\mathbf{b}_n$) which would lead to three separate news encodes $\mathbf{e}^p$, $\mathbf{e}^t$ and $\mathbf{e}^u$ for click-probability, active-time, and user representation respectively (see Fig. 1a).

3.2 User Encoder

The user encoder is to learn user representation in accordance with user's click history. Currently, there are two ways which have been proved effective. To be specific, the RNN networks like GRU and LSTM are skilled at encoding sequential data while attention network is better to distinguish different importance of the clicked news articles. In this paper, we choose the second manner because the recently read articles are not strictly continuous sequences (there usually exist a long interval between two clicks). Denote the weight of the i-th clicked news as α_i^u, then we could arrive at:

$$\alpha_i^u = \mathbf{q}_u^T \times \tanh(\mathbf{V}_u \times \mathbf{e}_i^u + \mathbf{b}_u), (i = 1, 2, \cdots, 30), \tag{18}$$

$$\widetilde{\alpha}_i^u = \frac{\exp(\alpha_i^u)}{\sum_j \exp(\alpha_j^u)}, \quad (j = 1, 2, \cdots, 30), \tag{19}$$

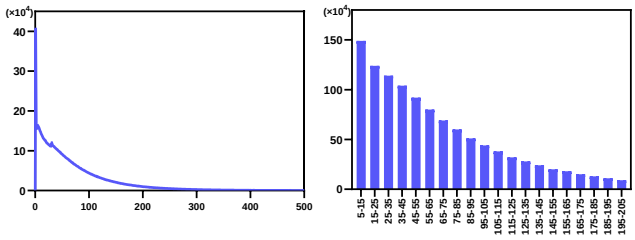
and

$$\mathbf{e}_U = \sum\nolimits_i \widetilde{\alpha}_i^u \mathbf{e}_i^u, \quad (i = 1, 2, \cdots, 30), \tag{20}$$

where $\mathbf{q}_u$, $\mathbf{V}_u$ and $\mathbf{b}_u$ are the parameters of attention unit to be learned, and $\mathbf{e}_U$ is the final user embedding. Note that we collect the recently clicked 30 news articles to model the user dynamic interests.

3.3 Click Probability

The freshness of a news article for a user, defined as the elapsed time between its 'publish-time' and the user's 'click-time', is pretty important for personalized news recommendation, therefore we design a *timeliness* module (Fig. 1a) to refine the embedding from the news encoder. Specifically, we divide the continuous time according to a variable length interval $[1h, 2h, \cdots, 1d, 2d, \cdots, 1m, 2m, \cdots]$ to do with its inhomogeneous variations, where h, d and m correspond to hour, day and month respectively. For example, if it is 30 minutes, we assign an id=1; if it's 1.5 hours (between $1h$ and $2h$), we assign an id=2; and so forth. For each interval id,



(a) #events vs. active-time. (b) #events vs. time-intervals

Figure 2: The distribution of events in the *Adressa* news dataset.

a low-dimensional (i.e., 200) vector $\mathbf{e}_f$ could be randomly initialized and then forwarded to a dense layer:

$$\mathbf{e}'_f = \tanh(\mathbf{W}_f \times \mathbf{e}_f + \mathbf{b}_f), \tag{21}$$

where $\mathbf{e}_f$, $\mathbf{W}_f$ and $\mathbf{b}_f$ are parameters to be learned. In what follows, a new time-enriched news embedding for news recommendation can be reached via:

$$\mathbf{e}_N = \mathbf{e}^p \odot \mathbf{e}'_f, \tag{22}$$

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

$$\hat{\mathbf{y}}_\mathbf{p} = \text{sigmoid}\left(\mathbf{e}_U^T \times \mathbf{e}_N\right), \tag{23}$$

based on which a news recommendation system could offer a user with the top-K (e.g., K=20) largest valued news articles.

3.4 Active-Time Prediction

Fitting an accurate value for active-time is a challenging task and in fact not that necessary. For example, usually there is no obvious difference between 50 and 60 seconds that a user has spent on a clicked news. Thus we transfer the time regression task into a 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's a long-tailed distribution; then we partition the time range $[5, 205]$ equally into 20 time intervals (Fig. 2b). The reasons why we conduct this division are twofolds: (1) the events in $[5, 205]$ accounts for about 90% which covers most of the cases; (2) the active-time fewer than 5 seconds usually indicates abnormal events. Note that the 20 time intervals are numbered $1, 2, \cdots, 20$ as class labels.

To the end, we combine the news embedding and user embedding as $\mathbf{e}_T = [\mathbf{e}_t; \mathbf{e}_U]$, and then utilize a softmax classifier to calculate the probabilities over the 20 time intervals:

$$\hat{\mathbf{y}}_t = \text{softmax}(\mathbf{W}_t \times \mathbf{e}_T + \mathbf{b}_t), \tag{24}$$

where $\mathbf{W}_t$ and $\mathbf{b}_t$ are classifier's parameters. Afterwards, we could conduct the active-time prediction by selecting the most likely time interval corresponding to the largest value of $\hat{\mathbf{y}}_t$.

3.5 Model Training

Similar to Ref. [Zhu *et al.*, 2019], we also regard the observed click events as positive samples and not observed ones as negative samples. In what follows, we denote a training sample

as $\mathbf{X} = (\{\mathbf{x}_1, \cdots, \mathbf{x}_r\}, \mathbf{x}_{r+1}, \mathbf{y}_p, \mathbf{y}_t)$, where $\{\mathbf{x}_1, \cdots, \mathbf{x}_r\}$ represents a user's recently browsed r news, $\mathbf{x}_{r+1}$ is a candidate news, $\mathbf{y}_p$ is click label (1 for clicked and 0 otherwise), $\mathbf{y}_t$ is time label (a one-hot vector for interval id, invalid for negative samples). Since $\hat{\mathbf{y}}_p$ and $\hat{\mathbf{y}}_t$ are the outputs of our HyperNews model, the double-task oriented loss functions could be designed respectively as below:

$$\mathbf{L}_p = -\left\{ \sum_{\mathbf{X} \in \mathcal{S}^+} \mathbf{y}_p \log(\hat{\mathbf{y}}_p) + \sum_{\mathbf{X} \in \mathcal{S}^-} (1 - \mathbf{y}_p) \log((1 - \hat{\mathbf{y}}_p)) \right\}, \tag{25}$$

and

$$\mathbf{L}_t = -\sum_{\mathbf{X} \in \mathcal{S}^*} \mathbf{y}_t \log(\hat{\mathbf{y}}_t), \tag{26}$$

where S^+ is the positive sample set, S^- is the negative sample set and S^* is the positive sample set containing the active-time attribute.

As Fig. 2b clearly says, the active-time prediction task will be a severely class-unbalanced one, hence we resort to a class-balanced method based on estimating the effective size of a class [Cui *et al.*, 2019]. Specifically, it is defined as:

$$\mathbf{E}_m = (1 - \beta^m)/(1 - \beta), \tag{27}$$

where m is the actual number of samples in the class and β is a hyperparameter, usually a value close to 1 (e.g., 0.99, 0.999, etc.). Based on the above analysis, the class-balanced variant of $\mathbf{L}_t$ can be written into:

$$\mathbf{L}'_t = -\sum_{\mathbf{X} \in \mathcal{S}^*} \frac{1}{\mathbf{E}_{m_{\mathbf{y}_t}}} \mathbf{y}_t \log(\hat{\mathbf{y}}_t), \tag{28}$$

where $m_{\mathbf{y}_t}$ is the actual number of samples in class $\mathbf{y}_t$.

Taking $\mathbf{L}_p$ (Eq. (25)) and $\mathbf{L}'_t$ (Eq. (28)) into considerations, the overall loss function to minimize is:

$$\mathbf{L} = \mathbf{L}_p + \lambda \mathbf{L}'_t, \tag{29}$$

where λ is a preset 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 verify the effectiveness of HyperNews by comparing it with several competitive baselines.

4.1 Datasets

Adressa[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 ever 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.

Table 1: The descriptive statistics of our datasets. "words/title": average words per news title; "words/content": average words per news content; "events ('active-time')": events with 'active-time'.

Number	*Adressa-1week*	*Adressa-4week*
#users	601,215	1,540,168
#news	17,692	37,067
#words/title	6.63	6.50
#words/content	552.15	530.86
#categories	245	252
#events	3,123,261	11,584,797
#events ('active-time')	1,062,793	3,951,288

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

4.2 Competitors and Metrics

To evaluate the recommendation performance of our HyperNews, several state-of-the-art competitors are invited, including: **LibFM**[2] [Rendle, 2012], **DeepFM**[3] [Guo *et al.*, 2017], **DSSM**[4] [Huang *et al.*, 2013], **Wide&Deep**[5] [Cheng *et al.*, 2016], **LSTUR**[6] [An *et al.*, 2019], and **DAN**[7] [Zhu *et al.*, 2019]. Moreover, since HyperNews is a double-task model, we also measure the second active-time prediction's performance by comparing with **HyperNews$_{\text{NoRecom}}$**: the HyperNews model without the news recommendation part, i.e., a single-task active-time prediction model.

In terms of metrics, we employ the popular AUC [Fawcett, 2006] and F_1 [Li *et al.*, 2008] scores to measure the performance of news recommendation, and just F_1 to measure that of active-time prediction.

The source code of **HyperNews** and the processed datasets will be made available online for the reproducibility of our experiments.

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*

[1] http://reclab.idi.ntnu.no/dataset/

[2] https://github.com/srendle/libfm

[3] https://github.com/ChenglongChen/tensorflow-DeepFM

[4] https://github.com/InsaneLife/dssm

[5] https://github.com/kaitolucifer/wide-and-deep-learning-keras

[6] The source code is kindly provided by the authors.

[7] https://github.com/zhuqiannan/dan-for-news-recommendation

Table 2: News recommendation performance (AUC and F_1).

Methods	*Adressa-1week*		*Adressa-4week*	
	AUC	F_1	AUC	F_1
LibFM	0.7025	0.6953	0.6781	0.6594
DeepFM	0.7358	0.7288	0.7054	0.6821
Wide&Deep	0.7415	0.7244	0.7133	0.6909
DSSM	0.7511	0.7232	0.7269	0.6932
LSTUR	0.8077	0.7558	0.7725	0.7261
DAN	0.8234	0.7676	0.7864	0.7397
HyperNews$_{\text{NoPred}}$	0.8377	0.7798	0.7913	0.7452
HyperNews	**0.8661**	**0.8027**	**0.8264**	**0.7754**

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

Methods	*Adressa-1week*	*Adressa-4week*
HyperNews$_{\text{NoRecom}}$	0.8398	0.7843
HyperNews	**0.8946**	**0.8590**

al., 2019; Zhu *et al.*, 2019], for the former group, we just utilize the concatenation of news title, news content and its categories as inputs; whereas for the latter group, we make use of the configurations as described in the method's corresponding papers. The baseline methods' parameters have been tuned on *Adressa-1week* to get competitive results and then applied for both datasets. Similarly, the parameters of HyperNews have been configured as $\lambda = 0.8$ and $\beta = 0.99999$ (see Section 4.5).

4.4 Results

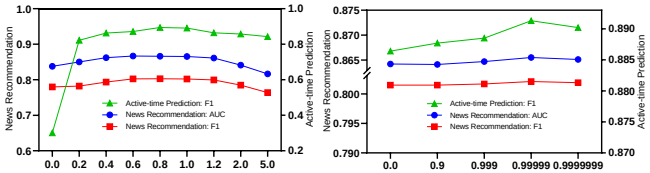
Table 2 shows the AUC and F_1 results of different recommendation methods. Note that HyperNews$_{\text{NoPred}}$ is an adapted model with the active-time prediction part removed from HyperNews, i.e., a single-task news recommendation methods. Obviously, our multi-task learning approach, the complete HyperNews system, defeats all the other baseline methods, which verifies its effectiveness.

Comparing the general-purpose models with the news-oriented ones, the latter usually generate higher scores than the former because the general recommendation models (LibFM, DeepFM, Wide&Deep, and DSSM) only resort to the basic texts (title, content, categories) ignoring the task-specific information utilized in the latter (LSTUR, DAN, and HyperNews). It is worth pointing out that HyperNews shows a clear performance gain over HyperNews$_{\text{NoPred}}$, which tells us that the active-time prediction task does help the news recommendation task through their joint learning of the neural network.

Moreover, the active-time prediction results are shown in Table 3, where HyperNews$_{\text{NoRecom}}$ is a variant of HyperNews with the news recommendation part removed. Clearly, the F_1-scores of the double-task model HyperNews are higher than those of the single-task model HyperNews$_{\text{NoRecom}}$, which tells us that the news recommendation task could in turn reinforce the active-time prediction task.

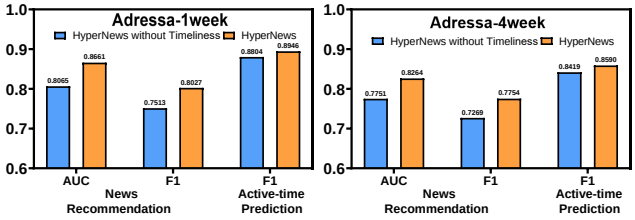
4.5 Hyperparameter Analysis

HyperNews contains two hyperparameters: λ that controls the balance between news recommendation and active-time prediction, and β that controls a class's effective size [Cui



(a) λ (b) β

Figure 3: HyperNews with various λ or β values on *Adressa-1week*.



(a) On *Adressa-1week*. (b) On *Adressa-4week*.

Figure 4: HyperNews with/without the timeliness module.

et al., 2019]. Keeping β=0.99999 fixed, we vary λ from 0 to 5 and plot the experimental results in Fig. 3a. It can be observed that on *Adressa-1week*, no matter what task it is for and which performance measure is used, λ=0.8 always leads to competitive results. Similarly, keeping λ=0.8 fixed, Fig. 3b reveals that when β=0.99999 (among $\{0, 0.9, 0.999, 0.99999, 0.9999999\}$), HyperNews yields the best results for these two tasks. Note that when β=0, Eq. (28) would be equal to Eq. (26). Therefore, λ and β are finally set as 0.8 and 0.99999 respectively, on both datasets *Adressa-1week* and *Adressa-4week*.

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 verifies the important 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

[An *et al.*, 2019] Mingxiao An, Fangzhao Wu, Chuhan Wu, Kun Zhang, Zheng Liu, and Xing Xie. Neural news recommendation with long- and short-term user representations. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 336–345, Florence, Italy, July 28-August 2, 2019.

[Blei *et al.*, 2003] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.

[Cheng *et al.*, 2016] HengTze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pages 7–10, Boston, MA, USA, September 15, 2016.

[Cui *et al.*, 2019] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge J. Belongie. Class-balanced loss based on effective number of samples. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 9268–9277, Long Beach, CA, USA, June 16-20, 2019.

[Fawcett, 2006] Tom Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006.

[Gulla *et al.*, 2017] Jon Atle Gulla, Lemei Zhang, Peng Liu, Özlem Özgöbek, and Xiaomeng Su. The adressa dataset for news recommendation. In *Proceedings of the International Conference on Web Intelligence*, pages 1042–1048, Leipzig, Germany, August 23-26, 2017.

[Guo *et al.*, 2017] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. DeepFM: A factorization machine based neural network for CTR prediction. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 1725–1731, Melbourne, Australia, August 19-25, 2017.

[Huang *et al.*, 2013] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry P. Heck. Learning deep structured semantic models for web search using clickthrough data. In *The 22nd ACM International Conference on Information and Knowledge Management*, pages 2333–2338, San Francisco, CA, USA, October 27-November 1, 2013.

[Le and Mikolov, 2014] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31th International Conference on Machine Learning*, pages 1188–1196, Beijing, China, June 21-26, 2014.

[Li *et al.*, 2008] Xiao Li, Ye-Yi Wang, and Alex Acero. Learning query intent from regularized click graphs. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 339–346, Singapore, July 20-24, 2008.

[Lian *et al.*, 2018] Jianxun Lian, Fuzheng Zhang, Xing Xie, and Guangzhong Sun. Towards better representation learning for personalized news recommendation: a multi-channel deep fusion approach. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pages 3805–3811, Stockholm, Sweden, July 13-19, 2018.

[Morales *et al.*, 2012] Gianmarco De Francisci Morales, Aristides Gionis, and Claudio Lucchese. From chatter to headlines: harnessing the real-time web for personalized news recommendation. In *Proceedings of the Fifth International Conference on Web Search and Web Data Mining*, pages 153–162, Seattle, WA, USA, February 8-12, 2012.

[Okura *et al.*, 2017] Shumpei Okura, Yukihiro Tagami, Shingo Ono, and Akira Tajima. Embedding-based news recommendation for millions of users. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1933–1942, Halifax, NS, Canada, August 13-17, 2017.

[Phelan *et al.*, 2011] Owen Phelan, Kevin McCarthy, Mike Bennett, and Barry Smyth. Terms of a feather: Content-based news recommendation and discovery using twitter. In *Advances in Information Retrieval-33rd European Conference on IR Research*, pages 448–459, Dublin, Ireland, April 18-21, 2011.

[Rendle, 2012] Steffen Rendle. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology*, 3(3):57:1–57:22, 2012.

[Wang *et al.*, 2018] Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. DKN: deep knowledge-aware network for news recommendation. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 1835–1844, Lyon, France, April 23-27, 2018.

[Wu *et al.*, 2019a] Chuhan Wu, Fangzhao Wu, Mingxiao An, Jianqiang Huang, Yongfeng Huang, and Xing Xie. Neural news recommendation with attentive multi-view learning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 3863–3869, Macao, China, August 10-16, 2019.

[Wu *et al.*, 2019b] Chuhan Wu, Fangzhao Wu, Mingxiao An, Jianqiang Huang, Yongfeng Huang, and Xing Xie. NPA: neural news recommendation with personalized attention. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2576–2584, Anchorage, AK, USA, August 4-8, 2019.

[Wu *et al.*, 2019c] Chuhan Wu, Fangzhao Wu, Mingxiao An, Yongfeng Huang, and Xing Xie. Neural news recommendation with topic-aware news representation. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 1154–1159, Florence, Italy, July 28-August 2, 2019.

[Zhu *et al.*, 2019] Qianan Zhu, Xiaofei Zhou, Zeliang Song, Jianlong Tan, and Li Guo. DAN: deep attention neural network for news recommendation. In *The Thirty-Third AAAI Conference on Artificial Intelligence*, pages 5973–5980, Honolulu, Hawaii, USA, January 27-February 1, 2019.