Octopus: Comprehensive and Elastic User Representation for the Generation of Recommendation Candidates

Zheng Liu
Microsoft Research Asia
Beijing, China
zhengliu@microsoft.com

Jianxun Lian
Microsoft Research Asia
Beijing, China
jialia@microsoft.com

Junhan Yang
USTC
Hefei, China
v-juny@microsoft.com

Defu Lian
USTC
Hefei, China
liandefu@ustc.edu.cn

Xing Xie
Microsoft Research Asia
Beijing, China
xingx@microsoft.com

ABSTRACT

Candidate generation is a critical task for recommendation system, which is technically challenging from two perspectives. On the one hand, recommendation system requires the comprehensive inclusion of user's interested candidates, yet typical deep user modeling approaches would represent each user as an onefold vector, which is hard to capture user's diverse interests. On the other hand, for the sake of practicability, the candidate generation process needs to be both accurate and efficient. Although existing "multi-channel structures", like memory networks, are more capable of representing user's diverse interests, they may bring in substantial irrelevant candidates and lead to rapid growth of temporal cost. As a result, it remains a tough issue to comprehensively acquire user's interested items in a practical way.

In this work, a novel personalized candidate generation paradigm, **Octopus**, is proposed, which is remarkable for its comprehensiveness and elasticity. Similar with those conventional "multi-channel structures", Octopus also generates multiple vectors for the comprehensive representation of user's diverse interests. However, Octopus' representation functions are formulated in a highly elastic way, whose scale and type are adaptively determined based on each user's individual background. Therefore, it will not only identify user's interested items comprehensively, but also rule out irrelevant candidates and help to maintain a feasible running cost. Extensive experiments are conducted with both industrial and publicly available datasets, where the effectiveness of Octopus is verified in comparison with the state-of-the-art baseline approaches.

KEYWORDS

Recommendation System, Candidate Generation, User Representation and User Modeling

ACM Reference Format:
Zheng Liu, Jianxun Lian, Junhan Yang, Defu Lian, and Xing Xie. 2020. Octopus: Comprehensive and Elastic User Representation for the Generation of Recommendation Candidates. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20), July 25–30, 2020, Virtual Event, China.* ACM, – , – , – , 10 pages. https://doi.org/10.1145/3397271.3401088

1 INTRODUCTION

Candidate generation is one of the fundamental problems in recommendation system, where candidates relevant to user interest must be extracted in realtime from a tremendous pool of items. The mainstream candidate generation methods are based on similarity search. Particularly, such kind of approaches would represent user and item in the same space, meanwhile user and item's relevance is reflected by their representation similarity. Therefore, user's nearest neighbours in the representation space turn out to be the most relevant items, which enables the candidates to be quickly acquired via KNN search.

Back in the early days, people used to exploit raw features intensively; e.g., users could be represented by a set of keywords (or tags), where items associated with the same keywords were regarded as proper candidates. Due to the coarse-granularity of raw features, it is hard to measure user and item's relevance precisely. In recent years, increasing attention has been paid to representations learned by deep neural networks. Typical methods include (but not be limited to) YouTube-DNN [5], DSSM [8, 11], where user and item are encoded via deep neural networks and their relevance is reflected by their closeness in the common latent space. However, the conventional approaches would assign each user with one single vector, which is hard to represent a user comprehensively when her interest is diversified. As a result, only a small portion of user's interested items can be obtained with them.

Some of the recent "multi-channel structures" seem to be promising solutions towards the above problem (each channel is virtually a user representation function), e.g., memory networks [4, 23]. With the deployment of multiple channels, a user can be modeled from different perspectives and her diverse interests can be jointly captured with different representations. However, because of the rigid formulation of channels, conventional structures may introduce irrelevant candidates and bring about unnecessary running cost when applied to candidate generation. In fact, these defects can be

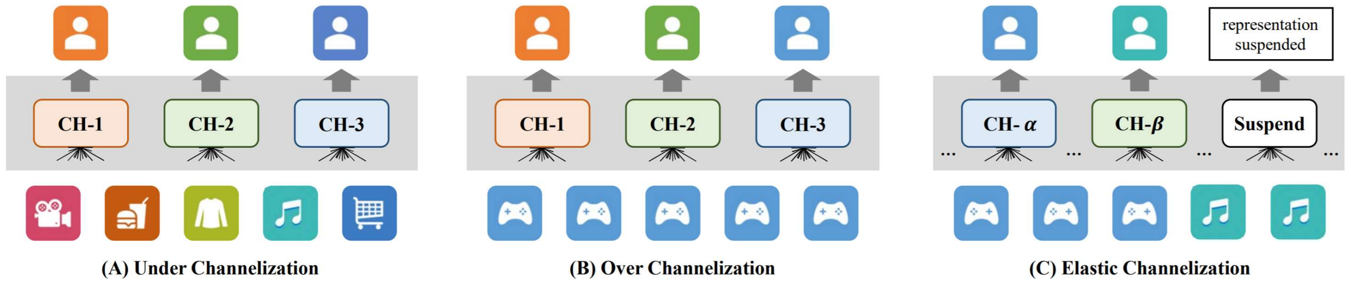


Figure 1: Illustration of Toy Examples. (For each 3-layer sub-figure, the bottom layer stands for user history, the mid-layer represents the multi-channel structure, and the top-layer demonstrates the user representations.)

traced back to a common underlying problem, which is referred as the "**dilemma of over/under channelization**" in our discussion.

• **Under Channelization**. A multi-channel structure could be assigned with merely a small number of channels, whose expressiveness is highly limited due to its insufficient deployment of channels. In that case, it will still be difficult to represent user's diverse interests with fine-granularity.

• **Over Channelization**. Once a multi-channel structure employs a sufficient number of channels, it will probably become capable of representing user's diverse interests. However, it is accompanied with three potential defects: **noise**, **attenuation** and **inefficiency**, which severely harms its candidate generation performance. Particularly, it may introduce a lot of noisy candidates, as many of the deployed channels are potentially irrelevant to user's actual interest. Besides, given a predefined size of candidate set, the existence of noisy candidates will also cut down the quota of those relevant ones, which prevents the comprehensive inclusion of user's interested items. Even worse, the computation cost of candidate generation grows linearly with the number of channels, which will easily become prohibitive if too many channels are to be deployed.

In this place, a toy example is presented for better illustration.

Example 1.1. Suppose a three-channel structure is deployed for user presentation, shown as Fig. 1 (A) and (B).

Now let there be a user who has highly diversified interests across a variety of subjects (shown as Fig. 1 (A)). Apparently, the deployment of three channels is hard to generate a fine-grained representation for each individual interest. Instead, it may merely capture a fraction of user interests with coarse-granularity.

In the second case, we come across another user whose interest is concentrated on one single subject (shown as Fig. 1 (B)). The homogeneity of her interest makes the deployment of three channels redundant: although one channel (e.g., Channel 3) is well aligned with user interest and gives rise to a meaningful representation, the rest ones have little to do the user's actual interests, whose representations are probably inaccurate. Such inaccurate user representations which will bring in irrelevant candidates (noise), cut down the quota of those relevant candidates (attenuation), and incur unnecessary running cost (inefficiency).

In this work, a novel multi-channel structure, **Octopus**, is proposed to overcome the dilemma of over/under channelization. The remarkable property about Octopus is its **Comprehensiveness** and **Elasticity**, where each user's diverse interests can be comprehensively captured with the elastic formulation of channels. The underlying idea of Octopus is quite intuitive. First of all, we will employ a sufficient amount of channels, with which all kinds of user interests (i.e., the interests exist in the whole crowd) can be precisely captured. Then, for each specific user, we may simply "activate" a fraction of the channels which are relevant to her individual background. For example, given a user with two types of interests (illustrated as Fig. 1 (C)), only the two closely related channels will be activated for user representation. The rest channels, which are corresponding to other types of user interests, are suspended and no user representations are generated from them. In this way, user's diverse interests are comprehensively captured with purely the close-related channels. Therefore, it 1) rules out noise, 2) prevents attenuation and 3) incurs no needless running cost. On top of this fundamental framework (referred as **Elastic Archive Network**), the following operations are carried out to improve the quality of candidate generation.

• First of all, orthogonality is introduced to the deployed channels such that redundancy between the channels can be largely reduced; meanwhile, diversified user interests can be extracted more effectively.

• Secondly, the grouped attentive aggregation is utilized, where each user representation is generated purely with its closely related user behaviors. (Back to the example in Fig. 1 (C), the first user representation is generated purely with the behaviors on games, while the second user representation is generated based on the behaviors on music.) Such a treatment helps to generate more "purified representations"; whereby, each user interest can be better highlighted.

• Thirdly, because of the existence of multiple user representations, candidate integration strategies are proposed so that the neighbourhood items from different user representations can be assembled for a better candidate set.

To summarize, the following contributions are made in this work.

- We propose Octopus[1], a novel multi-channel structure which comprehensively captures user's diverse interests in a highly elastic manner.

[1]This project is named as Octopus for two reasons: 1) the deployment of multiple channels is analogous to Octopus' multi-armed structure, 2) the elastic channel deployment is analogous to Octopus capability of dropping some of its arms when necessary.

- On top of the proposed elastic framework, three operations are explored for better channel deployment, user representation and candidate integration.
- Extensive experimental studies are carried out with both industrial and publicly available datasets, where Octopus achieves substantial improvements over state-of-the-art baseline methods.

The rest of the paper is organized as follows. We will first review the related works in Section 2 and formulate our problem in Section 3. Afterwards, we will discuss our technical designs in Section 4 and analyze the experimental studies in Section 5. Finally, we will conclude our work and discuss about the future work in Section 6.

2 RELATED WORK

The related works are reviewed from two perspectives: 1) deep recommendation system, and 2) personalized candidate generation.

2.1 Deep Recommendation System

Recommendation system powered by deep learning has been intensively applied to various kinds of web services, such as e-commerce, online advertisement and personalized content feed [1, 16, 18, 29]. Generally speaking, deep learning techniques contribute to recommendation system in two ways. First of all, thanks to the superior representation capacity of deep neural networks, compact and discriminative features can be automatically learned from raw data, e.g., text representation learned superior language models [3, 6, 7, 19], and relational features extracted by GNN and graph embedding [9, 26, 27]. Secondly, with sophisticated network structures, user's complex behavioral patterns can be modeled with higher precision. For example, with recurrent neural networks [10], user's temporal interest can be effectively represented; leveraging memory networks [4, 20, 23], user's diverse interests can be better captured; and on top of attention mechanisms [12, 29], mutual interaction and long-range dependency can be established for user's historical behaviors.

It should be noted that most of today's deep recommendation algorithms are mainly used for the "ranking step", i.e., the final selection of recommendation items from a small candidate set; in contrast, relatively limited progress is made in candidate generation.

2.2 Personalized Candidate Generation

Personalized candidate generation is the first step of making recommendations. Unlike the subsequent ranking operation, it requires the satisfaction of both precision and efficiency: proper candidates must be selected from a huge number of items in realtime. As a result, mainstream algorithms would transfer the candidate generation problem to a similarity-search problem. On top of those well-developed searching paradigms, such as inverted-index [14], KNN-search [25], Maximum Inner Product Search (MIPS) [13] and binarized representations [17], candidates can be generated with tremendous speed.

The mainstream candidate generation algorithms have two common prerequisites. Firstly, user and item must follow the same way of representation, e.g., keywords from a common collection, or vectors within the same latent space. Secondly, user and item's semantic relationship must be reflected by their representation similarity; e.g., the co-occurrence of keywords, or the distance between two vectors. Apparently, the candidate quality is largely influenced by the capacity of representation. Back in the early days, people would resort to the direct comparison of raw-features [2, 28], e.g., similarity in raw texts. Due to the coarse-granularity of raw features, it is hard to identify user's interested items precisely. In recent years, increasing attention is paid to the representations learned by deep neural networks, e.g., [5, 8, 30]. However, conventional representation methods would generate one single vector for each individual user, which is hard to capture user's diverse interests. Although existing multi-channel structures [4, 20] will generate diversified user representations, the dilemma of over/under channelization will severely restrict the performance in reality.

Apart from the mainstream way of candidate generation, some alternative approaches are proposed recently [31, 32]. The underlying idea is quite similar with hierarchical softmax [22], where items are organized as a tree via hierarchical clustering (e.g., K-means); in addition, a deep ranking model (e.g., DEIN [29]) is used, which looks for the candidates from the root node to the leaves via beam-search. With this paradigm, the candidate generation will take $O(M \log N)$ rounds of deep model inference (M: beam-width, also the number of candidates; N: the number of whole items; $\log N$: the tree depth). The comparatively expensive running cost turns out to be the major limitation in contrast to those similarity-search based methods, as k-NN search merely involves simple distance calculations, and the latest approaches (e.g., [21, 25]) can be as fast as O(log 1).

3 PROBLEM FORMULATION

In this work, we are considering the following classic scenario, where personalized candidates are acquired via similarity search.

• **Candidate Generation via Similarity Search.** Suppose each user u is associated with her historical items $\{x_1^u, ..., x_N^u\}$, and her interested item x_T^u at the current timestamp (referred as ground-truth item). Each item x is encoded as a latent representation θ_x. Now, we are looking for the user representation ϕ^u, such that the representation of ground-truth item ($\theta_{x_T^u}$) is within ϕ^u's nearest neighbours (w.r.t. a predefined distance function, e.g., Cosine):

$$\theta_{x_T^u} \in k\text{-NN}(\phi^u). \tag{1}$$

As mentioned before, a user may have diverse interests in reality; thus, multiple user representations $\Phi^u = \{\phi_1^u, ..., \phi_L^u\}$ can be generated, each of which is corresponding to a specific type of user interest. In this situation, the relationship in Eq. 1 is generalized accordingly, where $\theta_{x_T^u}$ needs to be included within the neighbourhood of one of the user representations:

$$\exists\, \phi_l^u \in \Phi^u : \ \theta_{x_T^u} \in k\text{-NN}(\phi_l^u). \tag{2}$$

• **Elastic Formulation.** Unlike the conventional practice where user representation is generated in a rigid form for all users, the following elastic formulation is proposed, which is adaptive to each individual user.

First of all, we will employ "**a family of representation functions**" $\mathbf{F} = \{f_i\}_{i=1}^M$. Such a function family is expected to be "over-complete" w.r.t. the inherent user interests: i.e., for any type of user interest, there will always be at least one function in $\mathbf{F}$, which is able to extract it from the complex user behaviors.

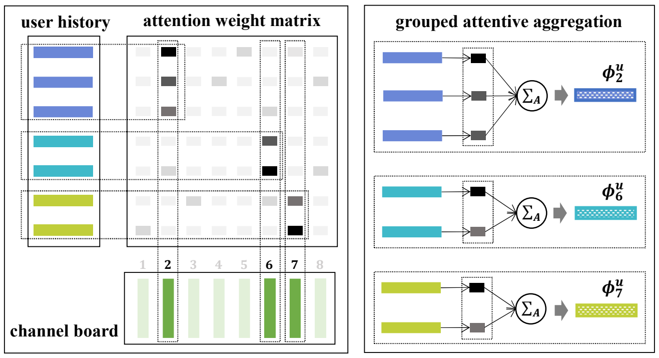


Figure 2: Framework Overview. Left box: channel activation; right box: user representation w.r.t. the activated channels.

Suppose such a function family is acquired. Then for each individual user, we will select a subset of representation functions $\mathbf{F}^u \subseteq \mathbf{F}$, which are closely related with user history. Now the user representations ($\mathbf{\Phi}^u$) will be generated based on the selection:

$$\mathbf{\Phi}^u \leftarrow \{\ \phi_l^u \mid \phi_l^u = f_l(u),\ \forall f_l \in \mathbf{F}^u\}. \tag{3}$$

Intuitively, a user with diverse background could activate a lot of representation functions, while a user with monotonous background may just require a few; thus, user interest can be comprehensively represented with little redundancy or inaccuracy. Meanwhile, users with different background will probably get highly differentiated subsets of functions, which best fit their own backgrounds.

4 THE OCTOPUS

In this part, we will first present the structure of Elastic Archive Network (which is the backbone of Octopus), how to generate diversified user representations with it, and how to learn it from data. Secondly, given the existence of multiple user representations, discussion is made on how to integrate their neighbourhood items for the best candidate set.

4.1 Elastic Archive Network

The comprehensive and elastic user representation is carried out with the Elastic Archive Network. The family of representation functions is substantialized as its "**channels**", each of which is to generate one user representation corresponding to a type of user interest. The selection step is carried out as "**channel activation**", meaning that a subset of the channels will be activated for a specific user. Then user behaviors will pass through the activated channels, which gives rise to the comprehensive representations of user's diverse interests. To realize the above purposes, each channel is equipped with two heads, 1) the grouping head, which is used to identify different kinds of user behaviors; and 2) the aggregation head, which is used to aggregate user behaviors into an representation vector.

4.1.1 Channel Deployment. Because of the existence of multiple channels, the channel coordination mechanism from [20] is utilized. Particularly, the grouping heads $\mathbf{H}^{\mathrm{g}}$ are initialized to be the

Algorithm 1: Training of Elastic Archive Network

```
1  Input: user history $\{\mathbf{\Theta}_{X^u}\}_{\mathbf{U}}$, all items' representations $\mathbf{\Theta}_X$.
2  Output: the Elastic Archive Network.
3  Initialize the deployment of channels $\mathbf{H}^{\mathrm{g}}$ as Eq. 4;
4  while not converge do
5      for each user u in mini-batch $\mathbf{U}_i$ do
6          get attention-weight matrix $\mathbf{A}_{N\times M}$ as Eq. 6;
7          get activated channels $\mathbf{C}^u$ as Eq. 7 and 8;
8          partition user behaviors into groups as Eq. 9;
9          get user representations $\mathbf{\Phi}^u$ as Eq. 10;
10         get targeted user representation $\phi_l^u$ via Eq. 11;
11         get distance with the ground-truth: $\mathrm{dist}(\theta_{x_T^u}, \phi_l^u)$;
12     get orthogonal regularization $\Omega$ as Eq. 5;
13     get overall training loss: $\sum_{\mathbf{U}_i} \mathrm{dist}(\theta_{x_T^u}, \phi_l^u)/|\mathbf{U}_i| + \lambda\Omega$;
14     back-propagate the gradients and update.
```

orthogonal basis of the whole items' representations $\mathbf{\Theta}_X$:

$$\mathbf{H}^{\mathrm{g}} \leftarrow \underset{\mathbf{H}^{\mathrm{g}}}{\operatorname{argmin}} \sum\nolimits_{\mathbf{\Theta}_X} \|\theta_x - \theta_x {\mathbf{H}^{\mathrm{g}}}^T \mathbf{H}^{\mathrm{g}}\|_2, \tag{4}$$

where θ_x is $1 \times d$ and $\mathbf{H}^{\mathrm{g}}$ is $M \times d$. By solving the above problem, we will get M latent vectors which can jointly represent all kinds of user behaviors. In addition, one regularization item Ω is adopted to maintain the orthogonality in subsequent training process:

$$\Omega \leftarrow \|\mathbf{H}^{\mathrm{g}} {\mathbf{H}^{\mathrm{g}}}^T - \mathbf{I}\|_F, \tag{5}$$

where $\mathbf{I}$ is the $d \times d$ identity matrix and $\|\cdot\|_F$ is the Frobenius norm.

4.1.2 Diversified User Representation. Given the encoded user history $\mathbf{\Theta}_{X^u}$ ($\mathbf{\Theta}_{X^u} = \{\theta_{x_1^u} ... \theta_{x_N^u}\}$), diversified user representations are generated by Octopus via two consecutive steps: channel activation and grouped attentive aggregation.

• **Channel Activation.** We will first let the user history $\mathbf{\Theta}_{\mathbf{X}^{\mathrm{u}}}$ attend to the grouping heads $\mathbf{H}^{\mathrm{g}}$, such that we may figure out each historical behavior's relevance to the channels. Particularly, the following attention-weight matrix will be computed:

$$\begin{aligned} \mathbf{A}_{N\times M} &= \mathbf{ATT}(\mathbf{\Theta}_{\mathbf{X}^{\mathrm{u}}}, \mathbf{H}^{\mathrm{g}}), \text{ as} \\ \mathbf{A}_{i,j} &= \mathbf{ATT}(\theta_{x_i^u}, h_j^g),\ 1 \le i \le N \text{ and } 1 \le j \le M, \end{aligned} \tag{6}$$

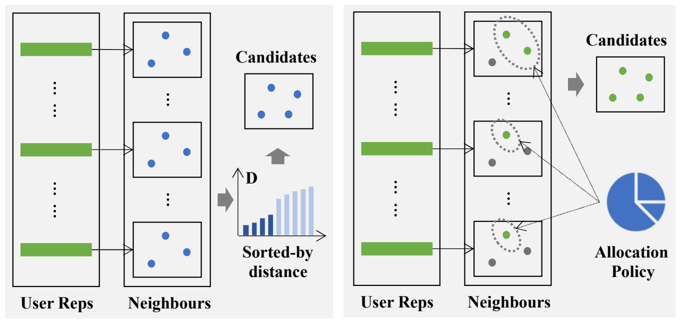
in which ATT() stands for the attention function[2]. Intuitively, if $\mathbf{A}_{i,j}$ is large, x_i^u will probably belong to the underlying user interest which the j-th channel c_j is corresponding to. Given the above matrix, the j-th channel is said to be activated by x_i^u, if h_j^g generates the largest attention weight of row $\mathbf{A}_{i,*}$:

$$h_j^g = \underset{h^g \in \mathbf{H}^{\mathrm{g}}}{\operatorname{argmax}} \mathbf{ATT}(\theta_{x_i^u}, h^g). \tag{7}$$

Besides, the activated channels of user u (denoted as $\mathbf{C}^u$) refer to the subset of channels, which are activated by at least one of the user's historical behaviors:

$$\mathbf{C}^u = \{c_j \mid \exists\, x_i^u : h_j^g = \underset{h^g \in \mathbf{H}^{\mathrm{g}}}{\operatorname{argmax}} \mathbf{ATT}(\theta_{x_i^u}, h^g)\}. \tag{8}$$

[2]We use inner product in our implementation, as there's no significant empirical improvement when using other more complicated forms.



(A) Integration-By-Competition (B) Integration-By-Allocation

Figure 3: Candidate Integration Strategies.

For example, channel 2, 6 and 7 are the activated channels for the sample user in Fig. 2, as the largest attention weights of the user's behaviors take place in these channels.

• **Grouped Attentive Aggregation.** The user representations are generated with the following two steps: 1) grouping w.r.t. the activated channels $\mathbf{C}^u$, and 2) attentive aggregation.

Firstly, user behaviors are partitioned into groups, such that behaviors within the same group share the common activated channel. In other words, both x_i and x_j are within the l-th group $\mathbf{X}_l^u$ if

$$\underset{h^g \in \mathrm{H^g}}{\operatorname{argmax}} \mathbf{ATT}(\theta_{x_i^u}, h^g) = \underset{h^g \in \mathrm{H^g}}{\operatorname{argmax}} \mathbf{ATT}(\theta_{x_j^u}, h^g) = h_l^g. \tag{9}$$

For example, the first 3 user behaviors in Fig. 2 (the blue slots) are within $\mathbf{X}_2^u$ as all of them activate the 2nd channel.

Secondly, a user representation will be generated for each group by attentively aggregating its included user behaviors:

$$\phi_l^u = \sum\nolimits_{x_i \in \mathrm{X}^u} \alpha_{i,l} * \theta_{x_i^u},$$
$$\alpha_{i,l} = \frac{\exp(\mathbf{ATT}(\theta_{x_i^u}, h_l^a))}{\sum_{\mathrm{X}^u} \exp(\mathbf{ATT}(\theta_{x_i^u}, h_l^a))}. \tag{10}$$

In this place, h_l^a is the aggregation head of the l-th channel.

• **Remarks.** Compared with the "read-write" operations in conventional multi-channel structures, the above user representation has two notable properties. The first one is the "Elasticity": although a large number of channels can be deployed, only those closely related with user's actual interest will activated for computation. Thus, irrelevant representation and unnecessary running cost will be eliminated. The secondly one is the "Purification": one type of user interest is represented purely with its relevant behaviors, i.e., the behaviors within the common group, rather than mixing up different kinds of user behaviors. Therefore, each underlying interest will get better highlighted.

4.1.3 Training Process. Our training objective is quite straightforward: the user representation needs to be close to the ground-truth as much as possible, so that the ground-truth can be confined in its neighbourhood (as stated in Eq. 2). However, one underlying problem is that there can be multiple user representations, and different user representations are used for distinct user interests. Thus, it makes no sense to require all of them get close to the truth. Instead, we will only apply our training objective to the relevant user representation (with the ground-truth). (*For example, if we have a couple of user representations on sports and music, and the ground-truth is "Taylor Swift Albums", then we will only have the representation on music move close to the ground-truth.*) Particularly, we will first identify the "most relevant user representation" w.r.t. the predefined distance function:

$$\phi_l^u = \operatorname{argmin}_{\{\phi_*^u\}^u} \operatorname{dist}(\theta_{x_T^u}, \phi_l^u). \tag{11}$$

Then, we will pick ϕ_l^u as our training targeted and mask the rest. Finally, the distance loss is minimized between ϕ_l^u and the ground-truth item: $\operatorname{dist}(\theta_{x_T^u}, \phi_l^u)$.

Our training process is formalized as Alg. 1. We will first initialize the channels as the orthogonal basis of the whole items' representations (line 3). Then for each presented user, we will 1) attend the user history to the channels, 2) find out the activated channels, 3) partition user behaviors into groups, and 4) aggregate each of the groups to get the user representations (line 6-9). Afterwards, we will identify the targeted user representation and compute the distance loss w.r.t. the ground-truth (line 10-11). The overall training loss is formulated as the summation of average loss in distance, and the orthogonal regularization (λ is the weight coefficient); and finally, the whole network is update w.r.t. the generated gradients until its convergence (line 12-14).

Remarks. In conventional learning of multi-channel structures, representations from all channels, regardless of their relevance to the ground-truth, are aggregated and trained to minimize the distance to the ground-truth item. However, this is problematic as supervision signal will be back-propagated to the irrelevant channels. By identifying the right channels for updating, such a problem is alleviated and the training process will become more robust.

4.2 Integration of Candidates

The nearest neighbours of all user representations $\mathbf{\Phi}^u$ will constitute the final candidate set. As the size of candidate set is predefined, it is necessary to figure out how many neighbourhood items to take from each user representation. A direct strategy is to partition the candidate set equally so that the same number of candidates are generated from each user representation. However, it may not optimize the overall quality of the whole candidates since different user representation could have distinct importance. *For example, when making recommendations for online advertisement, user's web browsing behaviors on commercial websites, like Walmart and Hotel.com, are far more important than those on News websites, like CNN.* In the part, the following two integration strategies are introduced which determines the candidate size individually for different user representations.

4.2.1 Integration By Competition. Because the relevance between an item and a user representation is reflected by their distance: x_i is more relevant to ϕ_l^u than x_j if $\operatorname{dist}(\theta_{x_i}, \phi_l^u) < \operatorname{dist}(\theta_{x_j}, \phi_l^u)$, it is quite natural to extend this principle as our integration strategy (shown as Fig. 3 (A)). Particularly, we will first retrieve the K-nearest neighbours of each user representation ϕ_l^u (denoted as NN_l). Then, for each $x \in \mathrm{NN}_l$, we will associate it with its distance towards ϕ_l^u. Now, we will merge the neighbourhood items of all user representations and sort them with the ascending order of their associate

Algorithm 2: Training of Estimator

```
1 Input: $\{\Phi^u\}_U$ from the well-trained Elastic Archive
    Network.
2 Output: the Relative Importance Estimator.
3 while not converge do
4     for each user u in mini-batch $U_i$ do
5         get the utility of each representation $\gamma_l$ as Eq. 12;
6         get the label of classification $y^u$ as Eq. 13;
7         get the cross-entropy of classification: $\psi(\{\gamma_*\}^u, y^u)$;
8     get the overall training loss: $\sum_{U_i} \psi(\{\gamma_*\}^u, y^u)/|U_i|$;
9     back-propagate the gradients and update.
```

distances. Finally, the items with the top-K shortest distances are selected to be our candidates.

One underlying problem about the above strategy is that distances towards different user representations may not be directly comparable, which breaks the consistency between user-item's relevance and their representation similarity. As a result, the selected candidates may not always be user's most interested ones.

4.2.2 Integration By Allocation. Alternatively, we may assign each user representation with a fixed quota k, where the k-nearest neighbours of all user representations will constitute the candidate set. As pointed out, different user representations may have different relative importance, thus the candidate quota should not be equally allocated. Instead, the allocation is made based on the estimation of user representations' relative importance. Such a problem is formulated as a multi-class classification problem, where a simple 2-layer feed-forward network is used to solve it.

Particularly, we will make use of a two-layer linear perception component (denoted as MLP), which is virtually the cascade of a $d \times d$ and $d \times 1$ linear-mapping layers activated by Sigmoid. With such a component, each of the user representation vectors (i.e., $\phi^u_l \in \Phi^u$) will be mapped to a real-valued utility. We will then normalize these utilities so as to derive the relative importance of all user representations:

$$\gamma_l = \frac{\mathrm{MLP}(\phi^u_l)}{\sum_{\phi^u_* \in \Phi^u} \mathrm{MLP}(\phi^u_*)}. \tag{12}$$

As we expect the ground truth item to be confined in the neighbourhood of one of the user representations, the importance of a user representation will be defined by its closeness with the ground truth item. Therefore, we introduce the following one-hot vector (denoted as y^u) as the label of classification:

$$y^u_l = 1 : \ \phi^u_l = \operatorname*{argmin}_{\Phi^u} \mathrm{dist}(\phi^u_*, \theta_{x^u_T}), \tag{13}$$

where dist(·) is the predefined distance function. In other words, the classifier will learn to identify the user representation which is the most relevant to the ground-truth item.

Finally, the whole training process of relative importance estimation is illustrated as Alg. 2. Notice that the training is performed on top of a well-trained Elastic Archive Network, instead of being coupled as an end-to-end pipeline.

The estimated relative importance will be the guidance of budget allocation: the user representation with higher relative importance should be assigned with a bigger share of budget. However, it is not necessary that the optimal allocation should be exactly proportional to the estimated relative importance. For example, if our estimator could always predict the correct label, then it should be fully trusted and the whole budget should be totally assigned to the predicted user representation (i.e., "winner-takes-all"). In our work, the relative importance is reshaped via "α-proportional" in order to get the best set of candidates.

Specifically, given the estimated relative importance $\{\gamma_*\}^u$, the candidate budget is allocated according to the following ratio:

$$\beta_l = \frac{(\gamma_l)^\alpha}{\sum_{\gamma_* \in \{\gamma_*\}^u} (\gamma_*)^\alpha}, \text{ where } \alpha > 0. \tag{14}$$

The variable α is the hyper-parameter, which is optimally tuned for the best recall rate. If α is less than 1, the ratio will become smoother than $\{\gamma_*\}^u$, which means we should have a more balanced allocation; on the contrary, the ratio will become sharper if α is greater than 1, which means the allocation should be more concentrated. Whatever value of α, the original order of $\{\gamma_*\}^u$ will be preserved, as a representation with higher importance will always get more.

5 EXPERIMENTAL STUDY

5.1 Experiment Settings

5.1.1 Data Description. Our work was originally developed for online advertisement recommendation, whose data is organized with the following schema. Each transaction is composed of a user's historical web-browsing behaviors before timestamp t, and her ad-click at timestamp t (Fig. 4). It can be observed that the recommendation item (advertisement) and user history (web-browsing) are virtually from two different domains, which is quite different from conventional recommendation scenarios. In fact, user's ad-click behavior is highly sparse in reality: the majority of users may seldom click on advertisement, which makes it extremely hard to make recommendation based on historical ad-clicks. However, user's web-browsing behavior is far more frequent and it largely reveals user's underlying interest.

Historical web-browsing before timestamp t
Ad-click @timestamp t

Figure 4: Data Schema.

We use **Bing Ads** production data as our major dataset, where user's browsing behavior is characterized by the title of web page. Similarly, each ad-click is represented by its description title. In our dataset, there are a total of 207,390 transactions, 2,078,047 web titles, 501,463 advertisement titles. On average, each transaction is associated with 18.88 web-browsing behaviors. The titles are encoded by DSSM[3] fine-tuned in production. We adopt 165,143, 12,494 and 29,753 transactions for training, validation and testing, all of which are ordered sequentially (training before validation before testing) so as to avoid data leakage.

[3]https://www.microsoft.com/en-us/research/project/dssm/

	Channel-5			Channel-10			Channel-20		
	Recall	Gain	#Usage	Recall	Gain	#Usage	Recall	Gain	#Usage
Mean	0.3060	42.39%	1.0	0.3060	42.39%	1.0	0.3060	42.39%	1.0
Uni	0.3353	29.94%	1.0	0.3353	29.94%	1.0	0.3353	29.94%	1.0
Multi	0.3909	11.46%	5.0	0.4180	11.05%	10.0	0.4356	11.82%	20.0
Hi-Fi	0.3894	11.89%	5.0	0.4153	11.77%	10.0	0.4243	14.80%	20.0
OCT	**0.4357**	--	3.742	**0.4642**	--	5.145	**0.4871**	--	6.135

Table 1: Comparison of all baselines in terms of recall rate, relative gain in recall rate and number of channel usage on Bing Ads, with 5, 10 and 20 channels deployed for all the Multiplet methods. The top recall rates are marked in bold.

In addition, We use **Amazon Review** (ratings only) datasets[4] for more comprehensive evaluation. Particularly, reviews on the following 5 categories are utilized: CDs and Vinyl, Games, Electronics, Beauty, Grocery and Gourmet Food. The data is organized in the same way as Bing Ads dataset, except that user history and the recommended items are coming from the same domain. The first n-3 (n is the length of user history) ratings of a user are used for training, the (n-2)-th and the last 2 ratings are reserved for validation and testing, respectively.

5.1.2 Baseline Methods. The following baseline methods are adopted in our experiments:

Mean-Pooling, where user's historical behaviors are encoded by deep networks and aggregated via mean-pooling. Such a method is adopted by YouTube-DNN [5], which is one of the most referenced baselines in relevant studies [15, 24, 32].

Uni-Channel, a direct adaption of Mean-Pooling, where the historical behaviors are aggregated via attentive pooling. Such an adaption achieves notable improvement over the original method, as demonstrated in [15].

Multi-Channel, where multiple channels are deployed. Each of the deployed channels will aggregate user history into a vector via attentive pooling [15].

Hi-Fi Ark, a variation of Multi-Channel, where orthogonality regularization is introduced for the deployed channels [20].

Both Mean-Pooing and Uni-Channel represent user with one single vector, while the last two baselines and Octopus generate multiple representations for a user. Based on such a difference, the methods are partitioned into two categories: **Singlet** (Mean-Pooling and Uni-Channel) and **Multiplet** (Multi-Channel, Hi-Fi Ark and Octopus).

Notice that two other common forms of user representation methods are omitted from discussion: 1) the MF based methods, which have been repetitively demonstrated to be inferior than the adopted baselines (e.g., [5, 15]), and 2) the query-sensitive user representation (e.g., DEIN [29]), which is not applicable to candidate generation due to the pre-request of specifying recommendation item in advance.

In addition to the baselines from previous works, we adopt the following variations of Octopus to further study the effect of every functional module. 1) **Basic Elastic**, which only includes the fundamental Elastic Archive Network, 2) **+Orth**, with the orthogonal deployment of channels, 3) **+Group**, with the group attentive pooling added, 4) **OCT (C)** and 5) **OCT (A)**, where all previous modules are included, but the candidate integration is made by competition and allocation, respectively.

[4] http://jmcauley.ucsd.edu/data/amazon/

	Channel-5		Channel-10	
	Min-Max	Avg-Sim	Min-Max	Avg-Sim
Mean	0.0491	0.5332	0.0491	0.5332
Uni	-0.0180	0.4737	-0.0180	0.4737
Multi	**0.3083**	**0.7524**	**0.4838**	**0.8468**
Hi-Fi	0.2991	0.7414	0.4638	0.8393
OCT	0.2994	0.7444	0.4113	0.8278

Table 2: Comparison of comprehensiveness, with 5 and 10 channels deployed for all Multiplet methods.

5.1.3 Metrics. Our ultimate concern is the quality of candidate generation. Therefore, the recall rate is taken as the major indicator of performance. In addition, it is also desirable to know whether the generated user representations comprehensively cover user's diverse interests. Therefore, we introduce two auxiliary metrics: 1) min-max similarity (Min-Max), which stands for the minimum of user's historical behaviors' ($\mathbf{X}^u$) maximum similarities towards the generated user representations ($\mathbf{\Phi}^u$), and 2) average similarity (Avg-Sim), which measures the average similarity between user's historical behaviors and user representations.

$$\begin{aligned} \textbf{Min-Max} &= \min \sum_{x \in \mathbf{X}^u} \max\{\text{sim}(x,\phi) | \forall \phi \in \mathbf{\Phi}^u\}, \\ \textbf{Avg-Sim} &= \frac{\sum_{x \in \mathbf{X}^u} \sum_{\phi \in \mathbf{\Phi}^u} \text{sim}(x,\phi)}{|\mathbf{X}^u||\mathbf{\Phi}^u|}. \end{aligned} \tag{15}$$

Apparently, sufficiently large values for both measurements will indicate the comprehensive coverage of user interest.

	Channel-5	Channel-10	Channel-20	Channel-30
Basic	0.3838	0.4015	0.4153	0.4189
+ Orth	0.3818	0.3989	0.4154	0.4210
+ Group	0.3774	0.4217	0.4487	0.4588
OCT (C)	0.3823	0.4228	0.4604	0.4632
OCT (A)	**0.4357**	**0.4642**	**0.4873**	**0.4848**

Table 3: Analysis of OCT's functional modules on Bing Ads with the number of deployed channels growing from 5 to 30.

5.2 Experiment Analysis

5.2.1 Comparison between all methods. The comparison between all methods are shown in Tab. 1, with 5, 10 and 20 channels deployed for all Multiplet methods (the number of channel deployment is not applied to Mean and Uni, whose performances will always be the same values).

There are two notable observations about the result. Firstly, the highest recall rates can always be achieved by Octopus, which verifies its effectiveness in candidate generation. Secondly, the multiplet methods outperform those singlet methods, which suggests the deployment of multiple channels substantially contribute the quality of candidate.

To further comprehend these phenomenons, additional experiments are carried out to compare the comprehensiveness between different methods. As demonstrated in Tab. 2, the multiplet methods will always give rise to much higher comprehensiveness (both Min-Max and Avg-Sim) in all experiment settings, compared with those singlet methods. The large gap in comprehensiveness indicates that many of the user's historical behaviors are far away from the uniform user representations generated by Mean-Pooling and Uni-Channel. As a result, it is almost impossible to acquire candidates of corresponding categories, which results in the limited recall rates of the Singlet methods.

One more interest point about Tab. 1 is Octopus channel usage. As we can see, only a small portion of channels are activated in Octopus; meanwhile, the growth of channel usage becomes slower when more channels are deployed. Such an observation echoes our previous discussion about how Octopus deals with the "over/under channelization". Particularly, instead of making use of all the deployed channels, Octopus will only activate the channels which are closely related with user's actual interest. Such a property will eliminate the low-quality user representations resulted from irrelevant channels; thus, higher recall rates can be achieved by Octopus despite that fewer channels are utilized. Besides, it will also contribute to reduction of candidate generation cost, as the running efficiency is determined by the rounds of k-NN search, which equals to the number of activated channels.

5.2.2 More analysis on comprehensiveness. As is discussed, the Multiplet methods will always give rise to much higher comprehensiveness, which explains its advantage over the Singlet methods. However, it can also be observed that Multi-Channel and Hi-Fi Ark are inferior to Octopus in terms of recall rate, despite that both methods achieve slightly higher comprehensiveness in contrast to Octopus. Such a phenomenon can be easily comprehended if the channel usage is taken into consideration. Specifically, as much fewer channels are actually used by Octopus, there will be significantly more candidates retrieved for each of its user representations. In other words, the neighbourhood items of each user representation will be covered more sufficiently, which leads to higher chances of hitting the ground truth.

5.2.3 Comparisons between Octopus variations. The evaluation for Octopus variations is shown with Tab. 3. According to the demonstrated results, the highest recall rates can always be obtained when all functional modules are included and the candidates are integrated by allocation, i.e., OCT (A). However, adding orthogonality alone to the basic model does not contribute to the recall rate as expected, despite its effectiveness in improving the ranking accuracy [20]. There are two possible reasons about this observation. Firstly, when small number of channels are deployed, there is a very limited chance of channel redundancy. Such an explanation is consistent with the observation that the performance of +Orth is lower than Basic in Channel-5 and 10, but becomes slightly better than Basic when 20 and 30 channels are deployed. Secondly, the orthogonality requirement will force the channels to be different from each other, which is actually a two-edged sword. On the one hand, it reduces the possibility of channel redundancy, which contributes to the coverage of user's diverse interests; on the other hand, it increases the chance of introducing irrelevant channels, which could lead to the inclusion of noisy candidates.

5.2.4 Comparison between Multiplet. Additional datasets are used for more comprehensive evaluation of the Multiplet methods. As demonstrated in Tab. 4, Octopus outperforms the other two baselines methods in most cases, which validates its effectiveness in dealing with the over/under channelization. Besides, the following interesting properties can be observed from result.

Firstly, although all the Multiplet methods tend to achieve better performance when more channels are deployed, the relative advantage of Octopus is expanded gradually. Such an observation is consistent with our previous discussion. On the one hand, when insufficient numbers of channels are deployed, all the methods suffer from under-channelization; as a result, all performances are limited and the gap between difference methods is small. On the other hand, when a large number of channels are deployed, conventional Multiplet methods are likely to be over-channelized, which slows down their performance growth. However, thanks to Octopus elasticity, the deployed channels are activated only when highly relevant user behaviors are provided. Therefore, noisy candidates from irrelevant channels can be avoided, which leads to more persistent growth of the performance.

Secondly, there is no consistent relationship between Multi-Channel and Hi-Fi Ark. Since the major difference between both methods is the orthogonality of channels, it echoes our previous analysis about the orthogonality, as both positive and negative effects are generated from it. For the current stage, the inclusion of orthogonality has to be determined empirically, which drives us to think about more effective way of coordinating the channels for better candidate generation.

	Bing Ads			Amazon CD			Amazon Game		
	Channel-5	Channel-10	Channel-20	Channel-3	Channel-5	Channel-10	Channel-3	Channel-5	Channel-10
Multi	0.3909	0.4180	0.4356	0.2256	0.2624	0.2827	0.2152	0.2233	0.2418
Hi-Fi	0.3894	0.4153	0.4243	0.2273	**0.2790**	0.2861	0.2233	0.2302	0.2314
OCT	**0.4357**	**0.4642**	**0.4871**	**0.2310**	0.2741	**0.3201**	**0.2395**	**0.2706**	**0.3047**
	Amazon Electronic			**Amazon Beauty**			**Amazon Grocery**		
	Channel-3	**Channel-5**	**Channel-10**	**Channel-3**	**Channel-5**	**Channel-10**	**Channel-3**	**Channel-5**	**Channel-10**
Multi	0.2241	0.2575	0.2703	0.2959	0.2959	0.2976	0.2766	0.2987	0.3015
Hi-Fi	0.2177	0.2559	0.2658	0.2875	0.2953	0.2903	0.2895	0.3097	0.3217
OCT	**0.2456**	**0.2996**	**0.3340**	**0.3048**	**0.3470**	**0.3576**	**0.3327**	**0.3493**	**0.3502**

Table 4: Comparison of all Multiplet methods on 6 datasets. The top recall rates are marked in bold.

5.2.5 *Case Study.* Fig. 5 demonstrates the case study of two sample users. For the first user, a total of 3 user representations are generated by Octopus (with 10 deployed channels), whose nearest ad-titles are shown in the upper blue box. It can be observed that each of these Ad-titles is corresponding to one specific kind of web-browsing behaviors in user history: ticket service, TV and Robot Vacuum (marked with red pentagram, green 4-point star and golden hexagram, respectively). While for the second user (whose interest is more diverse), 6 out of the 10 channels are activated for user representation. Once again, the neighbourhood ad-titles of all user representations fully cover the exhibited user interests. Both cases clearly demonstrate Octopus' elasticity: the scale and type of user representation are adaptively determined so that user's diverse interests can be captured comprehensively and precisely.

5.3 Summary

The major experimental findings are summarized into the following points.

• The quality of candidate generation is substantially improvement by Octopus, thanks to the deployment of multiple channels and its capability of addressing over/under channelization.

• The comprehensive coverage of user's diverse history is crucial for candidate generation, which gives rise to Multiplet methods' performance advantage over those of Singlet.

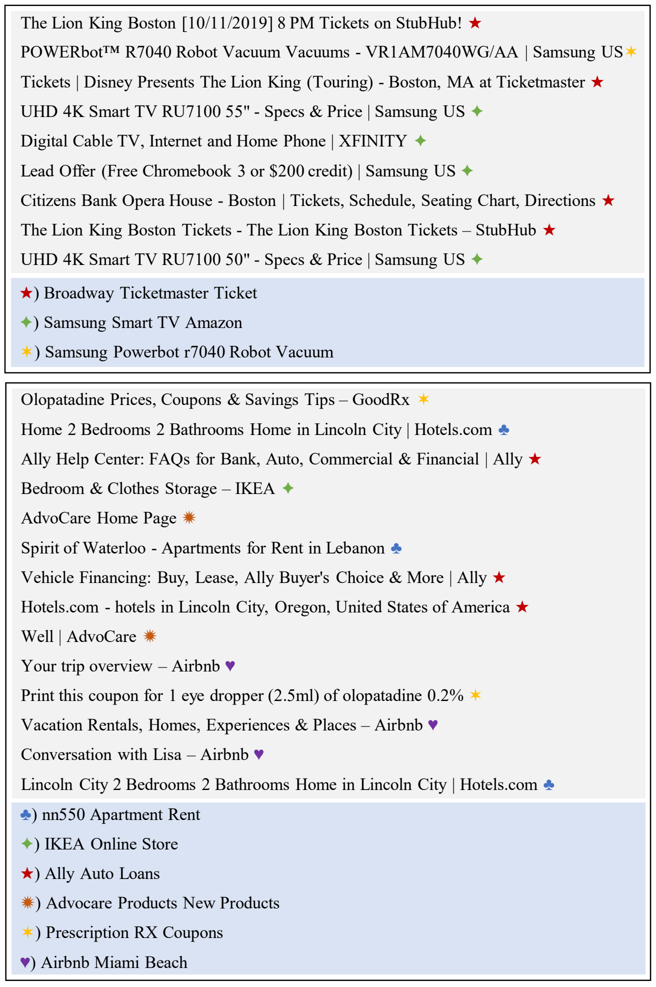
6 CONCLUSION AND FUTURE WORK

In this work, we propose a comprehensive and elastic user representation framework, Octopus, for personalized candidate generation. On top of the Elastic Archive Network, the type and scale of user representation can be adaptively determined based on user's individual background. Besides, with the proposed representation workflow, user's diverse interests can be better extracted and highlighted; and leveraging the integration strategies, different representations' neighbourhood items can be assembled for more effective candidate set. Extensive experiments on Bing Ads and Amazon datasets verify Octopus' effectiveness in candidate generation.

Notice that the current framework is static: no temporal order of user history is considered; besides, it is incapable of being updated continuously: the representations need to be computed all over again everytime a new user behavior comes. For the next step, a sequential framework will be developed, which will further improve the representation quality and reduce the maintaining cost.

REFERENCES

[1] Mingxiao An, Fangzhao Wu, Chuhan Wu, Kun Zhang, Zheng Liu, and Xing Xie. 2019. Neural news recommendation with long-and short-term user representations. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. ACL, 336–345.
[2] Ioannis Antonellis, Hector Garcia Molina, and Chi Chao Chang. 2008. Simrank++: query rewriting through link analysis of the click graph. *VLDB* 1, 1 (2008), 408–421.
[3] Siqi Bao, Huang He, Fan Wang, and Hua Wu. 2019. PLATO: Pre-trained Dialogue Generation Model with Discrete Latent Variable. *arXiv preprint arXiv:1910.07931* (2019).
[4] Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2018. Sequential recommendation with user memory networks. In *Proceedings of the eleventh ACM international conference on web search and data mining*. ACM, 108–116.
[5] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*. ACM, 191–198.
[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
[7] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems*. 13042–13054.
[8] 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*. 278–288.
[9] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in neural information processing systems*. 1024–1034.
[10] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).
[11] 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 Information & Knowledge Management*.
[12] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 197–206.


The Lion King Boston [10/11/2019] 8 PM Tickets on StubHub! ★
POWERbot™ R7040 Robot Vacuum Vacuums - VR1AM7040WG/AA | Samsung US ✶
Tickets | Disney Presents The Lion King (Touring) - Boston, MA at Ticketmaster ★
UHD 4K Smart TV RU7100 55" - Specs & Price | Samsung US ✦
Digital Cable TV, Internet and Home Phone | XFINITY ✦
Lead Offer (Free Chromebook 3 or $200 credit) | Samsung US ✦
Citizens Bank Opera House - Boston | Tickets, Schedule, Seating Chart, Directions ★
The Lion King Boston Tickets - The Lion King Boston Tickets – StubHub ★
UHD 4K Smart TV RU7100 50" - Specs & Price | Samsung US ✦

★) Broadway Ticketmaster Ticket
✦) Samsung Smart TV Amazon
✶) Samsung Powerbot r7040 Robot Vacuum

Olopatadine Prices, Coupons & Savings Tips – GoodRx ✶
Home 2 Bedrooms 2 Bathrooms Home in Lincoln City | Hotels.com ♣
Ally Help Center: FAQs for Bank, Auto, Commercial & Financial | Ally ★
Bedroom & Clothes Storage – IKEA ✦
AdvoCare Home Page ✹
Spirit of Waterloo - Apartments for Rent in Lebanon ♣
Vehicle Financing: Buy, Lease, Ally Buyer's Choice & More | Ally ★
Hotels.com - hotels in Lincoln City, Oregon, United States of America ★
Well | AdvoCare ✹
Your trip overview – Airbnb ♥
Print this coupon for 1 eye dropper (2.5ml) of olopatadine 0.2% ✶
Vacation Rentals, Homes, Experiences & Places – Airbnb ♥
Conversation with Lisa – Airbnb ♥
Lincoln City 2 Bedrooms 2 Bathrooms Home in Lincoln City | Hotels.com ♣

♣) nn550 Apartment Rent
✦) IKEA Online Store
★) Ally Auto Loans
✹) Advocare Products New Products
✶) Prescription RX Coupons
♥) Airbnb Miami Beach

Figure 5: Case Study: user's historical web-browsing behaviors are shown in the grey boxes, and the recommended Ad's titles are shown in the blue boxes.

[13] Noam Koenigstein, Parikshit Ram, and Yuval Shavitt. 2012. Efficient retrieval of recommendations in a matrix factorization framework. In *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, 535–544.

[14] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. 2014. *Mining of massive datasets*. Cambridge university press.

[15] Chao Li, Zhiyuan Liu, Mengmeng Wu, Yuchi Xu, Huan Zhao, Pipei Huang, Guoliang Kang, Qiwei Chen, Wei Li, and Dik Lun Lee. 2019. Multi-Interest Network with Dynamic Routing for Recommendation at Tmall. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. ACM, 2615–2623.

[16] Defu Lian, Qi Liu, and Enhong Chen. 2020. Personalized Ranking with Importance Sampling. In *Proceedings of The Web Conference 2020*. 1093–1103.

[17] Defu Lian, Haoyu Wang, Zheng Liu, Jianxun Lian, Enhong Chen, and Xing Xie. 2020. LightRec: A Memory and Search-Efficient Recommender System. In *Proceedings of The Web Conference 2020*. 695–705.

[18] Danyang Liu, Jianxun Lian, Ying Qiao, Jiun-Hung Chen, Guangzhong Sun, and Xing Xie. 2019. Fast and Accurate Knowledge-Aware Document Representation Enhancement for News Recommendations. *arXiv preprint arXiv:1910.11494* (2019).

[19] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

[20] Zheng Liu, Yu Xing, Fangzhao Wu, Mingxiao An, and Xing Xie. 2019. Hi-Fi ark: deep user representation via high-fidelity archive network. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. 3059–3065.

[21] Yury A Malkov and Dmitry A Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence* (2018).

[22] Andriy Mnih and Geoffrey E Hinton. 2009. A scalable hierarchical distributed language model. In *Advances in neural information processing systems*. 1081–1088.

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

[24] Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. DKN: Deep knowledge-aware network for news recommendation. In *Proceedings of the 2018 world wide web conference*. 1835–1844.

[25] Jingdong Wang, Naiyan Wang, You Jia, Jian Li, Gang Zeng, Hongbin Zha, and Xian-Sheng Hua. 2013. Trinary-projection trees for approximate nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence* 36, 2 (2013), 388–403.

[26] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 974–983.

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

[28] Wei Vivian Zhang, Xiaofei He, Benjamin Rey, and Rosie Jones. 2007. Query rewriting using active learning for sponsored search. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 853–854.

[29] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep interest evolution network for click-through rate prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 5941–5948.

[30] Xiao Zhou, Danyang Liu, Jianxun Lian, and Xing Xie. 2019. Collaborative metric learning with memory network for multi-relational recommender systems. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. AAAI Press, 4454–4460.

[31] Han Zhu, Daqing Chang, Ziru Xu, Pengye Zhang, Xiang Li, Jie He, Han Li, Jian Xu, and Kun Gai. 2019. Joint Optimization of Tree-based Index and Deep Model for Recommender Systems. *arXiv preprint arXiv:1902.07565* (2019).

[32] Han Zhu, Xiang Li, Pengye Zhang, Guozheng Li, Jie He, Han Li, and Kun Gai. 2018. Learning Tree-based Deep Model for Recommender Systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1079–1088.