FINN: Feedback Interactive Neural Network for Intent Recommendation

Yatao Yang
Alibaba Group
Hangzhou, China
yatao.yyt@alibaba-inc.com

Biyu Ma
Alibaba Group
Hangzhou, China
biyu.mby@alibaba-inc.com

Jun Tan
Alibaba Group
Hangzhou, China
tanjun.tj@alibaba-inc.com

Hongbo Deng*
Alibaba Group
Hangzhou, China
dhb167148@alibaba-inc.com

Haikuan Huang
Alibaba Group
Hangzhou, China
haikuan.hhk@alibaba-inc.com

Zibin Zheng
Sun Yat-sen University
Guangzhou, China
zhzibin@mail.sysu.edu.com

ABSTRACT

Intent recommendation, as a new type of recommendation service, is to recommend a predicted query to a user in the search box when the user lands on the homepage of an application without any input. Such an intent recommendation service has been widely used in e-commerce applications, such as Taobao and Amazon. The most difficult part is to accurately predict user's search intent, so as to improve user's search experience and reduce tedious typing especially on mobile phones. Existing methods mainly rely on user's historical search behaviors to estimate user's current intent, but they do not make full use of the feedback information between the user and the intent recommendation system. Essentially, feedback information is the key factor for capturing dynamics of user search intents in real time. Therefore, we propose a feedback interactive neural network (FINN) to estimate user's potential search intent more accurately, by making full use of the feedback interaction with the following three parts: 1) Both positive feedback (PF) and negative feedback (NF) information are collected simultaneously. PF includes user's search intent information that the user is interested in, such as the query used and the title clicked. NF indicates user's search intent information that the user is not interested in, such as the query recommended by the system but not clicked by the user. 2) A filter-attention (FAT) structure is proposed to filter out the noisy feedback and get more accurate positive and negative intentions of users. 3) A multi-task learning is designed to match the correlation between the user's search intent and query candidates, which can learn and recommend query candidates from user interests and disinterests associated with each user. Finally, extensive experiments have been conducted by comparing with state-of-the-art methods, and it shows that our FINN method can achieve the best performance using the Taobao mobile application dataset. In addition, online experimental results also show that our method improves the CTR by 8% and attracts more than 7.98% of users than the baseline.

*corresponding author

CCS CONCEPTS

• **Information systems → Query suggestion**; **Query intent**.

KEYWORDS

Intent Recommendation, Query Suggestion, Query Understanding

ACM Reference Format:
Yatao Yang, Biyu Ma, Jun Tan, Hongbo Deng, Haikuan Huang, and Zibin Zheng. 2021. FINN: Feedback Interactive Neural Network for Intent Recommendation. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3442381.3450105

1 INTRODUCTION

The intent recommendation method is an important feature in the e-commerce search system. It can recommend queries that match a user's search intent, so as to reduce the user's typing burden during the search process. As shown in Figure 1, we take Taobao mobile application as an example, and introduce the process of its intent recommendation system. Firstly, a user opens and lands on the homepage of Taobao. Then, the intent recommendation model will recommend a query by learning the user's historical behavior in real time, and the recommended query will be displayed in the search box. If it meets the user's search intent, the user can choose to submit the query directly without typing anything. Otherwise, the user can issue a new query, and then browses or clicks on the search results. When the user returns to the home page, the user's recent behavior will be learned by the model in real time, and, at the same time, the intent recommendation will update and suggest a new query to the user. In this way, the intent recommendation service can help users search more efficiently and quickly.

Since the intent recommendation method can improve the user's search experience, it has attracted more and more research works in recent years [11, 17, 30]. They use the user's historical search or click behavior to understand the user's search intent. However, they ignore the impact of the feedback interaction information between the user and the intent recommendation system. This information is important for capturing users' search intent in real time. Next, we will explain why we need to consider feedback interaction information and how our model can use this information.

Firstly, with the feedback interaction information between the user and the recommendation system, we can not only understand the search intents that the user is interested in, but also the search

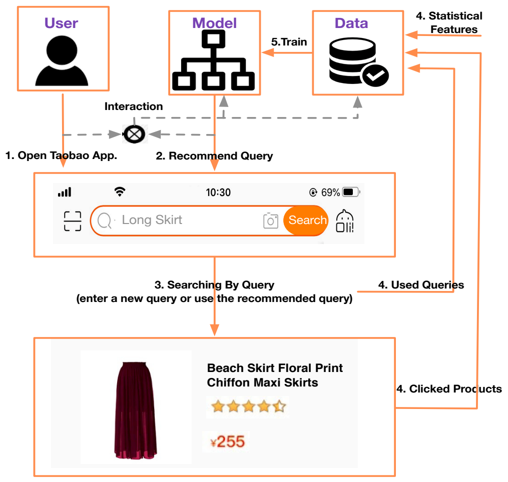


Figure 1: The process of Intent recommendation in Taobao App.

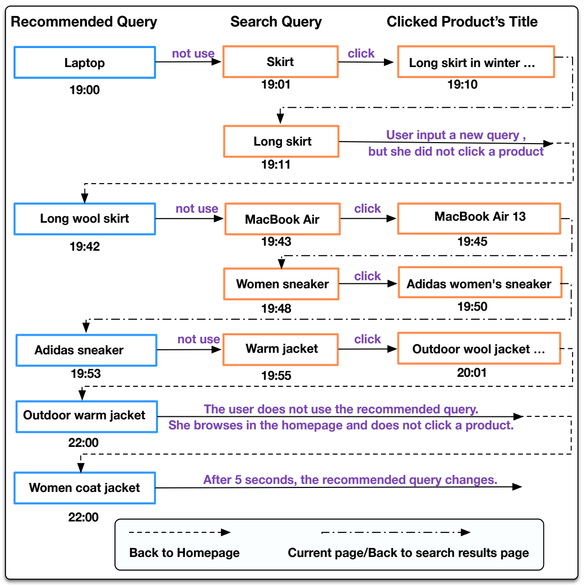

Recommended Query
Search Query
Clicked Product's Title
Laptop 19:00
not use
Skirt 19:01
click
Long skirt in winter ... 19:10
Long skirt 19:11
User input a new query , but she did not click a product
Long wool skirt 19:42
not use
MacBook Air 19:43
click
MacBook Air 13 19:45
Women sneaker 19:48
click
Adidas women's sneaker 19:50
Adidas sneaker 19:53
not use
Warm jacket 19:55
click
Outdoor wool jacket ... 20:01
Outdoor warm jacket 22:00
The user does not use the recommended query. She browses in the homepage and does not click a product.
Women coat jacket 22:00
After 5 seconds, the recommended query changes.
Back to Homepage
Current page/Back to search results page

Figure 2: A search process example on Taobao App.

intents that the user is not interested in. Figure 2 shows a search process example on Taobao mobile application. It should be noted that the intent recommendation system will suggest a query when the user enters or returns to the homepage. According to the queries used and titles clicked by the user, we can learn the information that the user is interested in "Long skirt in winter", "MacBook Air", "Women sneakers" and "wool jacket". We name these information as positive feedback (PF) in the following paper. On the other hand, there are many queries that have been suggested by the system and the user does not click them, such as "laptop", "Long wool skirt", "Adidas sneaker", "Outdoor warm jacket" and "Women coat jacket". We name those information as negative. Therefore, to fully capture the user's search intents, both the PF and NF information should be considered in the first component of our FINN model.

Secondly, identifying the noise in the feedback information can improve the model and recommend more appropriate queries to users. If we only consider used queries and clicked titles like previous methods, the intent recommendation model is more likely to recommend some queries related to user's recent search intent, such as "outdoor warm jacket" and "woman coat jacket" in Figure 2. However, from the interaction between PF and NF, we can know that "warm jacket", "women sneaker" and "long skirt" might be noisy signal in PF. Because these queries have been recommended to the user before, but the user did not use them. On the other hand, "Laptop" might be a noisy signal in NF. Because the related query ("MacBook Air") has been used by the user later. Therefore, we need to utilize the entire feedback interaction information to get the user's search intents accurately. In the second component of our FINN model, we propose a **F**ilter **AT**tention (FAT) structure to address this problem. It can utilize the interaction between PF and NF to reduce the impact of the noises.

Thirdly, we can utilize multiple views to match candidate queries with user search intent. After obtaining the user's search intents accurately, in the third component of our FINN model, we propose a multi-task learning framework that learns the following three tasks: a binary classification task (click/not click), a similarity task between the query and the PF information, an irrelevant task between the query and the NF information. As a result, our model can recommend the query which considers three aspects as follows: a high possibility to be clicked, close to the search intents that the user is interested in, and away from the search intents that the user is not interested in.

Overall, the contribution of our model can be summarized as follows:

- We propose a FINN method which can make full use of the feedback interaction information between users and the intent recommendation system.
- Our FINN method has the following advantages: 1) Considering both the user's interested and disinterested search intents. 2) Using feedback interaction between the user and the recommendation system to obtain the user's search intents more accurately. 3) Matching the candidate query with the user's search intents from multiple angles.
- Compared to the baseline methods, the experimental results on Taobao mobile application show that our FINN model can achieve the best results.

2 RELATED WORKS

In the early research phase, some works leveraged the "wisdom of crowds" by mining the query log data to find alternative queries for suggesting. Huang et al. [13] ranked the co-occurrence query pairs by the frequency and then got the top-ranked queries as the suggestions. Boldi et al. [4] built a query-flow graph by the query session dataset and estimated the transfer probability from one query to another. Then, a lot of related works tried to find similar queries as the query suggestion. Baeza-Yates et al. [2] utilized a k-means model to find the similar queries based on the clicked data and Bar-Yossef et al. [3] represented candidate queries and

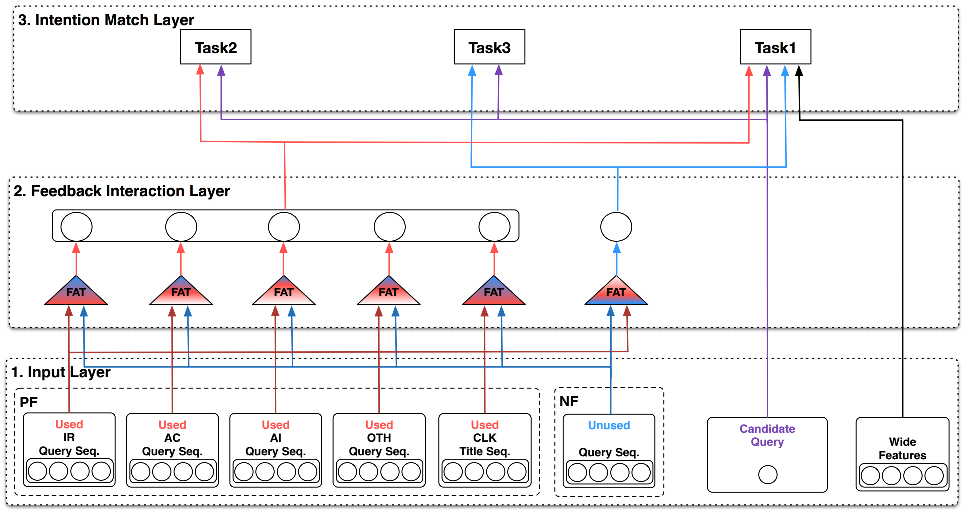


Figure 3: The structure of FINN.

contexts as high-dimensional term-weighted vectors and computed the cosine similarity between them. Then, the top-ranked candidate queries can be recognized as the suggestion. Chaudhuri et al. [7] used the edit distance to judge the similarity and obtained more accurate query suggestions. In addition, many works [5, 10, 14] considered that the query reformulation can be typecast into syntactic and semantic reformulation. Jiang et al. [15, 16] utilized the predefined reformulation strategies to model syntactic reformulation and leveraged the well-established ontologies to learn semantic reformulation. Then, some studies [25, 26] tried to extract candidate queries firstly and then utilized the learning-to-rank methods to rank candidate queries according to some syntactic and semantic reformulation hand-crafted features. Shokouhi et al. [27] utilized the user's long term search history features to build a personalized ranking model based on LambdaMART [6], which is one of the state-of-the-art ranking models. Jiang et al. [16] proposed 43 user reformulation-based features to capture user reformulation behaviors over search sessions with LamdaMART.

Due to the success of deep learning on computer vision, a lot of deep learning methods also have been proposed in the field of recommendation [8, 22, 31]. In recent years, many related works utilized the deep learning methods to generate query reformulation as the suggestion. Sordoni et al. [29] considered the encoder as two layers. The first layer learned the sequence of terms in each query. The second layer learned the sequence of each previously issued queries. Then, the decoder generated a sequence of terms as the query suggestion. In addition, the deep learning model also provided the log-likelihood score of the query generation models as one additional feature to enhance the learning-to-rank model. Jiang et al. [17] proposed a reformulation inference network. It not only learns the query generation task but also learns the difference between the previous query and the current query. Wu et al. [30] utilized the information of the unclicked products as the feedback to enhance the representation of the user's search intents and got a good performance in evaluation. Fan et al. [11] designed a heterogeneous information network to learn the representation of the complex objects and rich interactions in the intent recommendation. Then, the information could be used to model the intent recommendation task.

In recent years, since the feedback information can reflect the change of user's search preference in time, more and more query recommendation methods based on feedback information are proposed [21, 32, 33]. They all use users' negative feedback information to weaken the recommended intents that user did not used. But in the e-commerce system, the intent of user are very wide. It often changes by time and context information. However, the related methods ignored the interaction information between the user and intention recommendation system, which is very helpful to capture the users' real time search intents comprehensively and accurately. Then, in the following parts of our paper, we will introduce our method which can make full use of the interaction information.

3 FRAMEWORK

As shown in Figure 3, the structure of FINN can be divided into three parts as follows: the input layer, feedback interaction layer, and intention matching layer. Then, we will introduce the underlying intuition and content of each part briefly.

In order to capture the information of the user's search history comprehensively in the input layer, we collected four kinds of information with different meanings, such as the PF information, the NF information, the wide information, and the candidate query information. Among them, the PF information includes used queries from different channels and clicked titles. It means the search intents that the user is interested in. The NF information includes queries generated by the recommendation system but not clicked by the user. It refers to the search intents that the user is not interested in. The wide information contains many handcrafted features that are related to the candidate query and the user. The candidate query represents the candidate search intents that might be selected to recommend.

Then, to capture the search intents more accurately, we propose a FAT structure in the feedback interaction layer. It can utilize the feedback interaction between the query recommendation system and the user to help the model learn the changes of the user's search intents in real time and reduce the impact of the noise in PF and NF.

Finally, in order to match the user's search intents and the candidate query's intents more accurately and comprehensively, we design three loss functions. The detail information will be introduced in the next section.

4 THE FINN METHOD

4.1 Input Layer

In the input layer, our model learns four parts of information, such as PF information, NF information, candidate query information, and wide information.

4.1.1 Positive Feedback. We collect PF information through different channels. In this paper, the PF information is composed of different types of user's search behavior sequences, such as used intent recommendation (IR) query sequence, used auto-completion (AC) query sequence, actively input (AI) query sequence, the query sequence from other (OTH) searching channels, and clicked (CLK) product's title sequence. It should be noted that AC refers to using query auto-completion after the user enters some words. For example, after a user types a word with "outdoor", he utilizes the "outdoor jacket" query that is generated by the query auto-completion system. In addition, the number of records that are searched through AC, IR, and AI channels is more than 90% of the total search records in one day in the Taobao app. Therefore, we collect the query sequence from four channels as follows: AC, IR, AI, and OTH.

Next, we will introduce the modeling process of these sequences. Firstly, as Equation 1 shows, we split the query and item's title information by word level.

$$\mathbf{x}_i^* = [x_{i,1}^*, \ldots, x_{i,j}^*, \ldots, x_{i,k}^*], \tag{1}$$

wherein, $\mathbf{x}_i^*$ means the word sequence information of the i-th query or title in one type of user searching behavior sequence among the PF information. $* \in \{IR, AC, AI, OTH, CLK\}$, and $x_{i,j}^*$ is the j-th word in the $\mathbf{x}_i^*$. In addition, k means the number of word in $\mathbf{x}_i^*$. Then, we use the embedding method, which is common in deep learning methods in many research fields [9, 11, 20], to map each word to a low-dimensional space. In this paper, we define a word embedding weight matrix $\mathbf{M} \in \mathbb{R}^{n \times d1}$ to represent all words. It should be noted that the value of the $\mathbf{M}$ will be learned and updated in the model training process. n is the number of all words and $d1$ means the size of the embedding vector.

$$\mathbf{v}_i^* = g(\mathbf{x}_i^*) = g([\mathbf{M}_{x_{i,1}^*}, \ldots, \mathbf{M}_{x_{i,j}^*}, \ldots \mathbf{M}_{x_{i,k}^*}]), \tag{2}$$

wherein, $\mathbf{v}_i^*$ refers to the embedding vector of the i-th query or title in one type of the user's searching behavior sequences. $\mathbf{M}_{x_{i,j}^*}$ means the embedding vector of the word $x_{i,j}^*$. In addition, $g(\cdot)$ is the aggregation function. In this step, we use the average function as the aggregation function. The vector of the query or title is obtained by the average of its all words' vectors.

4.1.2 Negative Feedback. We utilize the information of the queries (query sequence) which are recommended by the recommendation system but not used by the user to represent the NF information. Then, in the same way from Equation 1 to Equation 2, we get $\mathbf{v}_i^{NF}$ that is the average of its all words' embedding vectors to represent the embedding vector of i-th query in NF information.

4.1.3 Candidate Query. The candidate query is obtained by the user's historical behavior through some rules, such as I2Q, U2Q, Q2Q, and so on. I2Q and Q2Q respectively represent a batch of high-frequency queries related to the products(items) recently clicked by the user and the queries that have been searched, U2Q represents a batch of queries related to the user's long-term behavior. Candidate query is a batch of queries recalled based on the user's short-term and long-term behavior, and PF/NF is the feedback information collected in the intention recommendation system. Candidate queries that have been recommended and clicked by the user will appear in PF. On the contrary, candidate queries that have been recommended but not clicked by the user will appear in NF. It should be noted that the recommended query is selected from about 400 candidate queries for each user in the forecast phase. In addition, in the training phase, if the user clicks on the recommended candidate query, the label of this sample is 1. If the user enters the search box but does not use the recommended candidate query, the label of this sample is 0.

In the same way from Equation 1 to Equation 2, we get $\mathbf{v}^{CQ}$ that is the average of its all words' embedding vectors to represent the embedding vector of the candidate query.

4.1.4 Wide Features. In this paper, we extract 49 wide (hand-crafted) features. It mainly consists of two components. One is related to the user, such as age, gender, purchasing power, and so on. The other is related to the candidate query, such as length, obtained way, CTR, and so on. For the fairness of the experiment, the hand-crafted features in all baselines are the same as our FINN model.

In order to learn these features more effectively by the neural network, we firstly utilize these features to train a simple GBDT [18] model. Then, we extract the information of each split point in all trees from the GBDT model (including the selected features and separated values) to discretize each statistical feature. Then, we define an embedding weight matrix $\mathbf{M} \in \mathbb{R}^{m \times d2}$ to represent the vector of all category ids. m is the number of all category ids (Sum of each feature's category) and $d2$ means the size of the embedding vector.

$$\mathbf{v}_i^{wide} = \hat{\mathbf{M}}_{c_i}, \tag{3}$$

$$\mathbf{h}^{wide} = Tanh(\mathcal{F}_{wide}([\mathbf{v}_1^{wide}, \ldots, v_{49}^{wide}]), \tag{4}$$

wherein, c_i means the category id of the i-th hand-crafted feature and $\hat{\mathbf{M}}_{c_i}$ means the embedding vector of the category c_i. In addition, $\mathcal{F}_{wide}(\cdot)$ is the fully-connected hidden layer and $Tanh$ [1] is a popular activation function in a lot of deep learning models. $\mathbf{h}^{wide}$ is the hidden state of wide information.

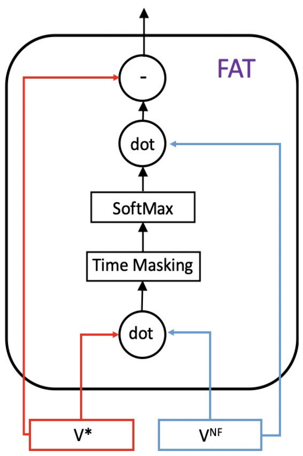


Figure 4: The structure of filter attention (FAT).

4.2 Feedback Interaction Layer

In this layer, we propose a filter attention structure (FAT) that can capture the user's search intents more accurately. The structure of the FAT can be found in Figure 4. We will introduce it as follows:

$$\mathbf{V}^* = [\mathbf{v}_1^*, \ldots, \mathbf{v}_i^*, \ldots, \mathbf{v}_{l_*}^*], \tag{5}$$

$$\mathbf{V}^{NF} = [\mathbf{v}_1^{NF}, \ldots, \mathbf{v}_j^{NF}, \ldots, \mathbf{v}_{l_{NF}}^{NF}], \tag{6}$$

wherein, $\mathbf{V}^* \in \mathbb{R}^{l_* \times d}$ is the vector-matrix of the one type ($* \in \{IR, AC, AI, OTH, CLK\}$) of the user's behavior sequences in positive feedback information, and $\mathbf{V}^{NF} \in \mathbb{R}^{l_{NF} \times d}$ is the vector-matrix of the query sequence in negative feedback information. In addition, l_* and l_{NF} represent the length of the sequence in PF and NF respectively.

$$\mathbf{e} = \mathbf{V}^* \cdot \mathbf{V}^{NF^T}, \tag{7}$$

$$\hat{e}_{i,j} = \frac{I_{i,j} exp(e_{i,j})}{\sum_{a=1}^{l_{NF}} exp(e_{i,a})}, \tag{8}$$

$$I(i,j) = \begin{cases} 0 & t(i) > t(j) \\ 1 & t(i) \le t(j) \end{cases} \tag{9}$$

$$\hat{\mathbf{e}} = \begin{pmatrix} \hat{e}_{1,1} & \hat{e}_{1,2} & \cdots & \hat{e}_{1,l_{NF}} \\ \hat{e}_{2,1} & \hat{e}_{2,2} & \cdots & \hat{e}_{2,l_{NF}} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{e}_{l_*,1} & \hat{e}_{l_*,2} & \cdots & \hat{e}_{l_*,l_{NF}} \end{pmatrix}, \tag{10}$$

wherein, $\mathbf{e} \in \mathbb{R}^{l_* \times l_{NF}}$ is the inner-dot product weight matrix of the PF and NF information. $e_{i,j}$ is the inner-dot product value between the i-th term in the PF sequence and the j-th term in the NF sequence. $I_{i,j}$ is a time masking function. t_i and t_j are the timestamp. When the time of the j-th behavior in the NF sequence is late than the time of the i-th behavior in the PF sequence ($t(i) \le t(j)$), it means that the j-th behavior in the NF sequence can be used to filter the i-th behavior in the PF sequence. For example, the "Outdoor warm jacket" can be used to filter the "warm jacket" information in Figure 2. Because the recommended queries that are related to "jacket" have not been accepted by the user, it means the user may not interested in "jacket" at the current time. In this way, the PF information can be filtered by the future NF information. In addition, $exp(\cdot)$ is the exponential function. and $\hat{e}_{i,j}$ is the result of $e_{i,j}$ after SoftMax and time masking.

$$\hat{\mathbf{V}}^* = \hat{\mathbf{e}} \cdot \mathbf{V}^{NF}, \tag{11}$$

$$\bar{\mathbf{h}}^* = AVG(\mathbf{V}^* - \hat{\mathbf{V}}^*), \tag{12}$$

wherein, $\hat{\mathbf{V}}^* \in \mathbb{R}^{l_* \times d}$ is the a weighted summation ($\hat{\mathbf{e}}$) of the hidden vectors in $\mathbf{V}^{NF}$. It can be considered as the term in the NF sequence which is relevant to the term in PF sequence will be selected. Then, $\mathbf{V}^* - \hat{\mathbf{V}}^*$ means the filtered PF information. We obtain the hidden state ($\bar{\mathbf{h}}^*$) of one type of PF information by the average (AVG) of each term's vector in the sequence.

$$\mathbf{h}^{PF} = Tanh(\mathcal{F}_{PF}([\bar{\mathbf{h}}^{IR}, \bar{\mathbf{h}}^{AC}, \bar{\mathbf{h}}^{AI}, \bar{\mathbf{h}}^{OTH}, \bar{\mathbf{h}}^{CLK}])), \tag{13}$$

In addition, to get the hidden state of all types of PF information, we concatenate these hidden states, such as $\bar{\mathbf{h}}^{IR}$, $\bar{\mathbf{h}}^{AC}$, $\bar{\mathbf{h}}^{AI}$, $\bar{\mathbf{h}}^{OTH}$, and $\bar{\mathbf{h}}^{CLK}$. $\mathcal{F}_{h_{PF}}(\cdot)$ is the fully-connected hidden layer.

$$\mathbf{h}^{NF} = Tanh(\mathcal{F}_{NF}(\bar{\mathbf{h}}^{NF})), \tag{14}$$

Similarly, if we want to use PF information to filter NF information. We need to swap $\mathbf{V}^*$ and $\mathbf{V}^{NF}$ in Figure 4. It should be noted that PF information ($\mathbf{V}^*$) is the concatenation of all types of PF sequences. Then, we get the $\bar{\mathbf{h}}^{NF}$ according to Equation 5 to formula 12. Finally, we get the hidden state of the filtered NF information $\mathbf{h}^{NF}$ by Equation 14. $\mathcal{F}_{NF}$ is a fully-connected hidden layer.

4.3 Intention Match Layer

After we get the representation (hidden) vectors of PF information ($\mathbf{h}^{PF}$), NF information ($\mathbf{h}^{NF}$), candidate query information ($\mathbf{h}^{CQ}$), and wide information ($\mathbf{h}^{wide}$). We need to use them to determine whether the search intents of the candidate query matches the real intention of the user. Therefore, we design the following three kinds of loss to match the intention.

4.3.1 Binary Loss. The first is a binary classification loss which judges whether the candidate query will be clicked by the user based on the information of PF, NF, candidate query and Wide. When the user clicks the candidate intention, the label is 1. Otherwise it is 0.

$$\hat{y} = \sigma(\mathcal{F}_2(Tanh(\mathcal{F}_1([\mathbf{h}^{PF}, \mathbf{h}^{NF}, \mathbf{h}^{wide}, \mathbf{h}^{cq}])))), \tag{15}$$

wherein, we firstly concatenate the hidden states of PF, NF, Candidate query, and Wide. Then, we utilize a fully-connected layer ($\mathcal{F}_1$) and $Tanh$ to learn the hidden state of them. Finally, we use a fully-connected layer ($\mathcal{F}_2$) and Sigmod activation function (σ) to get the prediction ($\hat{y}$). It should be noted that the range of $\hat{y}$ is [0,1]. When the prediction value is larger, the probability of candidate query is clicked by users will be higher.

$$loss_1 = -(ylog(\hat{y}) + (1-y)log(1-\hat{y})), \tag{16}$$

[1]https://en.wikipedia.org/wiki/Hyperbolic_functions

Table 1: The description of datasets. *All Num.* **means size of all samples.** *Pos Num.* **means size of all positive samples.** *Neg Num.* **means size of all negative samples. We randomly sample negative samples so that the proportion of positive samples is about 15%.**

Day	All Num.	Pos Num.	Neg Num.	Pos Ratio
20200129	151,672,346	4,030,280	20,074,708	16.72%
20200130	152,762,004	3,780,268	21,048,048	15.23%
20200131	167,044,613	4,021,406	22,111,950	15.39%
20200201	181,886,621	4,319,115	24,225,271	15.13%
20200202	166,932,853	3,901,247	22,049,259	15.03%
20200203	159,598,998	3,586,869	20,227,481	15.06%
20200204	151,672,346	3,477,201	19,843,419	14.91%

Equation 16 is a classical binary cross entropy loss function. The greater the difference between the predicted value and the real label, the greater the loss will be got.

4.3.2 Similarity Loss. The second is the similarity loss between the candidate query and the PF information. Because PF is the user's interested search intents, we hope that the candidate query is close to this part of information.

$$cos\_sim(\mathbf{h}^{PF}, \mathbf{h}^{CQ}) = \frac{\mathbf{h}^{PF} \cdot \mathbf{h}^{CQ}}{\sqrt{\mathbf{h}^{PF} \cdot \mathbf{h}^{PF}}\sqrt{\mathbf{h}^{CQ} \cdot \mathbf{h}^{CQ}}}, \tag{17}$$

wherein, $cos\_sim(\mathbf{h}^{PF}, \mathbf{h}^{CQ})$ means the cosine similarity score between the $\mathbf{h}^{PF}$ and $\mathbf{h}^{CQ}$.

$$loss_2 = \frac{1 - cos\_sim(\mathbf{h}^{PF}, \mathbf{h}^{CQ})}{2}, \tag{18}$$

As the range of $cos\_sim(*)$ is [-1,1], we design the similarity loss function in Equation 18. When the candidate query and the PF information are closer, the similarity score is close to 1 and the loss will be close to 0. On the contrary, it approaches 1.

4.3.3 Irrelevant Loss. The third is the irrelevant loss between the candidate query and the NF information. Because NF is the user's disinterested search intents, we hope that the candidate query is not close to this part of information.

$$loss_3 = \frac{1 + cos\_sim(\mathbf{h}^{NF}, \mathbf{h}^{CQ})}{2}, \tag{19}$$

wherein, $cos\_sim(\mathbf{h}^{NF}, \mathbf{h}^{CQ})$ means the cosine similarity score between the $\mathbf{h}^{NF}$ and $\mathbf{h}^{CQ}$. Its range is [-1,1]. When the candidate query and the NF information are closer, the similarity score is close to 1 and the loss will be close to 1. On the contrary, it approaches 0.

$$loss = loss_1 + loss_2 + loss_3, \tag{20}$$

Finally, we add three kinds of loss to match the user's search intents comprehensively.

5 EXPERIMENTS

5.1 Dataset

In this paper, we collect a large-scale dataset from Taobao mobile application. It should be noted that a sample is collected when the system recommends a query to the user. If the user clicks the recommended query, the sample label is 1. Otherwise, it is 0. In addition, the feature of each sample contains four parts: the PF information, the NF information, the candidate query information, and the wide (hand-crafted static features) information. The details of each part can be found in Section 4.1.

In addition, this dataset contains more than 1.1 billion samples which are related to more than 200 million users in seven days. The detail information of the data can be found in Table 1. It should be noted that the number of negative samples is the result after sampling. Because the proportion of positive and negative samples is very unbalanced. Finally, the ratio of positive samples is about 15%.

In the offline experiment, we use the first five days in the data as the training set and the next two days as the test set. To validate the impact of days, we also build three training datasets as follows: 1-day (20200202), 3-day (20200131 - 20200202), and 5-day (20200129 - 20200202).

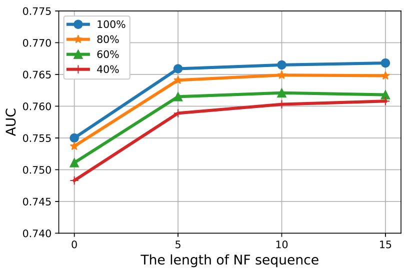
5.2 Baselines and Evaluation Metrics

In order to verify the effect of our FINN model, there are five popular baseline methods in both academic and industry in our paper as follows:

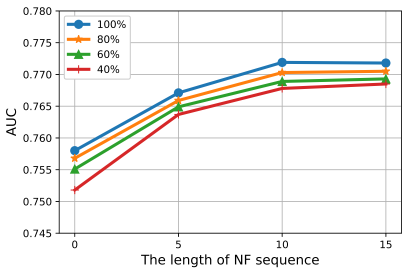
- **Logistic Regression (LR) [24]**: LR is a fast and effective linear model. In addition, the model is interpretable and is widely used in the industry.
- **Gradient Boosting Decision Tree (GBDT) [12, 18]**: GBDT is a tree-based ensemble model, which has the ability to explain the importance of the features and deal with nonlinear features. It also is an important model in the industry.
- **Multilayer perceptron (MLP) [28]**: MLP is a traditional feed-forward artificial neural network. It is also a common base model in both industry and academia.
- **Hierarchical Recurrent Encoder-Decoder (HRED) [29]**: HRED is the first proposed text generative model in the field of query suggestion. It utilizes the hierarchial recurrent neural network to represent the user search behavior sequence. Then, it combines the hidden state of the hierarchical recurrent neural network and some hand-crafted features to learn the query suggestion task. It is an important baseline model in the query suggestion area.
- **Reformulation Inference Network (RIN) [17]**: RIN utilizes the heterogenous network embedding to represent queries and reformulations. Then, it builds a recurrent neural network with the attention mechanism to encode the search session and learn the query suggestion task.
- **Feedback Memory Network (FMN)** [30]: FMN is a query suggestion model which not only learns the query session but also considers the feedback on the search results (displayed products).
- **Metapath-guided Embedding method for Intent Recommendation (MEIRec) [11]**: MEIRec is an intent recommendation model that design a metapath-guided heterogeneous Graph Neural Network (GNN) to learn the embeddings of objects in the intent recommendation. The problem that

Table 2: The performance of Our FINN model and other baselines with different size of the training dataset. * represents the best score of AUC in the baselines. The optimal result under different methods and conditions will be bold.

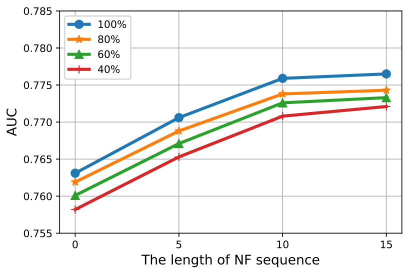
	1-day				3-day				5-day			
Model	**40%**	**60%**	**80%**	**100%**	**40%**	**60%**	**80%**	**100%**	**40%**	**60%**	**80%**	**100%**
LR	0.6904	0.6901	0.6901	0.6903	0.6900	0.6900	0.6904	0.6909	0.6905	0.6909	0.6910	0.6912
GBDT	0.6995	0.7001	0.6999	0.7000	0.6998	0.6997	0.6999	0.7001	0.6998	0.6999	0.6998	0.6998
MLP	0.6983	0.6983	0.6985	0.6988	0.6987	0.6990	0.6991	0.6993	0.6990	0.6994	0.6997	0.7000
HRED	0.7201	0.7205	0.7207	0.7213	0.7217	0.7219	0.7221	0.7227	0.7223	0.7229	0.7229	0.7232
RIN	0.7233	0.7235	0.7239	0.7245	0.7246	0.7250	0.7253	0.7259	0.7255	0.7261	0.7267	0.7280
FMN	0.7257	0.7259	0.7267	0.7273	0.7275	0.7279	0.7286	0.7291	0.7289	0.7296	0.7307	0.7319
MEIRec	0.7297*	0.7313*	0.7340*	0.7351*	0.7362*	0.7367*	0.7377*	0.7385*	0.7376*	0.7392*	0.7403*	0.7421*
FINN	**0.7603**	**0.7621**	**0.7649**	**0.7665**	**0.7678**	**0.7689**	**0.7703**	**0.7719**	**0.7708**	**0.7726**	**0.7738**	**0.7759**
Improvement	4.19%	4.21%	4.21%	4.27%	4.29%	4.37%	4.42%	4.52%	4.50%	4.52%	4.53%	4.55%



(a) 1-day



(b) 3-day



(c) 5-day

Figure 5: Impact of Negative Feedback (NF)

is solved by this method is the same as ours (FINN), and it is also the main baseline that is compared with our method.

In this paper, as same with the evaluation metric in MEIRec, the performance under different baseline is compared according to the Area Under Curve (AUC) [23]. The higher AUC score means a better ability to give the best intent recommendation.

5.3 Detailed Implementation

In this paper, we implement the FINN model through Tensorflow [1] and learn the gradient of the parameters with Adam [19] optimizer. For our method and the related deep learning baseline methods (HRED, RIN, FMN, and MEIRec), the dimension of the word embedding is 64 and the number of hidden neurons of the fully-connected layer is 64. The length of the history searched query session and clicked title session is 10. The maximum number of words in each query and each title is 10 and 11, respectively. In addition, the dimension of the category feature's embedding in our FINN is 16. For the GBDT method, the number of trees is 200 and the max depth is 8. Finally, we set the batch size as 1024 to train the related deep learning baseline and our FINN model.

5.4 Experimental Results

In the offline experiment, in order to compare the performance of our FINN model and baseline methods comprehensively, we conduct the experiments on three different training datasets, such as 1-day, 2-day, and 3-day. In addition, to verify the robustness of different models, we also vary the size of the training dataset from 40% to 100%. Besides, the baseline methods can be divided into two types. The first type can be named as the traditional models, such as LR, GBDT, and MLP. The second type can be called the deep learning model, such as HRED, RIN, FMN, and MEIRec. The experimental results of our FINN model and baselines can be found in Table 2 and summarized as follows:

(1) **Among the baseline models, the experimental result of the deep learning models is better than the traditional models**. In traditional models, LR gets the lowest AUC scores. The reason is that LR can't map the hand-crafted statistic features to high dimensional space and have a limited learning ability. As the GBDT and MLP have this ability, they get a higher AUC score than LR. In deep learning models, because HRED not only can learn the hand-crafted statistic features but also auto-learn the hidden states of the textual information, it gets a higher AUC score than the traditional models. Then, as the RIN model leverages the attention mechanism to encode the textual information much more precisely, it can get a better result than HRED. In addition, FMN not only utilizes the information of clicked products but also consider the information of unclicked products. The AUC score of FMN is higher than the RIN method. Finally, we can found the MEIRec get the best performance among the baselines. The reason may be that it

leverages the graph neural network to enhance the representation of the searched queries and clicked products.

(2) **Our FINN model outperforms all baseline methods on three datasets**. After training on three different training data sets, the AUC scores of our FINN model on the test set increased by more than 4% (4.19% - 4.55%) than the best performance on baselines.

(3) **With the amount of training data increases, the AUC of the model will increase**. As Table 2 shows, we find that the AUC score of our FINN increases with the number of days of training data significantly (0.7665 - 0.7759). In addition, with the increase of the training dataset, the improved range of the AUC score of the traditional models is lower than the deep learning models. The reason should be that the deep learning models are more complex than the traditional models, which need more data to train more parameters.

5.5 Effect of NF

In this paper, to capture the search intents that the user was not interested in, we utilize the information of the query sequence (NF) which was recommended by the recommendation system but not used by the user. In order to verify the influence of NF information on our FINN model, we vary the length of the NF sequence from 0 to 15. It should be noted that 0 means the FINN model without NF information.

As Figure 5 shows, with the increase of the length of the NF information sequence (0 to 15), we can find that the change of the AUC scores under the change of the length of the NF information sequence on different datasets. When the length of the NF sequence information increases from 0 to 5, we can find that the AUC scores increase significantly on three training datasets. The AUC scores of three datasets all increase near 1 percentage points. It can verify that the information that users are not interested in recently has a great impact on the recommendation accuracy of the next query.

Then, when the length of the NF sequence changes from 5 to 10, AUC tends to be stable in the 1-day training dataset. In addition, compared with the length of the NF sequence changes from 5 to 10, the growth rate of the AUC begins to slow down in the other two datasets. It shows that when the amount of data training increases, the ability to learn the long history NF information will increase.

Finally, when the length of the NF sequence information increases from 10 to 15, the AUC scores of three datasets all tends to be stable. It also shows that the influence of the user's disinterested search intention on the future will become weak over time.

5.6 Effect of FAT

In this paper, we propose a FAT structure that can learn the interaction information between PF and NF information to capture the search intents much more accurately. To verify the impact of FAT structure, we design some FINN structures as follows:

- FINN(FAT): This is our proposed model which utilizes the FAT structure to learn the interaction between the PF and NF information.
- FINN(PF,NF): This model means that the FINN model remove the FAT structure. It learns the interaction between the PF and NF information by concatenating them.

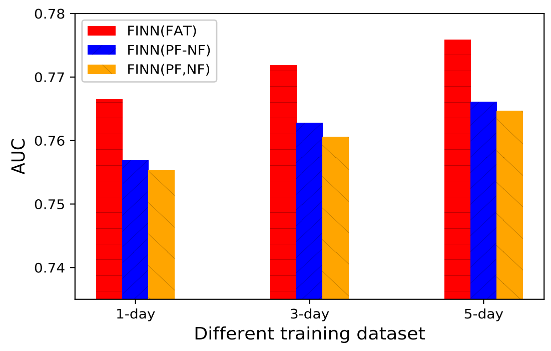


Figure 6: The effect of FAT.

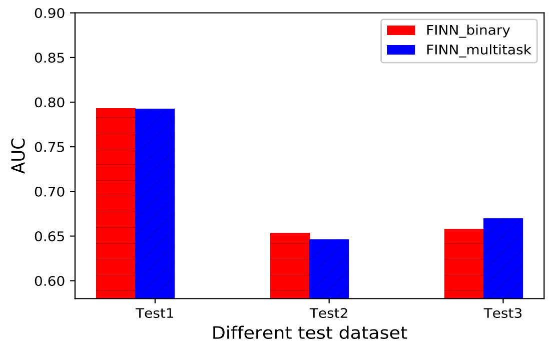


Figure 7: The effect of Multi-task Learning.

- FINN(PF-NF): This model means that the FINN model remove the FAT structure. It learns the interaction between the PF and NF information by the PF information subtract the NF information.

As Figure 6 shows, we can observe the performance of three types of FINN models on three different datasets. Compared with FINN(PF,NF) and FINN(PF-NF), FINN(FAT) get the highest AUC score. It shows that the FAT structure can make full use of the interaction information between PF and NF, so as to obtain the user's search intents more accurately. In addition, we also can find that the performance of FINN(PF-NF) is better than FINN(PF,NF). The reason should be that the NF and PF information contains a lot of noise. For example, the system recommends a query "skirt" and the user does not use it. Then, "skirt" will add in the NF sequence. However, the user may type it to search in the following search process and "skirt" should not be the user's disinterested search intent. The experimental result also shows that it is very important to utilize the FAT structure to weaken the weight of the user's interested search intents in NF and weaken the weight of the user's disinterested search intents in PF in real time.

5.7 Effect of Multi-task Learning

As Table 3 shows, there are some weak negative samples in the training dataset. For example, the system recommended a query

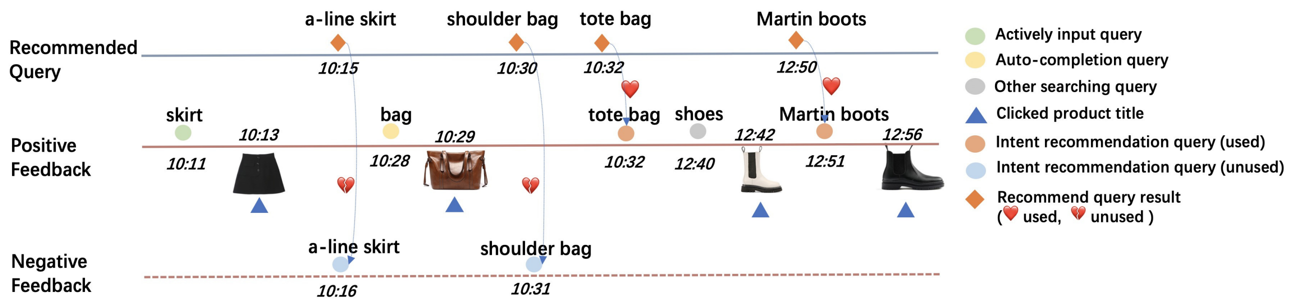


Figure 8: A real search example on Taobao app.

Table 3: Some examples of weak negative samples in training dataset. *Recommendedquery* **means the query recommended by the system.** *Enteredquery* **refers to the query that the user actively enters**

Recommended query	Entered query	Similarity
Trouser	Long Trouser	0.5
Basketball Shoe	Basketball Shoe Nike	0.67
Long Skirt	Skirt Summer	0.33

("Trouser") to a user. However, he did not click it. This sample will be labeled as negative samples. If we optimize the binary classification loss, the model will consider that it is bad to recommend the search intents that are related to "Trouser" under this user's search context. However, this user actually entered a similar query ("Long Trouser"). It indicates that the user is interested in the search intents that are related to "Trouser". To solve this problem, we also propose a similarity task and an irrelevant task. With the multi-task learning mechanism, the system should recommend a query that is related to "Long Trouser" and not similar to "Trouser".

To verify the effect of multitask learning, we first compute the similarity between the recommended query and entered query. It should be noted that the similarity score is computed by the size of the intersection set between words divided by the size of the union between words. Then, we build three test dataset as follows:

- **Test1**: It includes all positive samples (label = 1) and strong negative samples (label = 0) in the raw test dataset. It should be noted that strong negative samples mean that the entered query is not similar to the recommended query. The range of the similarity is [0,0.2].
- **Test2**: It includes all positive samples (label = 1) and weak negative samples (label = 0) in the raw test dataset. It should be noted that weak negative samples mean that the entered query is similar to the recommended query. The range of the similarity is (0.2,1].
- **Test3**: It includes weak positive samples (label = 1) and strong negative samples (label = 0) in the raw test dataset. It should be noted that we set the label of the weak negative sample to 1.

As Figure 7 shows, FINN_binary means the model only has the binary classification loss. FINN_multitask means the model has three losses. On the Test1 dataset, the performance of the FINN_binary and FINN_multitask is similar. This shows that they have a similar ability to distinguish positive samples and strong negative samples. On the Test2 dataset, the AUC score of FINN_binary is higher than FINN_multitask. On the Test3 data set, the AUC score of FINN_multitask is higher than FINN_binary. The result is what we want to see. Because we hope that the model can not have a strong ability to distinguish the weak negative samples from the positive samples, because the weak negative samples actually recommend a query whose intention is very close to the user's search intents. In addition, we also hope that the model has a strong ability to distinguish the weak negative samples from strong negative samples. Because the strong negative samples recommend a query that can not meet the user's search intents.

5.8 Case Study

Figure 8 is a specific example of an intent recommendation system, we can see that user's interests are constantly changing over time, and a wealth of positive and negative feedback behaviors can be obtained during the interaction between the intent recommendation system and the user. Through the FAT structure between positive and negative feedback, the NF information is used to measure which intentions of the user's past intentions will be weakened in the next step, such as "a-line skirt", because the user did not use the recommened query , and there is no similar PF information afterwards. On the other hand, the PF information can also be used to identify the noisy exposure information that may exist in the NF. For example, although the user did not use the recommended query "shoulder bag", but then a similar query "tote bag" was clicked, it is very likely that the "shoulder bag" is a pseudo-exposure information. Therefore, making full use of the positive and negative feedback information in the entire recommendation process can help the system better capture the user's current intents accurately.

6 ONLINE EXPERIMENTS

In order to verify the online effect of our FINN model, we conduct A/B testing experiments on Taobao mobile application. Firstly, we select two buckets to test GBDT (traditional model) and MEIRec (deep learning model) respectively. Then, we also test our FINN

Table 4: Compared with GBDT(Baseline 1), the improvement of FINN and MEIRec(Baseline 2) on online AB test performance.

Model	UV	PV	PCTR	UCTR
MEIRec	2.30%	2.00%	1.44%	2.66%
FINN	10.46%	10.33%	10.53%	10.40%

model on another bucket. The experimental results can be found in Table 4. It should be noted that the value of each metric is the average of the daily online experimental result from February 5, 2020 to February 11, 2020. We can find that our approach has a better performance than the two baselines. Due to the sensitivity of the data, only the relative improvement is shown in the experimental results.

For Unique Visitor (UV), which is the number of people who use the result of the intent recommendation system in a day, our FINN model attracts 10.46% more users than GBDT. For Page View (PV), which refers to the number of clicked pages by users who start to search by the result of the intent recommendation model in a day, our FINN model improves 10.33% than GBDT. For PCTR, which refers to the number of clicking the intent recommendation result is divided by the number of exposure, our FINN model also improves 10.53% than GBDT. For UCTR, which refers to the number of users who use the intent recommendation result is divided by the number of users, our FINN model improves 10.40% than GBDT. At the same time, compared with MEIRec model, our FINN model also has a significant improvement in these various indicators. All in all, the experimental results show that our model also has the best performance on online than baseline methods.

7 CONCLUSION AND FUTURE WORK

In this paper, in order to obtain the user's search intents accurately and then recommend an appropriate query, we proposed a feedback interactive neural network that can make full use of the feedback interaction between the user and the intent recommendation system to capture the change of user's intents in real time. The offline and online experimental results also verify the effectiveness of our FINN model.

In future work, we will explore the effectiveness of more negative feedback information between the user and other suggestion systems in the search engine (e.g., recommended products but not clicked by the user).

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *OSDI).* 265–283.

[2] Ricardo Baeza-Yates, Carlos Hurtado, and Marcelo Mendoza. 2004. Query recommendation using query logs in search engines. In *EDBT*. Springer, 588–596.

[3] Ziv Bar-Yossef and Naama Kraus. 2011. Context-sensitive query auto-completion. In *WWW*. ACM, 107–116.

[4] Paolo Boldi, Francesco Bonchi, Carlos Castillo, Debora Donato, and Sebastiano Vigna. 2009. Query suggestions using query-flow graphs. In *WSCD*. ACM, 56–63.

[5] Paolo Boldi, Francesco Bonchi, Carlos Castillo, and Sebastiano Vigna. 2009. From Dango to Japanese cakes: Query reformulation models and patterns. In *WI-IAT*. IEEE Computer Society, 183–190.

[6] Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning* 11, 23-581 (2010), 81.

[7] Surajit Chaudhuri and Raghav Kaushik. 2009. Extending autocompletion to tolerate errors. In *SIGMOD*. ACM, 707–718.

[8] Liang Chen, Yang Liu, Xiangnan He, Lianli Gao, and Zibin Zheng. 2019. Matching user with item set: collaborative bundle recommendation with deep attention network. In *IJCAI*. 2095–2101.

[9] Zitai Chen, Chuan Chen, Zong Zhang, Zibin Zheng, and Qingsong Zou. 2019. Variational graph embedding and clustering with laplacian eigenmaps. In *AAAI*. 2144–2150.

[10] Van Dang and Bruce W Croft. 2010. Query reformulation using anchor text. In *WSDM*. ACM, 41–50.

[11] Shaohua Fan, Junxiong Zhu, Xiaotian Han, Chuan Shi, Linmei Hu, Biyu Ma, and Yongliang Li. 2019. Metapath-guided Heterogeneous Graph Neural Network for Intent Recommendation. In *SIGKDD*.

[12] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189–1232.

[13] Chien-Kang Huang, Lee-Feng Chien, and Yen-Jen Oyang. 2003. Relevant term suggestion in interactive web search based on contextual information in query session logs. *Journal of the American Society for Information Science and Technology* 54, 7 (2003), 638–649.

[14] Jeff Huang and Efthimis N Efthimiadis. 2009. Analyzing and evaluating query reformulation strategies in web search logs. In *CIKM*. ACM, 77–86.

[15] Jyun-Yu Jiang and Pu-Jen Cheng. 2016. Classifying User Search Intents for Query Auto-Completion. In *ICTIR*. ACM, 49–58.

[16] Jyun-Yu Jiang, Yen-Yu Ke, Pao-Yu Chien, and Pu-Jen Cheng. 2014. Learning user reformulation behavior for query auto-completion. In *SIGIR*. ACM, 445–454.

[17] Jyun-Yu Jiang and Wei Wang. 2018. RIN: Reformulation Inference Network for Context-Aware Query Suggestion. In *CIKM*. ACM, 197–206.

[18] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. In *NIPS*. 3146–3154.

[19] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[20] Siwei Lai, Kang Liu, Shizhu He, and Jun Zhao. 2016. How to generate a good word embedding. *IEEE Intelligent Systems* 31, 6 (2016), 5–14.

[21] Ruirui Li, Liangda Li, Xian Wu, Yunhong Zhou, and Wei Wang. 2019. Click feedback-aware query recommendation using adversarial examples. In *The World Wide Web Conference*. 2978–2984.

[22] Juntao Liu and Caihua Wu. 2017. Deep learning based recommendation: A survey. In *ICISA*. Springer, 451–458.

[23] Jorge M Lobo, Alberto Jiménez-Valverde, and Raimundo Real. 2008. AUC: a misleading measure of the performance of predictive distribution models. *Global ecology and Biogeography* 17, 2 (2008), 145–151.

[24] Andrew Y Ng and Michael I Jordan. 2002. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *NIPS*. 841–848.

[25] Umut Ozertem, Olivier Chapelle, Pinar Donmez, and Emre Velipasaoglu. 2012. Learning to suggest: a machine learning framework for ranking query suggestions. In *SIGIR*. ACM, 25–34.

[26] Rodrygo LT Santos, Craig Macdonald, and Iadh Ounis. 2013. Learning to rank query suggestions for adhoc and diversity search. *Information Retrieval* 16, 4 (2013), 429–451.

[27] Milad Shokouhi. 2013. Learning to personalize query auto-completion. In *SIGIR*. ACM, 103–112.

[28] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature* 529, 7587 (2016), 484.

[29] Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *CIKM*. ACM, 553–562.

[30] Bin Wu, Chenyan Xiong, Maosong Sun, and Zhiyuan Liu. 2018. Query suggestion with feedback memory network. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. WWW, 1563–1571.

[31] Yaoming Wu, Fenfang Xie, Liang Chen, Chuan Chen, and Zibin Zheng. 2017. An embedding based factorization machine approach for web service qos prediction. In *ICSOC*. Springer, 272–286.

[32] Aston Zhang, Amit Goyal, Weize Kong, Hongbo Deng, Anlei Dong, Yi Chang, Carl A Gunter, and Jiawei Han. 2015. adaqac: Adaptive query auto-completion via implicit negative feedback. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*. 143–152.

[33] Bo Zhang, Bin Zhang, Shubo Zhang, and Chao Ma. 2015. Query recommendation based on irrelevant feedback analysis. In *2015 8th International Conference on Biomedical Engineering and Informatics (BMEI)*. IEEE, 644–648.