Learning to Recommend Related Entities with Serendipity for Web Search Users

JIZHOU HUANG, [1]Research Center for Social Computing and Information Retrieval, Harbin Institute of Technology, China [2]Baidu Inc., Beijing, China
SHIQIANG DING, Baidu Inc., Beijing, China
HAIFENG WANG, Baidu Inc., Beijing, China
TING LIU, Research Center for Social Computing and Information Retrieval, Harbin Institute of Technology, China

Entity recommendation, providing entity suggestions to assist users in discovering interesting information, has become an indispensable feature of today's Web search engine. However, the majority of existing entity recommendation methods are not designed to boost the performance in terms of serendipity, which also plays an important role in the appreciation of users for a recommendation system. To keep users engaged, it is important to take into account serendipity when building an entity recommendation system. In this paper, we propose a learning to recommend framework that consists of two components: related entity finding and candidate entity ranking. To boost serendipity performance, three different sets of features that correlate with the three aspects of serendipity are employed in the proposed framework. Extensive experiments are conducted on large-scale, real-world datasets collected from a widely used commercial Web search engine. The experiments show that our method significantly outperforms several strong baseline methods. An analysis on the impact of features reveals that the set of interestingness features is the most powerful feature set, and the set of unexpectedness features can significantly contribute to recommendation effectiveness. In addition, online controlled experiments conducted on a commercial Web search engine demonstrate that our method can significantly improve user engagement against multiple baseline methods. This further confirms the effectiveness of the proposed framework.

CCS Concepts: • **Information systems** → **Web search engines**; **Recommender systems**;

Additional Key Words and Phrases: Serendipity; Serendipitous Recommendations; Serendipitous Entities; Recommender System; Entity Recommendation; Web Search

ACM Reference format:
Jizhou Huang, Shiqiang Ding, Haifeng Wang, and Ting Liu. 2018. Learning to Recommend Related Entities with Serendipity for Web Search Users. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* 1, 1, Article 1 (January 2018), 21 pages.
https://doi.org/10.1145/3185663

1 INTRODUCTION

Entity recommendation, providing entity suggestions to assist users in discovering interesting information, has become an indispensable feature of today's Web search engine. Over the past few years, major commercial Web search engines have enriched and improved user experience of

2375-4699/2018/1-ART1 $15.00
https://doi.org/10.1145/3185663

information retrieval by presenting entity suggestions related to a query besides the regular search results. Figure 1 shows an example of the Baidu[1] Web search engine's search results page for the query "Obama".[2] On the right panel, a ranked list of celebrities related to "Obama" is presented in the group "People also search for", providing users a quick access to entities closely attached to their interests and enhance their information discovery experiences.

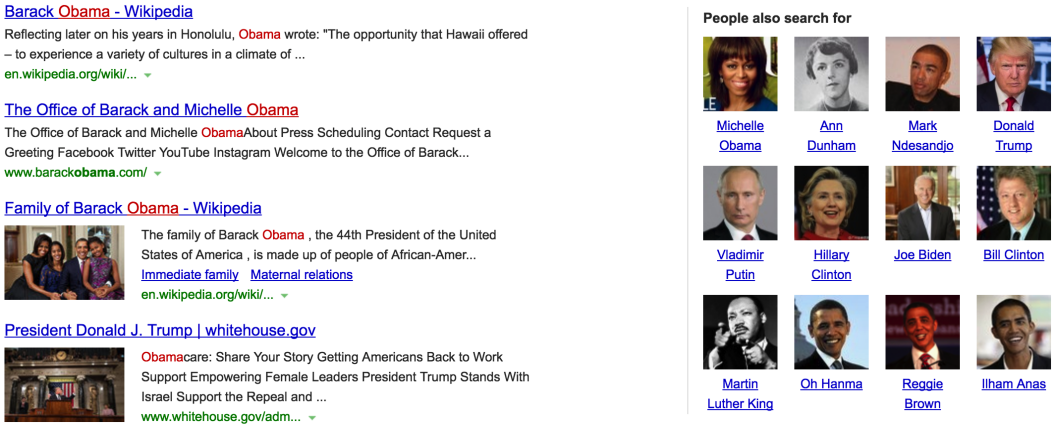


Fig. 1. An example of Baidu's search results page for the query "Obama". The entity recommendations are presented in the group "People also search for" on the right panel.

Serendipity plays an important role in the appreciation of users for a recommendation system and has shown to be a possible way to alleviate the problem of over-specialization [34]. Specifically, over-specialization is the problem that recommendation systems have no inherent method for finding something unexpected [34]. Therefore, recommendation systems generally suffer from the problem of over-specialization [26, 35]. To alleviate this problem, several approaches have been proposed in previous studies, e.g., adding some randomness [36], filtering out items that are too similar [41], or introducing serendipity into a recommendation system [1, 3, 7, 19, 26, 42]. Among these efforts, introducing serendipity has been shown to help users discover some surprisingly unexpected items and has been demonstrated to enhance users' information discovery experiences [1, 7, 26, 37, 42].

Although there is some previous work on building entity recommendation systems, e.g., [2, 4, 5, 13, 40], they are not designed to boost the performance in terms of serendipity, which also plays an important role in the appreciation of users as it might assist users in making surprisingly interesting discoveries. To keep users engaged, it is important to take into account serendipity when building an entity recommendation system. To the best of our knowledge, little research has been published on recommending related entities with serendipity to users.

In this paper, we study the problem of recommending entities with serendipity related to both a given query and a user. We propose a learning to recommend framework that consists of two components: related entity finding and candidate entity ranking. More specifically, given an entity query[3] e_q that a user u is searching for, we first extract a set of candidate entities $\mathcal{R}(e_q)$ related to e_q, and then learn a scoring function to rank the entities in $\mathcal{R}(e_q)$ according to how well they engage

[1]https://www.baidu.com/
[2]We translate the example from Chinese to English for the sake of understanding.
[3]Unless otherwise stated, we assume in this paper that the query is an entity query hereafter.

the interest of u. We define the concept of serendipity in the context of entity recommendation. To boost serendipity performance, three different sets of features that correlate with the three aspects of serendipity are employed in the proposed framework. Extensive experiments are conducted on large-scale, real-world datasets collected from a widely used commercial Web search engine. The experiments show that our method significantly outperforms several strong baseline methods. An analysis on the impact of features reveals that the set of interestingness features is the most powerful feature set, and the set of unexpectedness features can significantly contribute to recommendation effectiveness. In addition, online controlled experiments conducted on a commercial Web search engine demonstrate that our method can significantly improve user engagement against multiple baseline methods. This further confirms the effectiveness of the proposed framework.

The remainder of this paper is organized as follows. The problem statement of this study is introduced in Section 2. Section 3 presents the proposed framework. Experimental setup and result analysis are reported in Section 4 and 5. Important related work is reviewed in Section 6. Finally, we conclude the paper in Section 7.

2 PROBLEM STATEMENT

Before providing a detailed description of the proposed learning to recommend framework in Section 3, we first present the definition of serendipity in the context of entity recommendation. Then, we formally define our task.

2.1 Definition of Serendipity

The definition of serendipity has not yet been standardized, various definitions are proposed for serendipity by researchers in the recommendation system domain, e.g., [1, 7, 19, 23, 26, 42]. However, three important aspects of serendipity are commonly proposed in these studies: *relatedness*, *unexpectedness*, and *interestingness*. Based on these aspects of serendipity, we define *serendipitous entity* in the context of entity recommendation as follows: (1) the entity should be relevant to the query that a user is searching for (*relatedness*); (2) the relation between the entity and the query should not have been otherwise discovered by the user (*unexpectedness*); and (3) more importantly, the entity should engage the interest of the user when searching for the query (*interestingness*). To be brief, a *serendipitous entity* is the relevant entity to a query which could be surprisingly interesting to a user that she might not have otherwise discovered.

To illustrate the idea of serendipity, we consider the following examples of an entity recommendation system. To explain the difference between recommendation results with varying degrees of serendipity, we assume that a user u is familiar with the film "Titanic", her favorite director is "James Cameron", and she is searching for the query "Titanic". First, if the system simply recommends movies that were directed by "James Cameron", for example, "The Terminator", "True Lies", and "Avatar", these recommendations will be highly relevant to the query and the user's preference, but probably not serendipitous due to they might be well-known to u. Second, if the system recommends new movies directed by "James Cameron" that will be released after few years, for example, "Avatar 2", "Avatar 3", and "Avatar 4", which are planned sequels to the 2009 film "Avatar", these recommendations will be novel due to they might be unknown to u, but probably not serendipitous, because they might have been autonomously discovered by u or they would have been easy to discover. The two cases above both provide fewer opportunities for the user to find serendipitous entities. By contrast, if the system recommends movies that were directed by an unfamiliar director to u or by a new director, for example, "The Poseidon Adventure" (a 1972 disaster film directed by "Ronald Neame") and "Titanic II" (a 2010 low-budget disaster film directed by "Shane Van Dyke", and it is not a sequel to the 1997 film "Titanic" directed by "James Cameron"),

these recommendations are more likely to help the user u find serendipitous entities, because they uncover some interesting connections with the query and the user's preference, and they might not have been otherwise discovered by u. From the examples, we can see that recommendations that are serendipitous are by definition also novel [26], but recommendations that are novel are probably not serendipitous. To keep users engaged, it is important for the system to be capable of recommending not only relevant and novel entities, but also serendipitous entities.

2.2 Task Formulation

In this subsection, we formally define the task of recommending entities with serendipity related to both a given query and a user. Introducing serendipity requires to facilitate the recommendation of serendipitous entities, it is therefore important to choose an appropriate target to optimize for the entity recommendation model. In this work, we aim to boost serendipity performance of the system, thus we choose features and optimizing target that correlate with the three aspects of serendipity. To this end, we employ various features that may imply different degrees of *relatedness*, *unexpectedness*, and *interestingness*, and choose the *interestingness* target to rank candidate entities. Therefore, the three aspects of serendipity can be combined in one recommendation model to predict serendipitous entities with respect to a query and a user.

More specifically, given a query e_q that a user u is searching for, our task is to extract a set of candidate entities $\mathcal{R}(e_q) = \{e_1, e_2, \cdots, e_n\}$ related to e_q, and then learn a scoring function to rank the entities in $\mathcal{R}(e_q)$ according to how well they engage the interest of u. At the heart of this task is the notion of *interestingness*. Following [16, 18], we formally define the *interestingness* modeling task as learning the mapping function:

$$f : U \times Q \times E \rightarrow \mathbb{R}, \tag{1}$$

where U is the set of all users, Q is the set of all queries, E is the set of all entities for all queries in Q, and $\mathbb{R}$ is the results of *interestingness* scores. The function $f(u, e_q, e_c)$ is the quantified degree of interest that a user $u \in U$ has in a candidate entity $e_c \in E$ when searching for a query $e_q \in Q$.[4]

3 LEARNING TO RECOMMEND FRAMEWORK

In this section, we introduce the proposed learning to recommend framework, which consists of two components: related entity finding and candidate entity ranking. The former is used to extract a set of candidate entities $\mathcal{R}(e_q)$ related to a query e_q that a user u is searching for, while the latter ranks the entities in $\mathcal{R}(e_q)$ according to how well they engage the interest of u. The two-phase framework has been widely used in search applications of learning to rank (LTR) paradigm for information retrieval [29], e.g., query suggestion [32], query rewriting [22], and entity relationship explaining [24]. This framework enables us to explore ways to optimize the performance of candidate finding, and gives us flexibility to choose the target to optimize for candidate ranking. In the following subsections, we describe the learning to recommend framework in detail.

3.1 Related Entity Finding

We consider the following three ways to find related entities for a given entity query e_q.

First, we use a knowledge graph (KG) to extract related entities for e_q following [4, 40]. A knowledge graph is a centralized repository which contains entities and their relations with other entities. Most relations in the knowledge graph are common facts, thus the set of entities linked with e_q in a knowledge graph can be directly extracted as related entities of it, which is denoted by

[4]We set $f(u, e_q, e_c) = 0$ for all $e_c \notin \mathcal{R}(e_q)$, where $\mathcal{R}(e_q)$ is the set of related entities of e_q.

$\mathcal{K}(e_q)$. The relatedness score between e_q and a candidate entity $e_c \in \mathcal{K}(e_q)$ is estimated as follows:

$$P_k(e_c|e_q) = \begin{cases} 1 & \text{if there exists a link between } e_c \text{ and } e_q \text{ in the knowledge graph} \\ 0 & \text{otherwise} \end{cases}. \quad (2)$$

Second, since the knowledge graph might be largely incomplete, we supplement related entities for e_q based on search session co-occurrences. Specifically, the top N_s entities co-occurred most frequently with e_q in the same search session are extracted as a set of candidates, which is denoted by $\mathcal{S}(e_q)$. The main reason is that, some users are engaged in issuing multiple queries during a search session [39], this search behavior potentially implies some underlying relations between entities. Given e_q and a candidate entity $e_c \in \mathcal{S}(e_q)$, we use pointwise mutual information to estimate the relatedness score between them:

$$P_s(e_c|e_q) = \frac{PMI(e_c, e_q)}{\sum_{e_c' \in \mathcal{S}(e_q)} PMI(e_c', e_q)}, \quad (3)$$

and $PMI(e_c, e_q)$ is defined as follows:

$$PMI(e_c, e_q) = \log \frac{cnt(e_c, e_q)}{cnt(e_c) \cdot cnt(e_q)}, \quad (4)$$

where $cnt(e_c, e_q)$ is the total number of search sessions in which e_c and e_q co-occur, and $cnt(e_q)$ is the total number of search sessions in which e_q occurs.

The search session co-occurrence based approach can be effective with sufficient data. However, it can only find related entities for e_q that have previously been observed in the search session log before. Hence, it is unable to estimate the relatedness for newly introduced entities due to the lack of co-occurrence information for them. To alleviate this problem, we use the method proposed by [8] to find related entities for e_q based on co-occurrences between e_q and potential target entities in Web documents. First, we collect all the entities co-occurred with e_q in a given corpus, which is denoted by $\mathcal{D}_r(e_q)$. Then, we rank the entities in $\mathcal{D}_r(e_q)$ based on the relatedness score of them. Finally, the top N_d ranked entities in $\mathcal{D}_r(e_q)$ are extracted as the set of related entities of e_q, which is denoted by $\mathcal{D}(e_q)$.

Specifically, the relatedness score between e_q and a candidate entity $e_c \in \mathcal{D}_r(e_q)$ is estimated as follows. For simplicity, we denote the relatedness score between e_q and e_c by $P_w(e_c|e_q)$, and it is calculated by:

$$P(e_c|e_q, T, R) \approx P(e_c|e_q) \cdot P(R|e_q, e_c) \cdot P(T|e_c), \quad (5)$$

where T is the set of entity types of e_q, and R is the relation described in the narrative. $P(e_c|e_q)$ is the context-independent co-occurrence model and estimated in the same manner using Eq. 3. $P(R|e_q, e_c)$ is the context-dependent co-occurrence model and estimated as follows:

$$P(R|e_q, e_c) = P(R|\theta_{qc}) = \prod_{t \in R} P(t|\theta_{qc})^{n(t,R)}, \quad (6)$$

where t is the term in R, and $n(t, R)$ is the number of times t occurs in R, and θ_{qc} is the co-occurrence language model which represents the relation between a pair of entities. $P(T|e_c)$ is the type filtering used to filter entities by type and estimated as follows:

$$P(T|e_c) = \begin{cases} 1 & \text{if } cat'(T) \cap cat(e_c) \neq \emptyset \\ 0 & \text{otherwise} \end{cases}, \quad (7)$$

where $cat(e_c)$ is a mapping function that maps an entity e_c to a set of categories of e_c, and $cat'(T)$ is the set of entity types by performing category expansion on T. For more details, please refer to [8].

Finally, we merge the entities from the three sources to obtain the set of related entities $\mathcal{R}(e_q)$ for e_q, that is:

$$\mathcal{R}(e_q) = \mathcal{K}(e_q) \cup \mathcal{S}(e_q) \cup \mathcal{D}(e_q). \tag{8}$$

3.2 Candidate Entity Ranking

In what follows, we first introduce the strategy to generate training data for learning ranking models. Then, we detail the features and the training process to learn the scoring function.

3.2.1 Generating Training Data. The acquisition of training data in our task is crucial but challenging especially for a large set of users, queries, and candidate entities. Although using a manually labeling method is a straightforward way, it is expensive and limited in quantity. Therefore, the training data are restricted to a small number of labeled instances and would remain constant if human labels are not consistently and continuously updated, making it difficult to meet the demand of large-scale, ever-evolving, real-world applications for the Web-scale entity recommendation system. To alleviate this problem, we propose to automatically generate large amounts of training data for this task by leveraging clickthrough data of a Web search engine. Clickthrough data are widely used in previous studies of search engines. For example, Gao *et al.* [17] proposed to learn the translation probability between phrases for improving retrieval effectiveness by using the clickthrough data. He *et al.* [22] proposed to generate learning targets from the clickthrough data for the task of query rewriting. Ma *et al.* [30] proposed to learn semantic relations from clickthrough data for the task of query suggestion. To facilitate the understandability of entity recommendations in a Web search engine, Huang *et al.* [25] proposed to train machine translation models for a special application of sentence compression by leveraging the clickthrough data.

In this paper, we use the click behavioral signal of a user to model if a recommended entity interests the user when searching for a query, because click has shown to be a salient observed behavioral signal of interestingness that has been well accepted in the information retrieval literature [9, 31]. Intuitively, if a user clicks on a recommended entity when searching for a query, it is reasonable to assume that she is interested in learning more about the clicked entity. Aggregated clicks from sufficient search logs can therefore serve as a proxy for interestingness. That is, for a given query, entities that attract one or more aggregated clicks are more interesting than the ones that attract no clicks. Gao *et al.* [18] also made the same assumption and proposed to use the click transitions between two documents as signals to model interestingness and achieved promising results. Specifically, we use the following strategy to generate learning targets for training ranking models from the query-entity clickthrough data.

$$y_{ijk} = \begin{cases} 1 & \text{if } click(u^i, e_q^j, e_c^k) > 0 \\ 0 & \text{otherwise} \end{cases}, \tag{9}$$

where $click(u^i, e_q^j, e_c^k)$ is the aggregated clicks of a recommended entity e_c^k for a query e_q^j that a user u^i is searching for, y_{ijk} is a grade label which indicates the interestingness of a triplet $\langle u^i, e_q^j, e_c^k \rangle$. In this way, we can generate the pointwise learning target y_{ijk} for each triplet $\langle u^i, e_q^j, e_c^k \rangle$. Furthermore, if necessary, the pairwise learning targets can also be easily generated from this data for a pairwise learning to rank approach following [12, 24], that is, a pairwise training example is generated if a recommended entity receives more aggregated clicks than the other.

3.2.2 Feature Functions. For each triplet $\langle u, e_q, e_c \rangle$, three sets of feature functions that correlate with the three aspects of serendipity are built: (1) relatedness features that are used to ensure the relatedness between two entities e_q and e_c; (2) interestingness features that are used to estimate

how well a candidate entity e_c could engage the interest of the user u when searching for the query e_q; and (3) unexpectedness features that are used to model how well e_c could help a user u discover some unexpected connections with e_q that u might not have otherwise discovered. Table 1 summarizes the definitions and descriptions of these features. In the following, we introduce the details of them.

Table 1. Features for entity ranking.

Feature set	#	Feature
Relatedness features	$f1$	$P_k(e_c\|e_q)$, relatedness in KG estimated by Eq. 2
	$f2$	$P_s(e_c\|e_q)$, relatedness in search sessions estimated by Eq. 3
	$f3$	$P_w(e_c\|e_q)$, relatedness in Web documents estimated by Eq. 5
	$f4$	$\text{sim}_c(e_q, e_c)$, content similarity estimated by Eq. 10
Interestingness features	$f5$	$CTR(u, e_q, e_c)$, $\langle u, e_q, e_c\rangle$-specific entity interest calculated by Eq. 11
	$f6$	$CTR(e_q, e_c)$, (e_q, e_c)-specific entity interest calculated by Eq. 12
	$f7$	$CTR(e_c)$, e_c-specific entity interest calculated by Eq. 13
	$f8$	$CTR_t(u, e_q, e_c)$, $\langle u, e_q, e_c\rangle$-specific topic interest calculated by Eq. 14
	$f9$	$CTR_t(e_q, e_c)$, (e_q, e_c)-specific topic interest calculated by Eq. 15
	$f10$	$CTR_t(e_c)$, e_c-specific topic interest calculated by Eq. 16
	$f11$	$\text{sim}_s(e_q, e_c)$, semantic similarity measured by Eq. 21
Unexpectedness features	$f12$	$R_a(u, e_q, e_c)$, $\langle u, e_q, e_c\rangle$-specific relation awareness estimated by Eq. 23
	$f13$	$\text{dis}(e_c, \mathcal{E}(u, e_q))$, $\langle u, e_q, e_c\rangle$-specific dissimilarity estimated by Eq. 26
	$f14$	$\text{dis}_a(e_c, \mathcal{E}(u, e_q))$, $\langle u, e_q, e_c\rangle$-specific average dissimilarity estimated by Eq. 27
	$f15$	$R_a(e_q, e_c)$, (e_q, e_c)-specific relation awareness estimated by Eq. 25
	$f16$	$\text{dis}(e_c, \mathcal{E}_w(e_q))$, (e_q, e_c)-specific dissimilarity estimated by Eq. 29
	$f17$	$\text{dis}_a(e_c, \mathcal{E}_w(e_q))$, (e_q, e_c)-specific average dissimilarity estimated by Eq. 30
	$f18$	$\text{div}(e_q)$, diversity of query e_q estimated by Eq. 32

Relatedness Features Four features are used to measure the relatedness between e_q and e_c.

Relatedness in KG This feature is computed by Eq. 2 which examines whether there exists a link between e_q and e_c in a knowledge graph. In our experiments, the links among two entities were represented as the relations between them. The relations between two entities were extracted from an in-house knowledge graph that is used by the Baidu Web search engine, which contains more than 4.9 billion relations between entities.

Relatedness in search sessions This feature is measured by Eq. 3 which indicates the mutual information between e_q and e_c in search sessions. In our experiments, search session logs from a 1-month period of the Baidu Web search engine were used. For confidentiality reasons, we are not able to provide details about the sessions.

Relatedness in Web documents This feature is calculated by Eq. 5 based on the co-occurrences between e_q and e_c in Web documents. Similar to [8], the articles of the largest Chinese online encyclopedia Baidu Baike[5] were used as Web documents to compute the co-occurrences between two entities. In our experiments, all Baike articles that correspond to more than 14 million entities were used.

Content similarity To further improve the relevance of the candidate entities, we measure the similarity between two entities by computing the similarity between their corresponding Baike

[5]http://baike.baidu.com/

articles. In our experiments, we used Latent Dirichlet Allocation (LDA) [6] to model documents. We trained a LDA model with 1,000 topics on more than 14 million Baike articles. Then, we can obtain a representation for a document d based on topics using the trained LDA model, that is, a topic vector v_d. Finally, the similarity between two entities e_q and e_c can be measured as the cosine similarity between the topic vectors of their corresponding Baike articles d_{e_q} and d_{e_c}:

$$\text{sim}_c(e_q, e_c) = \cos(v_{d_{e_q}}, v_{d_{e_c}}) = \frac{(v_{d_{e_q}})^T v_{d_{e_c}}}{\|v_{d_{e_q}}\| \|v_{d_{e_c}}\|}. \tag{10}$$

Interestingness Features To model interestingness, the following features are employed.

Entity interest The features of entity clickthrough rates (CTR) have shown to be very strong signals for recommendation [4]. Following [4], we also use the same three CTR features to model interestingness for a recommended entity e_c, which are calculated as follows:

$$CTR(u, e_q, e_c) = \frac{click(u, e_q, e_c) + \alpha}{impression(u, e_q, e_c) + \alpha + \beta}, \tag{11}$$

$$CTR(e_q, e_c) = \frac{click(e_q, e_c) + \alpha}{impression(e_q, e_c) + \alpha + \beta}, \tag{12}$$

$$CTR(e_c) = \frac{click(e_c) + \alpha}{impression(e_c) + \alpha + \beta}, \tag{13}$$

where $click(\cdot)$ is the number of clicks on a recommended entity e_c under different conditions, and $impression(\cdot)$ is the number of page impressions in which e_c is presented under different conditions. For example, $click(u, e_q, e_c)$ indicates the number of clicks on a recommended entity e_c issued by a user u when searching for the query e_q. α and β are introduced to estimate a smoothed CTR with smaller variance and more stability for a rare event [38] especially when $click(\cdot)$ is small. In our experiments, we set $\alpha = 0$ and $\beta = 100$.

Topic interest Entity interest can be effective for modeling interestingness with sufficient entity click logs. However, it can only estimate the interestingness scores for clicked entities that have previously been observed in search logs. Hence, it is unable to estimate the interestingness scores for unclicked or newly introduced entities due to no clicks were observed for them. To alleviate this problem, we use the categories of entities to model topic interest. Intuitively, if a user frequently clicks on a recommended entity e_c that is an instance of a category T_c when searching for a query e_q that is an instance of a category T_q, it is reasonable to assume that she is more likely interested in entities of the category T_c than entities of other categories when searching for queries of the category T_q. For example, after analyzing the entity click logs, we found that when a user searched for a film star, she may be more likely interested in entities of some categories such as celebrity, movie, song, and look-alike, but would have little interest in some categories such as physics and biology. This observation suggests that category information of entities could be used to uncover and model topic interest for entities, as it can help generalize the user interest data observed on clicked entities, and thus obtain better estimated interest on unclicked or newly introduced entities. Similar to entity interest features, we also consider three different topic interest features, which are defined as follows:

$$CTR_t(u, e_q, e_c) = \frac{\sum_{T_q \in cat(e_q)} \sum_{T_c \in cat(e_c)} click_t(u, T_q, T_c) + \alpha}{\sum_{T_q \in cat(e_q)} \sum_{T_c \in cat(e_c)} impression_t(u, T_q, T_c) + \alpha + \beta}, \tag{14}$$

$$CTR_t(e_q, e_c) = \frac{\sum_{T_q \in cat(e_q)} \sum_{T_c \in cat(e_c)} click_t(T_q, T_c) + \alpha}{\sum_{T_q \in cat(e_q)} \sum_{T_c \in cat(e_c)} impression_t(T_q, T_c) + \alpha + \beta}, \tag{15}$$

$$CTR_t(e_c) = \frac{\sum_{T_c \in cat(e_c)} click_t(T_c) + \alpha}{\sum_{T_c \in cat(e_c)} impression_t(T_c) + \alpha + \beta}, \tag{16}$$

where $cat(e)$ is a mapping function that maps an entity e to a set of categories of e. α and β are also introduced to estimate a smoothed CTR. In our experiments, we set $\alpha = 0$ and $\beta = 100$, and the top 2,000 most frequent categories of all entities in Baidu Baike were used. $click_t(\cdot)$ is the number of aggregated clicks on a set of recommended entities under different conditions, and $impression_t(\cdot)$ is the number of aggregated page impressions in which the set of recommended entities are presented under different conditions. For example, $click_t(u, T_q, T_c)$ is calculated as follows:

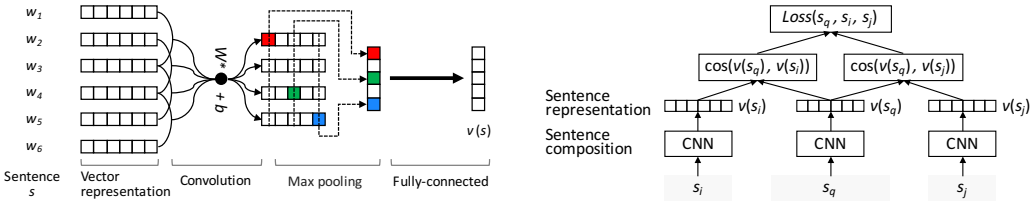
$$click_t(u, T_q, T_c) = \sum_{\hat{e_q} \in E(T_q)} \sum_{\hat{e_c} \in E(T_c)} click(u, \hat{e_q}, \hat{e_c}), \tag{17}$$

where $E(T_c)$ is a function that returns a set of entities that have a same category T_c.

Semantic similarity Semantics of texts has shown to be important for modeling interestingness [18]. To better capture and measure the underlying interest between any two entities, the semantic similarity between their corresponding descriptive sentences is calculated as their interestingness score. Huang *et al.* [24] proposed a pairwise ranking model which employs a convolutional neural network (CNN) to automatically learn semantic representations for sentences, and achieved promising results in the experiments. Following [24], we also employ a CNN to map a given sentence to its feature vector in a latent semantic space. The architecture of the CNN used for sentence composition is shown in Figure 2a. Given a sentence s that describes an entity e, a CNN is employed to obtain a distributed representation $v(s)$ of s.[6] First, words in the sentence (a sentence with 6 words is used for demonstrating) are transformed into vector representations via word embedding matrix, which capture semantic information of words. Second, in the convolutional layer, a set of filters is applied to a sliding window of length h (3 is used for demonstrating) over the sentence to extract a set of local features. To ensure that the filters can be applied to every element of the input matrix, zero-padding is applied to the input at both ends prior to convolution. The filters are automatically learned during the training phase of the neural network. Third, a max pooling layer performs max operation over a local neighborhood to retain only the most useful local features produced by the convolutional layer. Finally, the output of the max pooling layer is passed to a fully connected layer, which computes a non-linear transformation of these local features, here the sigmoid activation function is used. The network is trained using pairwise rankings as visualized in Figure 2b following [24]. In the following, we describe the training data acquisition and the training details.

To collect the pairwise training data for learning ranking models, we suppose that $click(e_q, e_c^i) > click(e_q, e_c^j)$, indicating that users are more interested in a recommended entity e_c^i when searching for e_q, as e_c^i receives more aggregated clicks than e_c^j. Given a function $S(e)$ that returns a descriptive sentence s of a given entity e, we can obtain the descriptive sentences for the entities e_q, e_c^i, and e_c^j. For simplicity, we denote $s_q = S(e_q)$, $s_i = S(e_c^i)$, and $s_j = S(e_c^j)$. Therefore, it is reasonable to assume that the interestingness between a pair of sentences s_q and s_i is higher than that of s_q and s_j, that

[6]Similar to [18], we also use the first sentence of a Baike article of e as the sentence s that describes e.



(a) CNN architecture for sentence composition.

(b) The architecture of pairwise ranking model.

Fig. 2. The architecture of semantic similarity model based on CNN.

is, $interest(s_q, s_i) > interest(s_q, s_j)$. In this way, we can construct a training set that consists of a set of triplets $\mathcal{P} = \{\langle s_q, s_i, s_j \rangle\}$.

Figure 2b shows the network architecture of the proposed pairwise ranking model. This network takes a triplet $\langle s_q, s_i, s_j \rangle$ as input that consists of three sentences, which are fed independently into three identical CNNs with shared architecture and parameters. Given a triplet $\langle s_q, s_i, s_j \rangle$, our goal is to learn a representation function $v(\cdot)$ for s_q, s_i, and s_j, and use the learned vector representations to calculate the cosine similarity between two sentences, such that given s_q, the more interesting sentence s_i can achieve higher similarity score:

$$\begin{aligned} &\cos(v(s_q), v(s_i)) > \cos(v(s_q), v(s_j)), \\ &\forall s_q, s_i, s_j \text{ such that } interest(s_q, s_i) > interest(s_q, s_j). \end{aligned} \tag{18}$$

Here the loss function is defined as the margin ranking loss:

$$Loss(s_q, s_i, s_j) = \max(0, 1 - \cos(v(s_q), v(s_i)) + \cos(v(s_q), v(s_j))). \tag{19}$$

Given the above loss function, the network is trained by minimizing an objective function over the training set $\mathcal{P}$. Specifically, given a set of triplets $\mathcal{P} = \{\langle s_q, s_i, s_j \rangle\}$, the representation function $v(\cdot)$ can be learned by minimizing the following objective function:

$$\min_{W} \sum_{\langle s_q, s_i, s_j \rangle \in \mathcal{P}} Loss(s_q, s_i, s_j) + \lambda \|W\|^2, \tag{20}$$

where λ is a regularization parameter used to improve the generalization of the learned ranking model. W is the parameters of the representation function $v(\cdot)$.

After the model is trained, given any two entities e_q and e_c, we can compute the semantic similarity between their corresponding descriptive sentences $s_q = S(e_q)$ and $s_c = S(e_c)$ as their interestingness score, which is calculated as follows:

$$\text{sim}_s(e_q, e_c) = \cos(v(s_q), v(s_c)) = \frac{v(s_q)^T v(s_c)}{\|v(s_q)\| \|v(s_c)\|}. \tag{21}$$

In our experiments, we used the entity panel logs from a 1-month period of the Baidu Web search engine to compute these interestingness features.

Unexpectedness Features To model unexpectedness, the following features are employed.

Relation awareness If a recommended entity e_c has already been discovered previously by a user u when searching for a query e_q, unexpectedness might be rarely encountered if e_c was recommended to u again when searching for e_q. This observation suggests that undiscovered relations between entities may have a higher chance to help a user get unexpected or surprisingly interesting discoveries, as they could bring completely new knowledge about a domain to this user. To better model unexpectedness, it is important to identify whether the relation between two

entities has already been discovered previously by a user. To this end, we use the historical click data to model the relation awareness of a user. Here search click log and entity click log are both used to collect the user historical click data. Given a query e_q, we first extract all entities in the titles of clicked documents from the search click log (denoted by $\mathcal{E}_{ct}(u, e_q)$) and all clicked entities from the entity click log (denoted by $\mathcal{E}_{ce}(u, e_q)$) issued by a user u, then we can obtain a set of discovered relations between e_q and other entities for the user u by merging them:

$$\mathcal{E}(u, e_q) = \mathcal{E}_{ct}(u, e_q) \cup \mathcal{E}_{ce}(u, e_q). \tag{22}$$

Then, given a query e_q and a recommended entity e_c, we can measure the relation awareness of a user u for these two entities as follows:

$$R_a(u, e_q, e_c) = \begin{cases} 1 & \text{if } e_c \in \mathcal{E}(u, e_q) \\ 0 & \text{otherwise} \end{cases}. \tag{23}$$

Intuitively, if the relation between a query e_q and a recommended entity e_c is well-known to many users, e_c might provide rare opportunities for a user to get unexpected discoveries when she searched for e_q. For example, if "Michelle Obama" was recommended to a user when she searched for "Barack Obama", it might have a lower chance to help this user get unexpected discoveries, as the two entities hold a well-known relation, and thus might be already well-known to this user and could not bring new knowledge to her. To better model unexpectedness, we further measure the degree of awareness of the relation between two entities e_q and e_c by the number of users who have clicked e_c previously when searching for e_q, which is denoted by $click_u(e_q, e_c)$, larger number means that the given relation has already been discovered by the majorities of users.

$$\mathcal{E}_w(e_q) = \{e_c \in \mathcal{R}(e_q) : click_u(e_q, e_c) \geq N_u\}, \tag{24}$$

where N_u is a certain threshold used to estimate whether the relation between e_q and e_c was well-known to most users.[7]

Then, we can measure the degree of awareness of the relation between a given query e_q and a recommended entity e_c as follows:

$$R_a(e_q, e_c) = \begin{cases} 1 & \text{if } e_c \in \mathcal{E}_w(e_q) \\ 0 & \text{otherwise} \end{cases}. \tag{25}$$

Content dissimilarity It has been shown that unexpectedness can be increased by recommending to a user those items that are semantically far from the items that are familiar to the user [26], or those items that depart from what she expects [1]. For this reason, we also introduce two dissimilarity features to better model unexpectedness. The first one is the distance of an recommended entity e_c from the set of discovered relations $\mathcal{E}(u, e_q)$ with respect to a user u and a query e_q, which is defined as follows:

$$\text{dis}(e_c, \mathcal{E}(u, e_q)) = \min_{e_k \in \mathcal{E}(u, e_q)} d(e_c, e_k), \tag{26}$$

where $d(\cdot)$ is a distance measure between two entities. Here the content similarity between two entities estimated by Eq. 10 is used to derive the distance between them, that is, $d(e_c, e_k) = 1 - \text{sim}_c(e_c, e_k)$. Intuitively, the higher $\text{dis}(e_c, \mathcal{E}(u, e_q))$ was, the more likely e_c would be unexpected to a user u when searching for the query e_q.

However, the recommended entities that are completely dissimilar might be deemed to be irrelevant by a user, and might be of little interest to the user. Hence, it is important to optimize the degree

[7] N_u could be set to a fixed number for all queries, or dynamic numbers related to different queries. In our experiments, we observed better results by using the latter method. Specifically, N_u was set as follows: $N_u = \frac{1}{2}(\min_{e_c \in \mathcal{R}(e_q)} click_u(e_q, e_c) + \max_{e_c \in \mathcal{R}(e_q)} click_u(e_q, e_c))$.

of dissimilarity with interestingness. To alleviate this problem, we take a user's interestingness distribution on $\mathcal{E}(u, e_q)$ into consideration, and calculate the weighted average dissimilarity as follows:

$$\text{dis}_a(e_c, \mathcal{E}(u, e_q)) = \sum_{e_j \in \mathcal{E}(u, e_q)} C\bar{T}R(u, e_q, e_j) \cdot d(e_j, e_c), \tag{27}$$

where $C\bar{T}R(u, e_q, e_j)$ is the normalization of $CTR(u, e_q, e_j)$ which satisfies the following constraints:

$$\sum_{e_j \in \mathcal{R}(e_q)} C\bar{T}R(u, e_q, e_j) = 1. \tag{28}$$

In the same way, we can calculate the dissimilarity and the weighted average dissimilarity of an recommended entity e_c from the set of relations of e_q that have already been discovered by the majorities of users in the past, respectively:

$$\text{dis}(e_c, \mathcal{E}_w(e_q)) = \min_{e_k \in \mathcal{E}_w(e_q)} d(e_c, e_k), \tag{29}$$

$$\text{dis}_a(e_c, \mathcal{E}_w(e_q)) = \sum_{e_j \in \mathcal{E}_w(e_q)} C\bar{T}R(e_q, e_j) \cdot d(e_j, e_c), \tag{30}$$

where $C\bar{T}R(e_q, e_j)$ is the normalization of $CTR(e_q, e_j)$ which satisfies the following constraints:

$$\sum_{e_j \in \mathcal{R}(e_q)} C\bar{T}R(e_q, e_j) = 1. \tag{31}$$

Diversity Above dissimilarity features might promote more diverse entities in the recommendation list for a given query e_q. However, if there is no need to provide diverse recommendations for e_q, these features might decrease the performance of the system. To alleviate this problem, we use click entropy to model the diversity of a given query e_q. A similar idea is also proposed in [42], where diversity is introduced into a music recommendation system to help counteract user satiety with homogeneous recommendations. Specifically, the diversity of a query in the context of entity recommendation is defined as follows:

$$\text{div}(e_q) = ClickEntroy(e_q) = \sum_{e_i \in C(e_q)} -P(e_i|e_q)\log_2 P(e_i|e_q), \tag{32}$$

where $ClickEntroy(e_q)$ is the click entropy [11] of query e_q. $C(e_q)$ is the set of entities clicked on query e_q. $P(e_i|e_q)$ is the percentage of the clicks on a recommended entity e_i among all the clicks on query e_q, that is,

$$P(e_i|e_q) = \frac{click(e_q, e_i)}{\sum_{e_j \in C(e_q)} click(e_q, e_j)}, \tag{33}$$

where $click(e_q, e_i)$ is the number of clicks on a recommended entity e_i on query e_q in the entity click log.

Click entropy is a direct indication of query click variation [11]. If all users click only one same entity on query e_q, then we have $ClickEntroy(e_q) = 0$. Smaller click entropy indicates that the majorities of users agree with each other on a small number of entities. In such cases, we should generate more homogeneous recommendations rather than more diverse recommendations for this query. On the other hand, large click entropy indicates that many entities were clicked for a given query. This means that there may be two situations: (1) a user has clicked several entities on this query to satisfy her diverse information needs; and (2) different users have clicked different entities on this query to explore diverse recommendations. In such cases, we should highlight more diverse recommendations for this query.

3.2.3 Training. Here we use stochastic Gradient Boosted Decision Tree (GBDT) [14, 15] as the learning to rank framework with the given features. The GBDT algorithm provides relative influence of variables [14] which can help further investigate the impact of different features. The higher the relative influence value of a feature, the greater contribution it makes in the model building phase, thus the more important it is in the prediction process. The hyperparameters of GBDT: number of trees, number of nodes, shrinkage rate, and sampling rate, are tuned on an independent validation set.[8]

The ranking model is trained by minimizing an objective function over the training set. More specifically, given a training set that consists of a triplet with its corresponding interestingness label, that is, $\mathcal{H} = \{(\langle u^i, e_q^j, e_c^k\rangle, y_{ijk})\}$, our goal is to learn from $\mathcal{H}$ a scoring function $f(\cdot)$ that measures the probability of interestingness $f(u^i, e_q^j, e_c^k)$ for a given triplet $\langle u^i, e_q^j, e_c^k\rangle$. The scoring function $f(\cdot)$ can be learned by minimizing the following objective function:

$$\hat{f} = \text{argmin } Loss(\mathcal{H}), \tag{34}$$

where the loss function $Loss(\mathcal{H})$ is a cross entropy loss defined as follows:

$$\begin{aligned} Loss(\mathcal{H}) &= -\log \prod_{(\langle u^i, e_q^j, e_c^k\rangle, y_{ijk})\in\mathcal{H}} f(u^i, e_q^j, e_c^k)^{y_{ijk}} \cdot (1 - f(u^i, e_q^j, e_c^k))^{(1-y_{ijk})} \\ &= -\sum_{(\langle u^i, e_q^j, e_c^k\rangle, y_{ijk})\in\mathcal{H}} y_{ijk} \cdot \log f(u^i, e_q^j, e_c^k) + (1 - y_{ijk}) \cdot \log(1 - f(u^i, e_q^j, e_c^k)). \end{aligned} \tag{35}$$

4 EXPERIMENTAL SETUP

This section describes the datasets, baseline methods, and evaluation metrics in our experiments.

4.1 Experimental Data

We used the clickthrough data and search logs of the Baidu Web search engine to extract data for learning and evaluating the ranking models. First, logs from a 1-month period were used to extract the features defined in Section 3.2.2. Second, to prevent models from overfitting on the training data, logs from the next 1-month period were used to generate the pointwise learning targets using the method described in Section 3.2.1. In this way, the historical user behavior data $\mathcal{H}_{all}$ could be obtained. Due to the huge search traffic, it is impractical to use all the historical data for training, since it would require considerable running time. Therefore, we randomly sampled a small portion[9] of data from $\mathcal{H}_{all}$ as the training set $\mathcal{H}$, which consists of 106,712,857 instances of $(\langle u^i, e_q^j, e_c^k\rangle, y_{ijk})$. In $\mathcal{H}$, there are 35,413,304 positive instances and 71,299,553 negative instances.[10] For validation set and test set, we randomly sampled a validation set $\mathcal{H}_v$ from a small portion of logs during a 1-day period, and a test set $\mathcal{H}_t$ from a small portion of logs during a different 6-day period. $\mathcal{H}_v$ was used to tune the parameters for the ranking models, and $\mathcal{H}_t$ was used to perform

[8]We used a grid search to determine the optimal parameters of the ranking models in the feasible space of selected parameters. We selected the number of trees among $\{500, 1000, \cdots, 4000\}$, learning rate among $\{0.001, 0.005, \cdots, 0.5\}$, and tree depth among $\{3, 4, 5, 6\}$. We tuned the GBDT models on the validation set, and the models with the parameter settings that best performed on the validation set were selected. The parameters used in the experiments are as follows: number of trees = 2500, tree depth = 5, learning rate = 0.2. To prevent the models from overfitting, sampling rate with the probability of 0.5 is applied.

[9]We are not able to provide details about the sampled percentages for confidentiality reasons.

[10]Since the label distribution in the historical data is highly imbalanced, i.e., $\mathcal{H}_{all}$ contains much more negative instances than positive instances, the instances were randomly sampled under the constraint that the amount of the negative instances cannot exceed three times (3×) the amount of positive instances. In which, if the interestingness label y_{ijk} of a triplet $\langle u^i, e_q^j, e_c^k\rangle$ equals 1, it is a positive instance. Otherwise, it is a negative instance.

the offline evaluation of the ranking models. $\mathcal{H}_v/\mathcal{H}_t$ consists of 8,311,168/48,117,012 instances of $(\langle u^i, e_q^j, e_c^k \rangle, y_{ijk})$, respectively.

We also performed an online evaluation by conducting an online controlled experiment. To this end, we randomly sampled 100,000 queries from the query logs of the Baidu Web search engine, and extracted the queries that can be linked with their corresponding entities in a knowledge graph by using the entity linking method proposed in [21] based on the search context information. Then, a test set of 33,836 entity queries was obtained and used in the online evaluation, which is denoted by $\mathcal{T}$. For each entity in $\mathcal{T}$, we used the method described in Section 3.1 to find related entities for it, in which the N_s and N_d used in our experiments were set to 1,000. For online evaluation, we conducted the online controlled experiments during a 4-day period, and a total of 5,283,120 users were involved in the experiments.

4.2 Baseline Methods

To evaluate the proposed method, five approaches are selected as baseline methods for comparison. The first four baseline methods are selected following [4], and the fifth baseline is proposed by [4].

Random This baseline method is a naive algorithm which randomly ranks the related entities with respect to a user and the query.

Co-click The co-click signal by itself has been shown to be effective and the strongest baseline method for entity recommendation [4]. The basic idea is that if two entities are frequently co-clicked, it is reasonable that an entity should be recommended for another entity that is issued as a query. Specifically, co-click ranks the candidate entities based on the number of their co-occurrences with a given query in the click log.

CTR-model This baseline only utilizes the CTR features of entities, i.e., $f5$, $f6$, and $f7$, to build the ranking model following [4].

Production This baseline represents the entity recommendation approach currently employed by the Baidu Web search engine, which reflects the state-of-the-art in the context of entity recommendation.

TEM This baseline is a personalized entity recommendation method proposed by [4]. However, this method has a limitation that it can be applied only when the target domain is known, as it requires a rich set of domain-dependent entity features to be successful in recommending relevant and personalized entities. By contrast, we do not focus on any particular domain, and aim to provide entity recommendations with serendipity for any type of entity. Therefore, domain-independent features are crucial for building our proposed recommendation model, making it difficult to make a straightforward comparison with the method proposed by [4]. Hence, for comparison purposes, we omit a set of features specific to film entities used in [4], and only choose the equivalent domain-independent features[11] in the experiments.

4.3 Evaluation Metrics

We conducted both offline evaluation and online evaluation to examine the effectiveness of our proposed learning to recommend framework.

4.3.1 Offline Evaluation. The offline evaluation aims to assess the quality of a ranked list of entities related to a given query e_q issued by a user u. Two metrics are used for offline evaluation: discounted cumulative gain (DCG) and mean reciprocal rank (MRR). To investigate the effectiveness with respect to different rank positions, we adopt DCG and MRR at various cut-off points to measure

[11] Specifically, the features $f1$, $f2$, $f5$, $f6$, and $f7$ in Table 1 are selected for use.

the quality of the ranked entities. In our experiments, we will report different rank positions in $\{1, 5, 10, 15\}$, respectively.

DCG is a measure of ranking quality in information retrieval, and has been widely used to assess the relevance of search results [27]. For a ranked list of entities, we use the following formulation to compute the DCG accumulated at a particular rank position p, which places stronger emphasis on retrieving relevant entities:

$$DCG_p = \sum_{i=1}^{p} \frac{2^{rel_i^{uq}} - 1}{\log_2(i+1)}, \tag{36}$$

where $rel_i^{uq} \in \{0, 1\}$ is the binary relevance (1 for relevant and 0 for irrelevant) of the entity ranked at position i with respect to user u and query e_q. In our experiments, the overall DCG is calculated by averaging over all the test instances in the test set $\mathcal{H}_t$, and we denote this averaged DCG value at a particular rank position p using the same symbol DCG_p.

We also use the standard mean reciprocal rank (MRR) to evaluate the ranking quality following [4, 40]. The reciprocal rank of a ranked list is the multiplicative inverse of the rank at which the first relevant entity is retrieved. The top-p MRR score is the average of the reciprocal ranks of the ranked lists for each test instance in the test set $\mathcal{H}_t$:

$$MRR_p = \frac{1}{|\mathcal{H}_t|} \sum_{j=1}^{|\mathcal{H}_t|} \frac{1}{rank(e_j^{uq})}, \tag{37}$$

where e_j^{uq} is the ground truth entity in the j-th test with respect to user u and query e_q, and $rank(e_j^{uq})$ represents the rank of e_j^{uq} determined by a certain ranking model, which will be ∞ if the ground truth entity e_j^{uq} is not in the top-p ranked results.

4.3.2 Online Evaluation. To quantify the user engagement of a recommendation model, a suitable way is to examine the clickthrough rate (CTR) of entities recommended by this model. CTR is an effective metric used to evaluate the performance of an online service, such as a backend recommendation system [10], sponsored search advertising [20], and the utility of a Web search result page [33]. Online controlled experiments are widely used to make data-driven decisions by running A/B tests[12] at search engine companies [28]. In our experiments, the goal of an online controlled experiment is to identify enhancements to the recommendation models that increase or maximize the CTR for the recommendation results. From the test results, we can evaluate the user engagement of different models by comparing their CTR values. Higher CTR would predict higher user engagement.

In an online controlled experiment, users who are searching for the given queries are randomly and evenly split into either a control or treatment group. In which, the split is carried out in a persistent manner, which ensures that a user is divided into the same group and receives the same experience in multiple visits. We implement five control groups for individual baseline methods and a treatment group for our proposed method. In each group, when a user u is searching for a query e_q in the test set $\mathcal{T}$, a corresponding recommendation model is employed to generate a ranked list of entities with respect to user u and query e_q. Then, the list of entities is recommended on the right panel of the search engine result page[13] for user u, and individual user behavior and interactions with each recommended entity are instrumented. Finally, we run the experiment for a period of time (e.g., three days) and the user behavior data are collected. The user engagement

[12]https://en.wikipedia.org/wiki/A/B_testing

[13]Note that, in the controlled experiments, the search result pages of all control and treatment groups with respect to a same query differ only in the region of their entity recommendation results.

of a recommendation model involved in the controlled experiment is evaluated by the CTR of its corresponding group, which is computed as follows:

$$CTR = \frac{\sum_{e_q \in \mathcal{T}} \sum_{e_c \in \mathcal{R}(e_q)} click(e_q, e_c)}{\sum_{e_q \in \mathcal{T}} \sum_{e_c \in \mathcal{R}(e_q)} impression(e_q, e_c)}, \tag{38}$$

where $\mathcal{R}(e_q)$ is the set of recommended entities for a query $e_q \in \mathcal{T}$, $click(e_q, e_c)$ is the number of clicks on a certain recommended entity e_c on query e_q, and $impression(e_q, e_c)$ is the number of page impressions in which e_c is presented when searching for e_q.

We implemented online controlled experiments for our proposed method and all the baseline methods. In our experiments, for a certain group (a control group or a treatment group), a corresponding recommendation model was employed to generate a ranked list of entities[14] for each query e_q in the test set $\mathcal{T}$ from the set of related entities $\mathcal{R}(e_q)$ of e_q.

5 RESULTS AND ANALYSIS

We report empirical results, model comparisons, and model analysis in this section. To comply with the company's non-disclosure policy, we normalized the raw CTR values by the maximum CTR value, and reported only relative CTR values. As depicted in the evaluation results in each table and figure, the statistical significance is indicated using a paired two-tailed t-test, and boldface indicates the highest score with respect to each metric. For CTR results, the t-test is performed on the raw CTR values. For offline evaluation results, we omit MRR_1 due to it is equivalent to DCG_1 in our experiments.

5.1 Model Comparisons

In this subsection, we compare our proposed model with baseline methods.

First, we investigate the quality of recommendation results generated by each method. Table 2 shows the offline evaluation results of the proposed method and five baseline methods. It is observed from the results that the other five methods all perform significantly better than the *Random* method. However, *Co-click* outperforms *Random* by a smaller margin than the other four methods do. The main reason is that, *Co-click* is based on the number of entity co-occurrences in the click log, which would be effective for clicked entities that have previously been observed in the user log before. Therefore, it inevitably suffers from the cold start problem for unclicked or newly introduced entities. It is also observed that *CTR-model*, *Production*, and *TEM* achieve significant improvement over the *Random* and *Co-click* methods. In addition, *CTR-model*, *Production*, and *TEM* yield similar results, as they generate entity recommendations mainly based on the CTR signals. This shows the great effectiveness of the user clickthrough information derived from the entity click log on building an entity recommendation system. Furthermore, the results show that, *LTRC* significantly outperforms all baseline methods by a large margin in terms of both DCG and MRR at various rank positions, which demonstrates that *LTRC* can provide entity recommendations of the highest quality to users. This also confirms the effectiveness of the proposed framework, which is designed to boost the serendipity performance by leveraging the signals from the three sets of relatedness, interestingness, and unexpectedness features that correlate with the three aspects of serendipity.

Second, we investigate the user engagement of recommendation results generated by each method. Figure 3 shows the online evaluation results of the proposed method and five baseline

[14]In our experiments, the number of related entities presented in each page impression changes dynamically according to a user's screen resolution, which is normally ranging from 9 to 16.

Table 2. The offline evaluation results of the proposed method and five baseline methods. $^{▲*}$ indicates statistical significance over all the other results in the same column using t-test for $p < 0.001$.

	DCG				MRR		
	DCG_1	DCG_5	DCG_{10}	DCG_{15}	MRR_5	MRR_{10}	MRR_{15}
Random	0.1230	0.3519	0.4893	0.5258	0.2643	0.3108	0.3195
Co-click	0.2308	0.4848	0.5898	0.6146	0.3868	0.4211	0.4269
CTR-model	0.2701	0.5319	0.6273	0.6464	0.4299	0.4607	0.4651
Production	0.2712	0.5276	0.6222	0.6450	0.4281	0.4587	0.4639
TEM	0.2725	0.5349	0.6295	0.6484	0.4326	0.4631	0.4674
LTRC	**0.2821**$^{▲*}$	**0.5490**$^{▲*}$	**0.6406**$^{▲*}$	**0.6575**$^{▲*}$	**0.4447**$^{▲*}$	**0.4742**$^{▲*}$	**0.4780**$^{▲*}$

methods. From this figure, we can see that *LTRC* significantly outperforms all baseline methods and achieves the highest CTR score, which demonstrates that *LTRC* can significantly improve user engagement against all baseline methods by recommending entities that are of most interest to users.

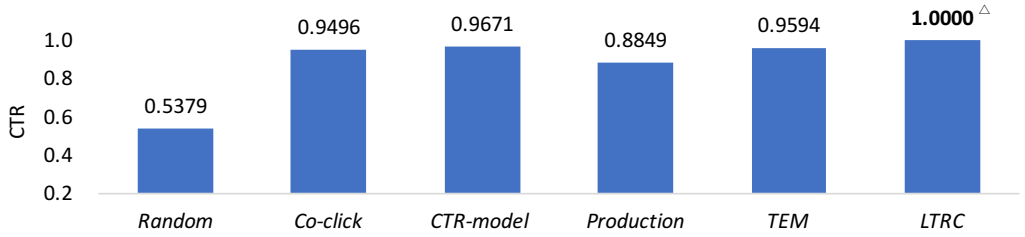


Fig. 3. CTR results of the proposed method and five baseline methods. $^{△}$ indicates statistical significance over all the other results using t-test for $p < 0.05$.

5.2 Model Analysis

In this subsection, we provide an analysis of the proposed model *LTRC* and the impact of different features used in this model.

To examine the effectiveness of the proposed model, variations of learning to recommend models with different combinations of feature sets are implemented for comparison. For simplicity, we denote the features in the sets of "Relatedness features", "Interestingness features", and "Unexpectedness features" described in Table 1 using the symbols R, I, and U, respectively. All the variations are listed as follows.

LTRC$_R$***, LTRC***$_I$***, LTRC***$_U$ These models are trained on individual feature sets R, I, and U, respectively.

LTRC$_{R+I}$***, LTRC***$_{R+U}$***, LTRC***$_{I+U}$ These models are trained on different combinations of two feature sets, e.g., $R + I$ denotes the combination of two feature sets R and I.

LTRC This model utilizes all the three feature sets, i.e., R, I, and U.

Table 3 shows the evaluation results of each model. From the results, we make the following observations after investigating the effectiveness of individual models.

First, $LTRC_I$ achieves the highest scores in comparison with the other two variations $LTRC_U$ and $LTRC_R$, which shows that the set of interestingness features is the most powerful feature set

Table 3. The offline evaluation results of different models. To measure the improvement of *LTRC* over all the other models, we depict the statistical significance over all the other results in the same column using t-test for $p < 0.01$ with ▲, and for $p < 0.05$ with △.

	DCG				MRR		
	DCG_1	DCG_5	DCG_{10}	DCG_{15}	MRR_5	MRR_{10}	MRR_{15}
$LTRC_R$	0.1687	0.4101	0.5335	0.5644	0.3174	0.3586	0.3659
$LTRC_I$	0.2780	0.5423	0.6355	0.6532	0.4390	0.4690	0.4730
$LTRC_U$	0.2185	0.4822	0.5865	0.6102	0.3809	0.4151	0.4206
$LTRC_{R+I}$	0.2788	0.5443	0.6370	0.6545	0.4407	0.4705	0.4745
$LTRC_{R+U}$	0.2277	0.4905	0.5934	0.6164	0.3894	0.4230	0.4284
$LTRC_{I+U}$	0.2811	0.5485	0.6398	0.6569	0.4441	0.4735	0.4773
LTRC	**0.2821**▲	**0.5490**△	**0.6406**▲	**0.6575**▲	**0.4447**▲	**0.4742**▲	**0.4780**▲

in our model. Moreover, the performance of the model is significantly decreased after removing the feature set I (from *LTRC* to $LTRC_{R+U}$), this further confirms that the set of interestingness features plays the most important role in improving the quality of recommendation results. The main reason is that, compared with the other two feature sets, the set of interestingness features is more effective in not only capturing underlying correlations among users, queries, and entities, but also in uncovering interesting connections between entities. This also demonstrates that user click signals derived from the entity click log are important for modeling interestingness, as all interestingness features are learned from the entity clickthrough data.

Second, the set of unexpectedness features can help to improve the quality of recommendation results, as it obtains a significant improvement in model performance after adding feature set U (from $LTRC_{R+I}$ to *LTRC*, and the paired t-test between them has p-value < 0.01). This confirms the effectiveness of introducing unexpectedness into an entity recommendation system. The main reason is that, the unexpectedness features in U can help to avoid finding and recommending entities that are already known by most users or stray too far from what they expect, and bring in new signals for recommendation which are not presented in the other two feature sets.

Third, the set of relatedness features can also help to improve the recommendation quality, as the paired t-test performed between $LTRC_{I+U}$ and *LTRC* shows p-value < 0.05. However, the difference between the performance of $LTRC_{I+U}$ and *LTRC* is very marginal. Furthermore, we do observe that feature set R makes relatively lower performance improvement in comparison with the other two feature sets I and U. This is because the features in R purely rely on the content information of entities, while the features in I and U make use of rich user click information. This observation suggests that it is important to use the user click information to learn effective features for uncovering the interesting or underlying connections among users, queries, and entities that cannot be modeled by content information. It also shows the great potential of the user click information on building an entity recommendation system.

To further investigate the impact of individual features, we calculate the relative importance scores for all the features. We assign the most influential feature a score of 100, and scale the importance scores of other features accordingly. Figure 4 shows the relative importance scores for all the features. From the top 10 features, we can see that these features all play important roles in the model *LTRC*, and the learning to recommend framework can benefit from the combination of the features in R, I, and U. The superiority of three unexpectedness features ($f18$, $f17$, and $f16$) further suggests that it is important to introduce unexpectedness into an entity recommendation model. The user-specific features ($f8$, $f14$, $f5$, $f13$, and $f12$) used here are accurate predictors of

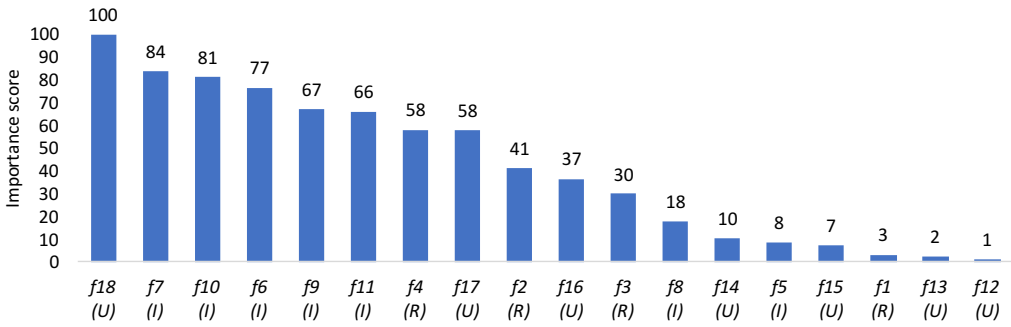


Fig. 4. Importance scores of all the features. The feature sets of individual features are depicted under them.

personalization, but the influence of them is much lower. The main reason is that, these features are only available for user-query-entity triples that have been observed in the past in search logs. Therefore, the coverage of them is limited to the clicked ones, making them hardly perform well on the click-absent ones.

6 RELATED WORK

Previous work that is the closest to our task is recommending related entities with respect to a user and/or a query. Previous work mainly focus on providing a user with the most relevant [5] and/or personalized [4, 13, 40] entity recommendations that score highly against the query and/or the user's preference. However, none of them address the problem of boosting recommendation performance in terms of serendipity that has been demonstrated to be critical for a satisfying and engaging user experience of recommendation systems [1, 37]. Moreover, the methods proposed in [4, 5, 40] have a limitation that they can be applied only when the target domain is known, as they require a rich set of domain-dependent entity features derived from a knowledge graph to be successful in recommending relevant and/or personalized entities. By contrast, we do not focus on any particular domain, and aim to provide entity recommendations with serendipity for any type of entity. Finally, Fernández-Tobías and Blanco [13] proposed a domain-agnostic method that only exploits query log data to proactively recommend relevant entities for a user's search session in a personalized fashion. Our work is different in that we aim to provide entity recommendations with serendipity related to both a given query and a user, rather than recommend entities for a user's search session.

A research topic closely related to our work is the task of increasing serendipity of a traditional item-based recommender system, although the definitions of serendipity and the research tasks are quite different from our work. Adamopoulos and Tuzhilin [1] proposed to increase serendipity by recommending to users those items that depart from what they expect. Iaquinta *et al.* [26] proposed to increase serendipity by recommending novel items that are semantically far from a user's profile. Zhang *et al.* [42] proposed to inject serendipity into recommendations whilst limiting the impact on accuracy. Although the definitions of serendipity are not directly adaptable in the context of entity recommendation, the ideas proposed to increase serendipity in these studies can be adopted as features for our proposed learning to recommend framework, which have been shown to be strong signals that are effective for finding unexpected items.

7 CONCLUSIONS

In this paper, we study the problem of recommending entities with serendipity related to both a given query and a user. We define the concept of serendipity in the context of entity recommendation. We propose a learning to recommend framework which employs three different sets of features that correlate with the three aspects of serendipity. Extensive experiments are conducted on large-scale, real-world datasets collected from a widely used commercial Web search engine. The experiments show that our method significantly outperforms multiple strong baseline methods. In addition, online controlled experiments conducted on a commercial Web search engine demonstrate that our method significantly improve user engagement against all baseline methods. This further confirms the effectiveness of the proposed framework.

As future work, we plan to boost the performance of the proposed learning to recommend framework in two directions, that are, exploring more ways to find related entities and introducing more features to rank the entities that uncover some interesting and unexpected connections with respect to a given query and a user. We are also interested in captioning the recommended entities, particularly the serendipitous entities, to help users better figure out the connections between the recommended entities and a query as suggested by [24, 25].

ACKNOWLEDGMENTS

This research is supported by the National Basic Research Program of China (973 program No. 2014CB340505). The authors would like to thank the anonymous reviewers for their insightful comments. We also thank Wei Zhang for his assistance to prepare the experimental data. We wish to thank Mingming Sun for his comments on an earlier draft of this paper.

REFERENCES

[1] Panagiotis Adamopoulos and Alexander Tuzhilin. 2014. On unexpectedness in recommender systems: or how to better expect the unexpected. *ACM Trans. Intell. Syst. Technol.* 5, 4 (2014), 54:1–54:32.
[2] Nitish Aggarwal, Peter Mika, Roi Blanco, and Paul Buitelaar. 2015. Insights into entity recommendation in web search. In *ISWC*.
[3] Paul André, M.C. Schraefel, Jaime Teevan, and Susan T Dumais. 2009. Discovery is never by chance: designing for (un) serendipity. In *C&C*. 305–314.
[4] Bin Bi, Hao Ma, Bo-June (Paul) Hsu, Wei Chu, Kuansan Wang, and Junghoo Cho. 2015. Learning to recommend related entities to search users. In *WSDM*. 139–148.
[5] Roi Blanco, Berkant Barla Cambazoglu, Peter Mika, and Nicolas Torzec. 2013. Entity recommendations in web search. In *ISWC*. 33–48.
[6] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research* 3, 4-5 (2003), 993–1022.
[7] Ilaria Bordino, Yelena Mejova, and Mounia Lalmas. 2013. Penguins in sweaters, or serendipitous entity search on user-generated content. In *CIKM*. 109–118.
[8] Marc Bron, Krisztian Balog, and Maarten de Rijke. 2009. Related entity finding based on co-occurrence. In *TREC*.
[9] Mark Claypool, Phong Le, Makoto Wased, and David Brown. 2001. Implicit interest indicators. In *Proceedings of the 6th international conference on Intelligent user interfaces*. 33–40.
[10] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, and Taylor Van Vleet. 2010. The YouTube video recommendation system. In *RecSys*. 293–296.
[11] Zhicheng Dou, Ruihua Song, and Ji-Rong Wen. 2007. A large-scale evaluation and analysis of personalized search strategies. In *WWW*. 581–590.
[12] Zhicheng Dou, Ruihua Song, Xiaojie Yuan, and Ji-Rong Wen. 2008. Are click-through data adequate for learning web search rankings?. In *CIKM*. 73–82.
[13] Ignacio Fernández-Tobías and Roi Blanco. 2016. Memory-based recommendations of entities for web search users. In *CIKM*. 35–44.
[14] Jerome H Friedman. 2000. Greedy function approximation: a gradient boosting machine. *Annals of Statistics* 29 (2000), 1189–1232.
[15] Jerome H Friedman. 2002. Stochastic gradient boosting. *Computational Statistics & Data Analysis* 38, 4 (2002), 367–378.

[16] Michael Gamon, Arjun Mukherjee, and Patrick Pantel. 2014. Predicting interesting things in text. In *COLING*. 1477–1488.
[17] Jianfeng Gao, Xiaodong He, and Jian-Yun Nie. 2010. Clickthrough-based translation models for web search: from word models to phrase models. In *CIKM*. 1139–1148.
[18] Jianfeng Gao, Patrick Pantel, Michael Gamon, Xiaodong He, and Li Deng. 2014. Modeling interestingness with deep neural networks. In *EMNLP*. 2–13.
[19] Mouzhi Ge, Carla Delgado-Battenfeld, and Dietmar Jannach. 2010. Beyond accuracy: evaluating recommender systems by coverage and serendipity. In *RecSys*. 257–260.
[20] Thore Graepel, Joaquin Q Candela, Thomas Borchert, and Ralf Herbrich. 2010. Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine. In *ICML-10*. 13–20.
[21] Xianpei Han, Le Sun, and Jun Zhao. 2011. Collective entity linking in web text: a graph-based method. In *SIGIR*. 765–774.
[22] Yunlong He, Jiliang Tang, Hua Ouyang, Changsung Kang, Dawei Yin, and Yi Chang. 2016. Learning to rewrite queries. In *CIKM*. 1443–1452.
[23] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. 2004. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)* 22, 1 (2004), 5–53.
[24] Jizhou Huang, Wei Zhang, Shiqi Zhao, Shiqiang Ding, and Haifeng Wang. 2017. Learning to explain entity relationships by pairwise ranking with convolutional neural networks. In *IJCAI*. 4018–4025.
[25] Jizhou Huang, Shiqi Zhao, Shiqiang Ding, Haiyang Wu, Mingming Sun, and Haifeng Wang. 2016. Generating recommendation evidence using translation model. In *IJCAI*. 2810–2816.
[26] Leo Iaquinta, Marco De Gemmis, Pasquale Lops, Giovanni Semeraro, Michele Filannino, and Piero Molino. 2008. Introducing serendipity in a content-based recommender system. In *HIS*. 168–173.
[27] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.* 20, 4 (2002), 422–446.
[28] Ron Kohavi, Alex Deng, Brian Frasca, Roger Longbotham, Toby Walker, and Ya Xu. 2012. Trustworthy online controlled experiments: five puzzling outcomes explained. In *SIGKDD*. 786–794.
[29] Tie-Yan Liu. 2009. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval* 3, 3 (2009), 225–331.
[30] Hao Ma, Haixuan Yang, Irwin King, and Michael R. Lyu. 2008. Learning latent semantic relations from clickthrough data for query suggestion. In *CIKM*. 709–718.
[31] Florian Mueller and Andrea Lockerd. 2001. Cheese: tracking mouse movement activity on websites, a tool for user modeling. In *CHI'01 extended abstracts on Human factors in computing systems*. 279–280.
[32] Umut Ozertem, Olivier Chapelle, Pinar Donmez, and Emre Velipasaoglu. 2012. Learning to suggest: a machine learning framework for ranking query suggestions. In *SIGIR*. 25–34.
[33] Ashok Kumar Ponnuswami, Kumaresh Pattabiraman, Qiang Wu, Ran Gilad-Bachrach, and Tapas Kanungo. 2011. On composition of a federated web search result page: using online users to provide pairwise preference for heterogeneous verticals. In *WSDM*. 715–724.
[34] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor. 2011. Recommender systems handbook. (2011).
[35] Upendra Shardanand and Pattie Maes. 1995. Social information filtering: algorithms for automating "word of mouth". In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 210–217.
[36] Beerud Sheth and Pattie Maes. 1993. Evolving agents for personalized information filtering. In *Proceedings of the Ninth Conference on Artificial Intelligence for Applications*. 345–352.
[37] Linqi Song, Cem Tekin, and Mihaela Van Der Schaar. 2014. Clustering based online learning in recommender systems: a bandit approach. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 4528–4532.
[38] Xuerui Wang, Wei Li, Ying Cui, Ruofei Zhang, and Jianchang Mao. 2011. Click-through rate estimation for rare events in online advertising. *Online Multimedia Advertising: Techniques and Technologies* (2011), 1–12.
[39] Zi Yang and Eric Nyberg. 2015. Leveraging procedural knowledge for task-oriented search. In *SIGIR*. 513–522.
[40] Xiao Yu, Hao Ma, Bo-June Paul Hsu, and Jiawei Han. 2014. On building entity recommender systems using user click log and freebase knowledge. In *WSDM*. 263–272.
[41] Yi Zhang, Jamie Callan, and Thomas Minka. 2002. Novelty and redundancy detection in adaptive filtering. In *SIGIR*. 81–88.
[42] Yuan Cao Zhang, Diarmuid Ó Séaghdha, Daniele Quercia, and Tamas Jambor. 2012. Auralist: introducing serendipity into music recommendation. In *WSDM*. 13–22.