Knowledge-Based Systems 000 (2018) 1–11

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Learning a unified embedding space of web search from large-scale query log

Lidong Bing[a,*], Zheng-Yu Niu[b], Piji Li[c], Wai Lam[c], Haifeng Wang[b]

[a] Tencent AI Lab, Shenzhen, China
[b] Baidu Inc, Beijing, China
[c] Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Hong Kong

ARTICLE INFO

Article history:
Received 24 September 2017
Revised 18 January 2018
Accepted 24 February 2018
Available online xxx

Keywords:
Web search
Query representation
Embedding space
Session analysis

ABSTRACT

In the procedure of Web search, a user first comes up with an information need and a query is issued with the need as guidance. After that, some URLs are clicked and other queries may be issued if those URLs do not meet his need well. We advocate that Web search is governed by a unified hidden space, and each involved element such as query and URL has its inborn position, i.e., projected as a vector, in this space. Each of above actions in the search procedure, i.e. issuing queries or clicking URLs, is an interaction result of those elements in the space. In this paper, we aim at uncovering such a unified hidden space of Web search that uniformly captures the hidden semantics of search queries, URLs and other involved elements in Web search. We learn the semantic space with search session data, because a search session can be regarded as an instantiation of users' information need on a particular semantic topic and it keeps the interaction information of queries and URLs. We use a set of session graphs to represent search sessions, and the space learning task is cast as a vector learning problem for the graph vertices by maximizing the log-likelihood of a training session data set. Specifically, we developed the well-known Word2vec to perform the learning procedure. Experiments on the query log data of a commercial search engine are conducted to examine the efficacy of learnt vectors, and the results show that our framework is helpful for different finer tasks in Web search.

1. Introduction

Web search is the fundamental tool for us to quickly retrieve the needed Web pages. To improve the retrieval performance, researchers have conducted extensive studies in different topics, such as query reformulation [8,9,50], query suggestion [2,34], query difficulty (or quality) prediction [21,22,29], query intent analysis [3,12,41], page quality evaluation and spam detection [7,11,20], and search result ranking [15,28,38]. In this paper, we exploit another paradigm which aims at mining distributed representation of Web search elements, such as terms, queries, pages/URLs, and websites. We advocate that Web search is governed by a unified hidden space, and each element can be embedded as a vector in the space. Fig. 1 depicts an example to show the intuition of this idea. The user has an information need (e.g. "I wanna repair my iPhone screen.") in mind which can be semantically represented as a vector of particular dimensions, 4 in this example, and each dimension indicates the relevance of his need with a particular hidden semantic topic. Under the guidance of this information need, three queries are issued. Although the user intends to formulate queries conveying his need on the third dimension, say "repair screen", the first two queries have a large semantic relatedness with the first dimension, say "Apple". Consequently, they retrieve pages mainly from the website of Apple. After browsing a few pages (e.g. u_1, u_2, and u_3) and feeling unsatisfied (maybe because of the high price), the user issues the last query with a hidden semantic representation well matching with his need, and accordingly, the returned URLs satisfy him better. To generalize the example, websites and query terms could also be involved and represented as vectors in the same space. Obviously, uncovering such a space governing the search procedure can be very useful for different finer tasks of Web search.

Researchers had observed the potential of generating semantic vectors for queries and URLs/pages, and conducted some pioneer investigations [19,25,46]. Our work is different from them in a few aspects. First, these works only learn vectors for queries and URLs, while our work also learns vectors for websites and terms. Therefore, the learnt vectors in our work can be applied to different tasks, not limited to result ranking. Second, our model can be eas-

* Corresponding author.
E-mail addresses: lyndonbing@tencent.com, binglidong@gmail.com (L. Bing), niuzhengyu@baidu.com (Z.-Y. Niu), pjli@se.cuhk.edu.hk (P. Li), wlam@se.cuhk.edu.hk (W. Lam), wanghaifeng@baidu.com (H. Wang).

https://doi.org/10.1016/j.knosys.2018.02.037
0950-7051/

ily extended to incorporate other types of elements, such as users. Third, the learnt term vectors enable our model to tackle new data such as unseen queries, while these previous works do not have such capability.

We conduct the semantic space learning from search session data since a search session can be regarded as an instantiation of a particular information need. To achieve the goal of learning continuous vector representations for different elements involved in search sessions (e.g., queries, URLs, and websites), a few research questions are raised, namely (a) How to represent the information from session data for vector learning? (b) What kind of models should one use to learn vector representations on session data? and (c) How to augment the model to deal with unseen queries from new sessions?

We cast this vector learning task with session data as a learning problem on a set of graphs, where each graph corresponds to a session, and the elements in a session are represented as vertices and related vertices are connected by edges. The use of graph serves as a suitable choice for session representation since it can capture the semantic interactions among the elements. Given the user's information need represented as a semantic vector, the probability of obtaining a session is jointly determined by the semantic meaning of involved elements, i.e., vertices of the session graph. Then we perform vector learning for vertices by maximizing the log-likelihood of a training session data set. With the learnt representation scheme, we can perform hidden semantic analysis on new session data, given queries or clicked URLs of the new session as source information.

Contributions. The main contributions of this work are as follows.

- We propose a framework of learning a unified semantic space of Web search, and different elements, such as queries, URLs, and terms, are embedded as vectors in this space. The vectors of different types of elements are directly comparable for similarity calculation. And our model also has good applicability on unseen data.
- We use graph structure for session data representation and develop an approach for vertex vector learning on a set of graphs. Our model can capture fine-grained structured information in click-through data. It is generic and naturally lends itself to extensions incorporating other types of elements from session data.
- Our model is trained on a large query log data generated by Baidu[1] search engine. Extensive experiments are conducted to examine the efficacy of the constructed semantic space, and the results show that the learnt vectors are helpful for different tasks.

This article substantially extends our previous work published as conference paper [10]. First, we elaborate on more technical details of the proposed model. Second, more experiments are conducted and more case studies are given, such as the experiments on entity recommendation in Section 6.6 and case studies in Section 5.2. Third, the differences between our work and previous works are discussed more thoroughly across different sections, such as Sections 1, 4, and 7.

The remainder of the article is organized as follows. We first give the problem definition in Section 2. After the first model is described in Section 3, an enhanced model is described in Section 4. The experimental dataset and case studies are given in Section 5, then the experimental settings and results are discussed in Section 6. After related works are reviewed in Section 7, we conclude the article and provide some possible directions for the future work in Section 8.

2. Problem formulation

We aim at uncovering a semantic space that governs the Web search procedure via projecting each involved element (such as search query and URLs) in Web search log as a vector of certain dimensions. The task is precisely defined as follows. Given a set of search sessions $S = \{s_i\}_{i=1}^{n}$ as training data, we aim at finding a unified semantic space to model Web search scenario so that the probability of observing the sessions in S is maximized. Let θ denote an instantiation of model parameters of the space. The log-likelihood objective function is expressed as follows:

$$\ell(\theta; S) = \sum_{s_i \in S} \log P(s_i; \theta), \tag{1}$$

where $P(s_i; \theta)$ denotes the probability that s_i is observed in the space with the parameters θ. Our goal is to find the best parameters by maximizing the above objective.

In our model, the parameters refer to the hidden vectors of involved elements in sessions. Let e_j denote an element such as a query or a URL in s_i, and $\mathbf{v}(e_j)$ denote the vector representation of e_j. Let $\mathbf{v}(s_i)$ denote the information need, represented as a vector, of the user corresponding to the session s_i. $\mathbf{v}(s_i)$ is also called session vector. $\mathbf{v}(s_i)$ and $\mathbf{v}(e_j)$ have the same dimensionality. Let $C(e_j)$ denote the context elements of e_j in s_i. We assume that the probability of e_j only depends on its context $C(e_j)$ and the user's information need, and it is denoted as $P(e_j; C(e_j), \mathbf{v}(s_i))$. Therefore, $P(s_i; \theta)$ can be calculated as:

$$P(s_i; \theta) = \prod_{e_j \in s_i} P(e_j; C(e_j), \mathbf{v}(s_i)). \tag{2}$$

$P(e_j; C(e_j), \mathbf{v}(s_i))$ is calculated with the vectors of elements in $C(e_j)$ and the vector $\mathbf{v}(s_i)$. The calculation will be described later. To summarize, our task is to learn vector representations of the elements in search sessions so that the objective function in Eq. (1) is maximized. To perform learning, we need to transform each session into training instances with the form $(e_j, C(e_j), s_i)$ for calculating $P(e_j; C(e_j), \mathbf{v}(s_i))$. To do so, a major task is to define the context $C(e_j)$ of the element e_j in the session s_i. For better capturing the structured information in click-through data, we introduce a graph representation of session data, which will be discussed in Section 3.

There are some existing studies conducting vector representation learning for words in Natural Language Processing and Speech Recognition [6,35,37,44]. However, they cannot be directly applied to our task due to the following reasons. Our training data is a set of sessions and each of them is represented as a session graph, while the training data of existing methods is a set of word sequences. In addition, a vector capturing the user's information need is incorporated into our learning procedure. Moreover, we intend to learn a unified space that embeds the elements of different granularities such as queries, URLs and terms.

3. Basic model for vector learning on session graphs

3.1. Session graph and training instances

In a search session, there are several types of involved elements. A user first issues a query, and some URLs are clicked in the result list. To obtain better results, she may issue more queries. When browsing a clicked page, the user may also browse other pages in the same website. And the corresponding website is also involved as an element of the session. Thus, a session involves three types

[1] http://www.baidu.com/.

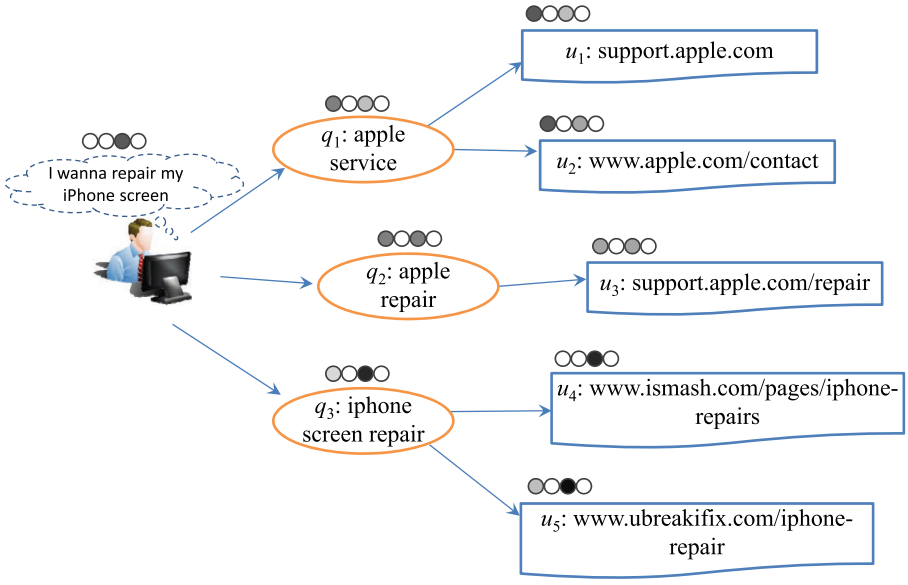

Fig. 1. An example of search session. The user has an information need in mind, which can be represented as a vector in a semantic space. With this information need in mind, the user issued three queries and clicked some URLs. Queries and URLs are also represented with vectors in the same space. The darker a dimension is, the larger the corresponding value is.

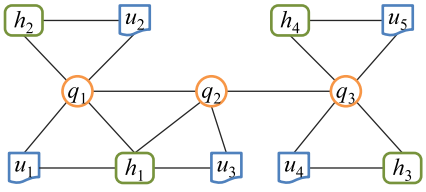

Fig. 2. An example of session graph, generated from Fig. 1.

of elements, namely, search query, URL, and website. To better capture the structural information in click-through data, we introduce a graph structure to represent sessions.

Definition 1 (Session Graph)**.** For a search session, its graph $\mathcal{G} = \{V, E\}$ is defined as an undirected graph. The vertex set V includes three types of vertices, namely, search query, clicked URL, and website. The edges are built as follows: (1) two successive queries are connected by an edge; (2) a clicked URL is connected with the corresponding query that retrieves the URL; (3) a website is connected with the URLs that come from it and the queries that result in the clicks. The intuition behind this procedure is that if there is possible semantic relationship between two elements, we connect them with an edge.

An example of session graph is given in Fig. 2. With a query q_1, a user clicked two URLs, namely, u_1 and u_2. Thus, we have two edges (q_1, u_1) and (q_1, u_2). The corresponding websites of u_1 and u_2 are h_1 and h_2, and they are also involved in the graph. Accordingly, we have the edges (u_1, h_1), (q_1, h_1), (u_2, h_2), and (q_1, h_2). After browsing u_1 and u_2, the user issued two more queries q_2 and q_3 and clicked more URLs. $C(e_j)$ is defined as the set of elements that are adjacent to e_j in the graph. For example, in Fig. 2, we have $C(q_1) = \{q_2, u_1, u_2, h_1, h_2\}$. To summarize, each generated training instances has the form of $(e_j, C(e_j), s_i)$ which means that the target element e_j comes from the session s_i with the context $C(e_j)$ from the same session.

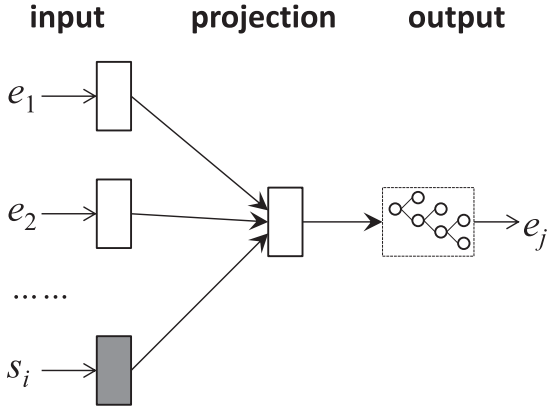

Fig. 3. Basic learning model: session2vec. The training instances in the input layer have the form of $(e_j, C(e_j), s_i)$ and $C(e_j) = \{e_1, e_2, \cdots\}$.

3.2. Basic learning model

The objective function of our basic model can be written as follows:

$$\ell(\theta; S) = \sum_{s_i \in S} \log P(s_i; \theta) = \sum_{s_i \in S} \sum_{e_j \in s_i} \log P(e_j; C(e_j), \mathbf{v}(s_i)). \quad (3)$$

We first develop a network for calculating the probability $P(e_j; C(e_j), \mathbf{v}(s_i))$ with the training instance $(e_j, C(e_j), s_i)$ by extending Word2vec [37]. As depicted in Fig. 3, this network has three layers, namely, the input layer, the projection layer, and the output layer. The input layer takes the vectors of elements in $C(e_j)$ and the session vector $\mathbf{v}(s_i)$ as the input. In the projection layer, the average of the element vectors is first calculated. Then the average is summed with $\mathbf{v}(s_i)$. The sum vector is denoted as $\mathbf{x}_j$. Note that the number of contextual elements may vary. To handle this, we calculate the average of contextual vectors. The output layer contains a structure of Huffman tree with each distinct element in the training sessions as a leaf. The more frequent an element is, the shorter its Huffman code is.

To compute the probability $P(e_j; C(e_j), \mathbf{v}(s_i))$ with the above network, we first introduce some notations. Let p denote the path from the root to the leaf e_j in the Huffman tree and L denote the

length of p. Let $v^p_{1:L}$ denote the vertices on the path p and thus we have $v^p_L = e_j$. Let $c_{1:L-1}$ be the sequence of binary codewords on the path p. Let $\boldsymbol{\gamma}_{1:L-1}$ denote the parameter vectors associated with the inner vertices $v^p_{1:L-1}$ on p, each of which is a vector of the same dimensionality as $\mathbf{x}_j$. $P(e_j; C(e_j), \mathbf{v}(s_i))$ is calculated as the probability of reaching the leaf e_j along the path p. To reach e_j along p, we need to go through $L-1$ binary selections. Specifically, at the inner vertex v^p_l, we select the branch having the codeword c_l with the probability $P(c_l; \boldsymbol{\gamma}_l, \mathbf{x}_j)$, which is defined with the sigmoid function σ as follows:

$$P(c_l; \boldsymbol{\gamma}_l, \mathbf{x}_j) = \{\sigma(\mathbf{x}_j \boldsymbol{\gamma}_l)\}^{1-c_l} \cdot \{1 - \sigma(\mathbf{x}_j \boldsymbol{\gamma}_l)\}^{c_l}. \quad (4)$$

Therefore, $P(e_j; C(e_j), \mathbf{v}(s_i))$ is equal to the product of these $L-1$ probabilities:

$$P(e_j; C(e_j), \mathbf{v}(s_i)) = \prod_{l=1}^{L-1} P(c_l; \boldsymbol{\gamma}_l, \mathbf{x}_j). \quad (5)$$

Thus, combining Eqs. (5) and (3), the objective function can be calculated with the network shown in Fig. 3.

3.2.1. Complexity and training

As discussed in Word2vec [36,37], the hierarchical softmax version's complexity is $Q = N \times D + D \times log_2(V)$ per training example, where N is the context size (equivalent to the average context size of training instances in our model), D is the vector dimensionality, and V is the vocabulary size (i.e. the number of distinct elements in our training query log). The total time complexity is $Q \times E \times T$, where E is the number of training epoches, and T is the number of training examples. In our basic model, we include the computation of an additional session vector. Therefore, our time complexity per training example is $(N+1) \times D + D \times log_2(V)$. Compared with Word2vec, our basic model needs $S \times D$ additional space for storing the session vectors, where S is the number of training sessions.

We use the stochastic gradient ascent algorithm to learn three types of parameters, namely, the vectors of elements, the vectors of the inner nodes in the Huffman tree, and the vectors of sessions. During the learning procedure, each instance generated in the previous section is fed into the network and the parameters related to the instance are updated. To maximize the objective in Eq. (3), the updating strategy is to make the individual probability $P(e_j; C(e_j), \mathbf{v}(s_i))$, calculated with the instance $(e_j, C(e_j), s_i)$, as larger as possible. The entire learning procedure is performed by scanning the whole set of training instances one or a few epoches depending on the efficiency requirement. For reference convenience, this model is called **session2vec**.

4. Enhanced model for vector learning

With session2vec, each element (i.e., query, URL, and website) appearing in the training data can be represented as a vector. However, session2vec cannot deal with unseen queries when analyzing new session data. To solve this issue, we extend session2vec and propose a unified learning model which is depicted in Fig. 4 and referred to as **session2vec+**. The upper part of session2vec+ is the same as session2vec depicted in Fig. 3. The lower part has the same design as the upper part and it incorporates the term-based training instances in the form of $(t_k, C(t_k), s_i)$, which are also generated from the corresponding session where the element-based instances come from. The session vector is shared by the two parts and plays the role of bridge to align the dimensions of element vectors learnt by the upper part and dimensions of term vectors learnt by the lower parts. Consequently, the terms are embedded in the same semantic space as the elements in the sessions. Thus, the term vectors can be utilized to build vectors for new elements such as unseen queries. Another advantage of session2vec+ is that these term vectors can help solve the sparsity issue in session2vec, since the vectors of sparse elements learnt in session2vec are likely unreliable.

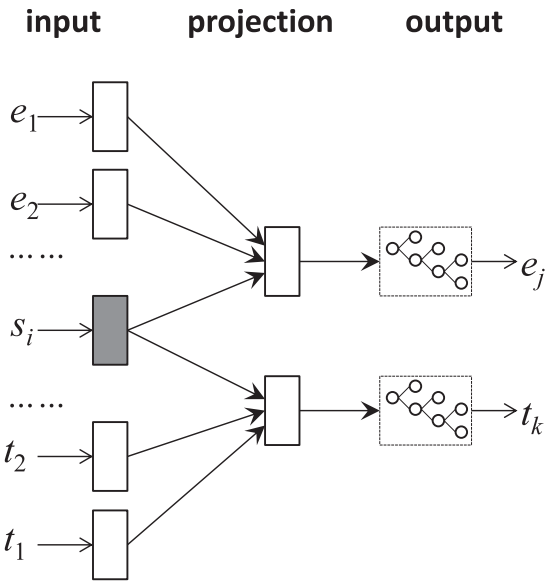

Fig. 4. Enhanced learning model session2vec+. The training instances of the upper part and the lower part have the forms of $(e_j, C(e_j), s_i)$ and $(t_k, C(t_k), s_i)$.

4.1. Training instances for term vector learning

We build term-based training instances by post-processing those element-based instances in Section 3.1. Specifically, if e_j of the instance $(e_j, C(e_j), s_i)$ is the query or the URL, it is transformed into a set of term-based instances. Let t_k denote a term in a query or a URL title. Each term t_k corresponds to one term-based instance in the form of $(t_k, C(t_k), s_i)$ where $C(t_k)$ is the context of t_k and contains all the terms of queries and URL titles in $C(e_j)$. If an element in $C(e_j)$ is a website, it does not contribute to the term context so as to avoid the general terms in website titles. In this stage, noun phrase chunking is conducted and a single term here may refer to a phrase such as "Stanford University" and "New York Times". Note that, in addition to existing queries, the terms may also come from the URL titles. Thus, our model is also augmented to handle the unseen query terms with those title terms.

4.2. Enhanced learning model

For session2vec+ depicted in Fig. 4, we define a new objective function as follows:

$$\ell'(\theta; S) = \sum_{s_i \in S} \log P(s_i; \theta) + \sum_{s_i \in S} \log P'(s_i; \theta), \quad (6)$$

where $P'(s_i; \theta)$ is the probability of s_i calculated with the term-based training instances. Similar to $P(s_i; \theta)$, $P'(s_i; \theta)$ is calculated as:

$$P'(s_i; \theta) = \prod_{t_k \in s_i} P(t_k; C(t_k), \mathbf{v}(s_i)), \quad (7)$$

where $P(t_k; C(t_k), \mathbf{v}(s_i))$ is the probability of t_k in s_i and its calculation procedure is the same as for $P(e_j; C(e_j), \mathbf{v}(s_i))$. Therefore, $\ell'(\theta; S)$ can be written as:

$$\ell'(\theta; S) = \sum_{s_i \in S} \{ \sum_{e_j \in s_i} \sum_{l=1}^{L^e-1} \log P(c^e_l; \boldsymbol{\gamma}^e_l, \mathbf{x}^e_j) + \sum_{t_k \in s_i} \sum_{l=1}^{L^t-1} \log P(c^t_l; \boldsymbol{\gamma}^t_l, \mathbf{x}^t_k) \}, \quad (8)$$

where superscripts e and t indicate the calculations for element instances and term instances respectively.

One may notice that both $P(s_i;\theta)$ and $P'(s_i;\theta)$ are the probabilities of s_i and they may be unequal. In fact, refer to the Eqs. (2), (4), and (5), the probability of a session is defined as the operation result of the sigmoid function on the semantic vectors and the parameter vectors associated with the Huffman tree. Consequently, it is possible that different types of input vectors, term-based or element-based, output different probabilities. We do not restrict $P(s_i;\theta)$ and $P'(s_i;\theta)$ to be equal as hard constraints in the formulation since such constraints may make our model less flexible in learning the semantic vectors for different elements. On the other hand, the session vector of s_i, as an intermediary, softly coordinates the learnt vectors of elements and terms to align their dimensions.

We employ the stochastic gradient ascent algorithm to maximize the objective function $\ell'(\theta;S)$. In the learning procedure, a major task is to derive the gradient with respect to the parameters related to a training instance. Here we handle the two types of training instances from the same session separately in each iteration. We first proceed with an element-based instance $(e_j, C(e_j), s_i)$ and let $\ell(j,l) = \log P(c_l^e;\boldsymbol{\gamma}_l^e,\mathbf{x}_j^e)$. After combined with Eq. (4), $\ell(j,l)$ is written as:

$$\ell(j,l) = (1-c_l^e)\log\{\sigma(\mathbf{x}_j^e\boldsymbol{\gamma}_l^e)\} + c_l^e\log\{1-\sigma(\mathbf{x}_j^e\boldsymbol{\gamma}_l^e)\}. \tag{9}$$

With some derivations, the partial derivatives with respect to $\mathbf{x}_j^e$ and $\boldsymbol{\gamma}_l^e$ are obtained as follows:

$$\frac{\partial\ell(j,l)}{\partial\mathbf{x}_j^e} = \{1-c_l^e-\sigma(\mathbf{x}_j^e\boldsymbol{\gamma}_l^e)\}\boldsymbol{\gamma}_l^e, \tag{10}$$

$$\frac{\partial\ell(j,l)}{\partial\boldsymbol{\gamma}_l^e} = \{1-c_l^e-\sigma(\mathbf{x}_j^e\boldsymbol{\gamma}_l^e)\}\mathbf{x}_j^e. \tag{11}$$

Therefore, the update formula of $\boldsymbol{\gamma}_l^e$ in the iterations is:

$$\boldsymbol{\gamma}_l^e \leftarrow \boldsymbol{\gamma}_l^e + \eta\{1-c_l^e-\sigma(\mathbf{x}_j^e\boldsymbol{\gamma}_l^e)\}\mathbf{x}_j^e, \tag{12}$$

where η is the learning rate. Recall that $\mathbf{x}_j^e$ is an intermediate vector calculated in the projection layer and our aim is to learn a vector $\mathbf{v}(e')$ for each element $e'\in C(e_j)$. To do so, $\mathbf{v}(e')$ is updated with the partial derivative of $\mathbf{x}_j^e$ as follows:

$$\mathbf{v}(e') \leftarrow \mathbf{v}(e') + \eta\sum_{l=1}^{L^e-1}\{1-c_l^e-\sigma(\mathbf{x}_j^e\boldsymbol{\gamma}_l^e)\}\boldsymbol{\gamma}_l^e. \tag{13}$$

Similarly, for a term-based instance $(t_k, C(t_k), s_i)$, let $\ell(k,l) = \log P(c_l^t;\boldsymbol{\gamma}_l^t,\mathbf{x}_k^e)$. And we have the update formulae as follows:

$$\boldsymbol{\gamma}_l^t \leftarrow \boldsymbol{\gamma}_l^t + \eta\{1-c_l^t-\sigma(\mathbf{x}_k^t\boldsymbol{\gamma}_l^t)\}\mathbf{x}_k^t, \tag{14}$$

$$\mathbf{v}(t') \leftarrow \mathbf{v}(t') + \eta\sum_{l=1}^{L^t-1}\{1-c_l^t-\sigma(\mathbf{x}_k^t\boldsymbol{\gamma}_l^t)\}\boldsymbol{\gamma}_l^t, \tag{15}$$

where $t'\in C(t_k)$. When updating the session vector $\mathbf{v}(s_i)$, both types of instances are jointly considered and we have:

$$\mathbf{v}(s_i) \leftarrow \mathbf{v}(s_i) + \eta\sum_{l=1}^{L^e-1}\frac{\partial\ell(j,l)}{\partial\mathbf{x}_j^e} + \eta\sum_{l=1}^{L^t-1}\frac{\partial\ell(k,l)}{\partial\mathbf{x}_k^t}. \tag{16}$$

With the above derivations of the learning procedure for session2vec+, the details of the learning procedure for session2vec in Section 3 can be easily derived. Similar to session2vec, the time and space complexities of session2vec+ could be derived straightforwardly.

5. Training data and case study

5.1. Training data

We employ a Chinese query log data set from Baidu search engine to conduct the vector learning with our proposed models. Our method for detecting user session boundaries combines the advantages of both time-gap-based session detection and task-based session detection [9]. First, the interval of two consecutive queries should be less than 15 minutes. Second, the similarity between two consecutive queries should be no less than a threshold. To calculate this similarity, we employ the term vector trained in a baseline system, called Word2vec.B which will be described later, to represent each query term and the sum of these term vectors is used as the feature vector of a query. The cosine similarity threshold is 0.5. We collected 23,676,669 sessions as the training data. On average, each session contains 2.1 queries and 2.3 clicks. Our data includes 10,413,491 unique queries, 13,126,252 URLs, 1,006,352 websites, and 3,965,539 terms (coming from queries and URL titles), respectively.

5.2. Case study

5.2.1. Semantic dimension demonstration

We show the salient elements and terms in some dimensions generated by our session2vec+. Specifically, three dimensions with the manually summarized titles “Star”, “Movie resource” and “Software resource” are illustrated in Table 1. For each dimension, five terms, five queries, and five websites are given. The English translations of the terms and queries are given in the brackets. These terms and elements have the largest values in the corresponding dimensions, meanwhile their frequency is no less than 100.

In the dimension of “Star”, the names of five stars from mainland China and Hong Kong are output as the salient terms, such as “Jet Li” and “Andy Lau”. The queries in this dimension are mainly searching for the personal information of stars. With respect to the websites, the entertainment homepages of the five largest websites in China are listed. In the dimension of “Movie resource”, the titles of popular movies are output as the salient terms and the queries are mainly about the information such as showtime and scheme song of the movies. One interesting observation is that although “Star” dimension and “Movie resource” dimension are related to some extent, our model can generate different salient term sets and salient query sets focusing on different aspects. The main reason is that searching star information and searching movie information are two quite different information needs. Our element-based and term-based training instances are transformed from the individual sessions so that these two different needs are well identified in the learning with such training instances. In addition, the websites involved in these two needs are also quite different so that they can help differentiate the two information needs.

5.2.2. Vector demonstration

The vectors of four smartphone-related terms are given in Fig. 5. The first three terms are hot smartphone brand names, namely, “苹果 (Apple)”, “三星 (Samsung)”, and “小米 (Mi)”,[2] in China. The other term is “智能手机 (smartphone)”. The dimension values of these four vectors are correlated very well. The cosine similarity values among them are given in Table 2. “Mi” is more similar to “Apple”, but “Samsung” is less similar to “Apple”. It is because Samsung has quite diverse product lines, such as refrigerator and air conditioning. Therefore, the vector of “Samsung” is less similar to the vectors of “Mi” and “Apple”. Another reason is that

[2] http://www.mi.com/en.

Table 1
Salient elements and terms in three dimensions.

	Terms	Queries	Websites
Star	刘德华 (Andy Lau) 周星驰 (Stephen Chow) 黄渤 (Bo Huang) 李连杰 (Jet Li) 周润发 (Yun-Fat Chow)	刘德华女儿 (daughter of Andy Lau) 李连杰的师父 (master of Jet Li) 周润发老婆 (wife of Yun-Fat Chow) 梁朝伟个人资料 (profile of Tony Leung) 周杰伦女朋友 (girlfriend of Jay Chou)	ent.sina.com.cn ent.qq.com ent.ifeng.com yule.sohu.com ent.163.com
Movie resource	那些年 (You Are the Apple of My Eye) 手机电影 (movie for smartphone) 非诚勿扰 (If You Are the One) 影视在线 (online movie) 阿凡达 (Avatar)	阿凡达上映时间 (showtimes of Avatar) 无间道在线观看 (Infernal Affairs online watching) 非诚勿扰主题曲 (theme song of If You Are the One) 叶问上映时间 (showtimes of Ip Man) 唐山大地震票房 (box office of Aftershock)	www.mtime.com movie.douban.com www.verycd.com www.1905.com www.iqiyi.com
Software resource	迅雷看看 (Xunlei player) 多特软件站 (Duote software) 华军软件园 (Onlinedown software) 酷我音乐盒 (Kuwo music box) 豌豆荚 (SnapPea)	酷狗音乐下载 (download Kugou music box) 豌豆荚使用 (how to use SnapPea) 搜狐影音下载 (download Sohu player) 华军软件园下载中心 (download center of Onlinedown) 天空下载站 (Skycn software)	www.wandoujia.com www.onlinedown.net www.pconline.com.cn www.skycn.com www.zol.com.cn

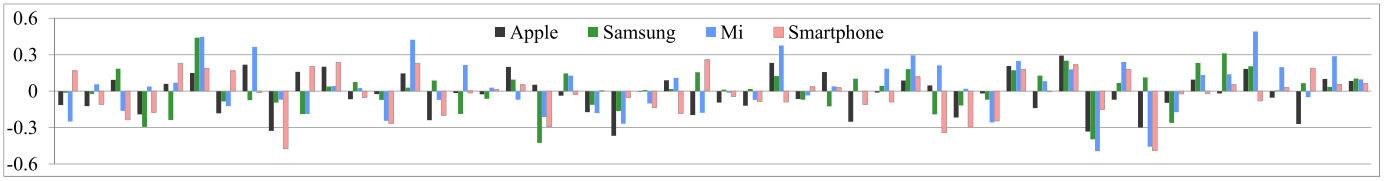

Fig. 5. The vectors (first 50 dimensions) of terms “Apple”, “Samsung”, “Mi”, and “Smartphone”.

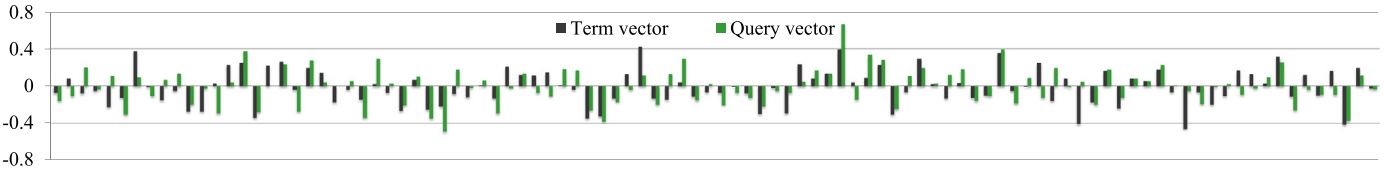

Fig. 6. The vector of the term “Peking University” and the vector of the query “Peking University”.

Table 2
Cosine similarity of vectors in Fig. 5.

	Apple	Samsung	Mi	smartphone
Apple	1	0.364	0.575	0.400
Samsung	0.364	1	0.425	0.385
Mi	0.575	0.425	1	0.408
smartphone	0.400	0.385	0.408	1

both Apple and Mi adopt the hunger marketing strategy which results in some similar semantic meanings, such as similar keywords in the queries related to available time and online booking.

The vector of the term “北京大学 (Peking University)” and the vector of the query “Peking University” learnt by session2vec+ are given in Fig. 6. The term vector and the query vector are generally correlated well and their cosine similarity is 0.591. As a result, we can reasonably derive the vector of an unseen query with the term vectors. These two vectors also show some differences. The reason is that the term “Peking University” appears in queries or URL titles with diverse meanings, such as “EMBA program in Peking University” and “Peking University Health Science Center”. For the query “Peking University”, the dominant information need is to find the university’s homepage or encyclopedia page.

6. Experimental results

In this section, we quantitatively examine the efficacy of our model in four different tasks, including query similarity prediction, website similarity prediction, URL ranking and entity recommendation.

6.1. Variants of our framework

Recall that we proposed two models, i.e., session2vec (s2v for short) and session2vec+ (s2v+ for short), as described in Sections 3 and 4 respectively. session2vec+ can generate semantic vectors for session elements and the terms from queries as well as URL titles at the same time. Therefore, we have three different variants for session2vec+ according to the manner of utilizing the learnt vectors. Variant **session2vec+.A** (s2v+.A for short) directly uses the learnt vectors of the elements. Variant **session2vec+.B** (s2v+.B for short) interpolates the element vector and the vectors of the terms from it. Precisely, for query q, the sum vector of the vectors of the terms in q is first calculated. Then, the sum vector is summed with the learnt vector of q, and the result vector is employed as the vector of q. Variant **session2vec+.C** (s2v+.C for short) uses the sum vector of the term vectors of an element as its vector representation. Note that s2v+.C is applicable for both existing elements in the training data and the unobserved elements.

6.2. Comparison systems

We compare our model with a few baselines that can also learn vector representations.

Word2vec.A (W2v.A for short). We build this baseline system by employing the Word2vec package[3] to conduct vector learning on an instance set generated from our training data. This instance set contains two types of sequences. The first type of sequence is

[3] https://code.google.com/p/word2vec/.

generated from the queries and the URL titles of a session, and the second type of sequence is generated from the queries and the visited websites. The elements in each sequence follow their occurrence order in the session. For example, for the session in Fig. 2, the generated sequences are (q_1, u_1, u_2, q_2, u_3, q_3, u_4, u_5) and (q_1, h_1, h_2, q_2, q_3, h_3, h_4). Then, the former sequence is transformed into a sequence of terms and the latter sequence is transformed into a sequence of terms and websites. After learning, W2v.A generates a vector for each term and each website.

Word2vec.B (W2v.B for short). We build this system by running the Word2vec package on a corpus containing 1 billion Chinese Web pages. Thus, a vector is generated for each term in the corpus. Note that this corpus is much larger than the data used in our model as described in Section 5.1, which brings in some advantage for this baseline.

PLSA. We run PLSA [23] on the pseudo-document corpus generated from our training data set. Specifically, we regard each term sequence generated for W2v.A as one pseudo-document. Then we use PLSA to learn topic vectors for the terms.

Note that researchers have proposed some approaches to handle a particular individual task in this section, and encouraging results have been reported [4,14,25,52]. Although these papers described their methods with some details, reliable reimplementation is still not a trivial task. Because different data sets are used, we cannot directly compare with their reported results. In fact, the vector representations generated by our model can serve as input features for most of them.

6.3. Results for query similarity prediction

In this section, we analyze our framework with a similar query ranking task to illustrate its behaviors under different variants, namely session2vec, s2v+.A, s2v+.B, and s2v+.C. The dimension number of vectors is 100 and NDCG [26] is employed as the metric to evaluate the performance.

6.3.1. Task setup

Task description. In this task, we employ the vectors generated by different methods to calculate the similarity between a source query and its candidate queries so as to rank the candidates. Cosine similarity is employed as the measure. For each source query, each of its candidate queries has a manually assigned relevance score. Our aim is to examine the effectiveness of different methods in vector representation generation. If a method is effective, its capability of ranking the candidate queries should be good so that a higher NDCG score is achieved.

Evaluation data. We employ a benchmark data set at Baidu that contains 500 source queries each of which is associated with 5 candidate queries. A Likert scale with three levels is employed to assess the candidate queries. Specifically, 3 means strongly relevant, such as “Bill Gates” and “founder of Microsoft”, 2 means relevant, such as “Bill Gates” and “Steve Jobs”, and 1 means irrelevant, such as “Bill Gates” and “Spider-Man”. Each candidate query is assessed by 3 assessors and their average score is rounded to the nearest relevance level. On average, each source query has 1.7 strongly relevant candidates, 1.2 relevant candidates, and 2.1 irrelevant candidates. Note that we employ a three-level Likert scale since it has less ambiguity in assessment. These 500 source queries are divided into two parts, namely, the observed part, and unobserved part. In the observed part, each source query and its candidate queries are observed in our training query log. This part has 129 source queries and it is referred to as **Q_obs**. The remaining 371 source queries compose of the unobserved part and we refer to this part as **Q_unobs**.

Table 3
Results of query similarity prediction on Q_obs..

	W2v.A	W2v.B	PLSA	s2v	s2v+.A	s2v+.B	s2v+.C
NDCG@1	0.765	0.769	0.727	0.784	0.792	0.797	0.799
NDCG@3	0.803	0.804	0.786	0.824	0.830	0.834	0.835
NDCG@5	0.825	0.845	0.810	0.838	0.841	0.849	0.853

6.3.2. Analysis of session2vec results on q_obs

Recall that our session2vec can generate a vector representation for each training query. For the baseline methods, the feature vector of a query is obtained by summing its term vectors. The results of different methods are given in the left part of Table 3. session2vec can outperform the baselines. Specifically, on NDCG@1, session2vec outperforms PLSA, W2v.A, and W2v.B by about 8% (significant with $P < 0.01$ in paired t-test[4]), 3% ($P < 0.05$), and 2% ($P < 0.05$), respectively. The following reasons can explain the performance difference. First, our training instances are generated from session graphs. In each graph, the elements have similar semantic meanings so that the contextual elements and the target element (i.e., e_j) in each training instance are semantically more cohesive. Such training instances bring in less noise. Second, the baselines generate query vectors by summing the term vectors, and the semantic meaning of a query cannot be fully derived by summing the semantic meanings of its terms. In contrast, session2vec regards each query as a basic item to generate vector representation. Third, our model maintains a session vector. Normally, the semantic of a session is less ambiguous than that of a query. The session vector is useful to guide vector generation for queries via deriving more precise information need.

The Word2vec baselines also perform well. Especially, W2v.B can outperform session2vec on NDCG@5. It is because W2v.B employs a larger training corpus compared with session2vec, which makes it have some advantages in ranking the queries with relatively lower similarity. session2vec can perform better on ranking more similar queries so that larger values on NDCG@1 and NDCG@3 are achieved. In addition, a large training corpus enables W2v.B to deal with sparse queries more effectively. W2v.A simply transforms the training sessions into sequences. Thus, it losses some click-through information of search sessions.

6.3.3. Analysis of session2vec+ results

As discussed in Section 4, session2vec is not able to tackle sparse queries effectively. In addition, if a query was not observed in the training stage, session2vec cannot generate a vector representation for it. To solve these problems, we design session2vec+ which incorporates the vector representation generation for terms in a unified model. The generated term vectors can be used in different variants as described in Section 6.1.

Results on Q_obs. To examine the effectiveness of session2vec+, we first compare its variants with session2vec on Q_obs and the results are given in the right part of Table 3. s2v+.A can outperform session2vec by 1% on NDCG@1 (significant with $P < 0.05$ in paired t-test). This shows that the unified learning in session2vec+ can generate better vector representation for queries. It is because the lower part of the network in Fig. 4 for term vector learning can help to overcome the sparsity problem of session2vec to some extent. Specifically, with the term-based learning part, the derived session vectors are more accurate since the sparsity problem of terms is less severe. Consequently, accurate session vectors can help generate better query vector representations. As we expected, s2v+.B can slightly outperform s2v+.A. It is because s2v+.A

[4] T-test is employed because the distribution of the results generally follow a normal distribution, similarly hereinafter for the subsequent results.

Please cite this article as: L. Bing et al., Learning a unified embedding space of web search from large-scale query log, Knowledge-Based Systems (2018), https://doi.org/10.1016/j.knosys.2018.02.037

Table 4
Effect of query frequency.

	A' [0-20%)	B' [20%-40%)	C' [40%-60%)	D' [60%-80%)	E' [80%-1]
A [0-20%)	**0.799** 0.798 *0.791*	0.799 **0.801** *0.796*	*0.793* 0.794 **0.799**	*0.792* 0.801 **0.803**	*0.790* 0.794 **0.796**
B [20%-40%)	**0.797** 0.796 *0.794*	0.796 **0.797** *0.793*	**0.790** 0.789 *0.786*	*0.788* 0.796 **0.798**	*0.786* 0.790 **0.797**
C [40%-60%)	*0.791* *0.791* **0.792**	*0.793* 0.795 **0.796**	*0.790* 0.796 **0.799**	*0.787* 0.797 **0.802**	*0.784* 0.796 **0.801**
D [60%-80%)	*0.785* 0.791 **0.799**	*0.783* 0.796 **0.797**	0.783 *0.782* **0.796**	*0.781* 0.785 **0.805**	*0.780* 0.789 **0.800**
E [80%-1]	*0.786* 0.792 **0.807**	*0.779* 0.787 **0.803**	*0.778* 0.790 **0.792**	*0.776* 0.782 **0.790**	*0.771* 0.797 **0.798**

Table 5
Results of query similarity on Q_unobs..

	s2v+.C	W2v.A	W2v.B	PLSA
NDCG@1	0.798	0.762	0.766	0.736
NDCG@3	0.836	0.805	0.812	0.787
NDCG@5	0.852	0.836	0.837	0.815

still suffers from the sparsity problem of the training data. s2v+.B has the term vectors interpolated with the query vector so that it can solve the sparsity issue better than s2v+.A.

We see that s2v+.C is the most effective one. Compared with the performance of the baselines given in Table 3, s2v+.C can outperform all results of them (significant with $P < 0.05$ in paired t-test). The comparison shows that the sum vector of term vectors generated by our session2vec+ can better derive the query vector. It is because the unified learning by session2vec+ can align the semantic meanings of queries and the terms better with the session vector playing the role of bridge. In contrast, the generated term vectors in the baselines are not as effective as ours for deriving the query vectors.

Compared with s2v+.A and s2v+.B, s2v+.C is also more effective. The reason for this outcome is that the sparsity of the training data still affects the reliability of the generated query vectors by session2vec+, especially for the low frequency queries. Such effect degrades the performance of s2v+.A and s2v+.B since both of them use the query vectors. To have a closer look of the sparsity problem, we divide Q_obs into 5 equal buckets according to the frequency of the queries. These buckets are referred to as Bucket A, B, C, D, and E in descending order of frequency. Similarly, the candidates queries are also divided into 5 equal buckets, and they are referred to as Bucket A', B', C', D', and E'. We evaluate the performance of these three variants in different frequency intervals and the results are shown in Table 4. In each cell, the results of s2v+.A, s2v+.B, and s2v+.C are given in the upper, middle, and lower positions, respectively. The largest value is underscored, in bold and of green color, the smallest value is in italic and of red color, and the median value is in normal font. As shown in the upper left of Table 4. s2v+.A and s2v+.B can perform better for more frequent queries, When the queries become less frequent, the performance of s2v+.A and s2v+.B declines. Meanwhile, s2v+.C is not affected much and outperforms the other two.

Results on Q_unobs. We also examine the performance of s2v+.C on the unobserved portion Q_unobs. Its performance is compared with the baselines and the results are given in Table 5. s2v+.C can outperform the baselines in all cases. Specifically, it achieves 8%, 5% and 4% improvements (significant with $P < 0.05$ in paired t-test) on NDCG@1 values compared with PLSA, W2v.A, and W2v.B, respectively. This comparison demonstrates that the term vector representation generated with our model is more effective due to the unified learning strategy and modeling the session vector. Combining the results in Tables 3 and 5, we observe that s2v+.C is the most effective system.

6.4. Results for website similarity prediction

6.4.1. Task setup

In this task, we employ the vectors generated by different systems in the calculation of website similarity. For the baselines PLSA and W2v.B, the vector representation of a website is obtained by summing the vectors of the terms in the title of its homepage. Recall that one type of the training sequence for W2v.A has website elements. Therefore, W2v.A adopts two methods to obtain the vector of a website. The first method is the same as the above two baselines and it is referred to as W2v.A.T. The second method directly uses the website vectors generated by learning and it is referred to as W2v.A.S. With respect to our model, we have three variants, namely, s2v.S, s2v+.S, and s2v+.T. s2v.S and s2v+.S also directly use the learnt website vectors in session2vec and session2vec+. s2v+.T adopts the same method as W2v.A.T to obtain the vector of a website by summing term vectors.

6.4.2. Evaluation data

This data set contains 500 source websites with different popularity degrees. Each source website is associated with 5 candidate websites. A Likert scale with three levels is employed to assess the candidate websites. Specifically, 3 means strongly relevant, such as "sports.sina.com.cn" and "sports.qq.com", 2 means relevant, such as "sports.sina.com.cn" and "www.sina.com.cn ", and 1 means irrelevant, such as "sports.sina.com.cn" and "mil.qq.com". On average, each source website has 1.6 strongly relevant candidates, 1.4 relevant candidates, and 2.0 irrelevant candidates. We find that all the source and candidate websites are covered by our training data set.

6.4.3. Results

The results are given in Table 6. The variants of our model outperform the baselines. Specifically, s2v+.S achieves 2% to 10% improvements (significant with $P < 0.05$ in paired t-test) comprised with the baselines on NDCG@1. Among the variants of our model, both s2v+.S and s2v.S perform better than s2v+.T. It shows that the directly learnt vectors of websites are more effective for similarity prediction than the sum of term vectors from the title. This observation is different from the observation for queries. One reason is that the sparsity problem for websites is not severe since the number of websites is much smaller than that of URLs and queries. Another reason is that the title of a homepage, such as "The best car website in China", sometimes contains unrelated terms such as "the best" and "China". These reasons can also explain why W2v.A.S performs better than W2v.B, even though the latter uses a very large training corpus.

6.5. Results for URL ranking

6.5.1. Task setup

In this experiment, we evaluate the task of re-ranking a set of result URLs for a query according to their relevance. For s2v+.C and the baselines, the relevance score between a query and its candidate URLs is computed as the cosine similarity of their vector representations. Still for each query, the vector representation is obtained by summing the vectors of its query terms. For each URL, the vector representation is obtained by summing the vectors of the terms in its title. In addition to the baselines given in Section 6.2, we introduce another baseline, namely BM25 [40],

Please cite this article as: L. Bing et al., Learning a unified embedding space of web search from large-scale query log, Knowledge-Based Systems (2018), https://doi.org/10.1016/j.knosys.2018.02.037

Table 6
Results for website similarity prediction.

	s2v+.S	s2v+.T	s2v.S	W2v.A.S	W2v.A.T	W2v.B	PLSA
NDCG@1	0.794	0.786	0.791	0.778	0.770	0.772	0.719
NDCG@3	0.855	0.843	0.849	0.833	0.831	0.832	0.763
NDCG@5	0.883	0.880	0.881	0.869	0.868	0.870	0.794

Table 7
Results for ranking result URLs.

	s2v+.C	W2v.A	W2v.B	PLSA	BM25
NDCG@1	0.611	0.580	0.587	0.576	0.559
NDCG@3	0.632	0.613	0.615	0.607	0.582
NDCG@5	0.640	0.627	0.631	0.630	0.616

Table 8
Results for entity recommendation.

	s2v+.C	W2v.A	W2v.B	PLSA
NDCG@1	0.325	0.305	0.312	0.260
NDCG@3	0.378	0.354	0.360	0.307
NDCG@5	0.396	0.376	0.386	0.339

which is a ranking function commonly used to rank matching documents according to their relevance to a search query. Specifically, our BM25 baseline is a revision of the original BM25 formula to consider some necessary modifications, such as the normalization of term frequency according to Singhal et al. [48] and the modification for inverse document frequency according to Fang et al. [16]. As discussed above, s2v+.C is the most effective variant of our framework and it also has better adaptability for unseen data. In addition, the URL vectors also face the sparsity issues. Therefore, we conduct the comparison between s2v+.C and the baselines in this task.

6.5.2. Evaluation data

This data set contains 1000 queries of various length and popularity. On average, each query has 19.8 marked URLs that are retrieved with the query. A Likert scale with five levels is employed to assess the relevance of each URL. Specifically, 5 means strongly relevant, 4 means relevant, 3 means somewhat relevant, 2 means somewhat irrelevant, and 1 means irrelevant. In this dataset, each candidate URL for a test query contains all query terms in the page content.

6.5.3. Results

The results are given in Table 7. All vector representation based methods can outperform BM25. It shows that relevance assessment based on the intent or topic analysis is more reliable than that with simple term matching. Our model achieves the best results in all cases compared with the baselines. Specifically, it outperforms the baselines with improvements (significant with $P < .01$ in paired t-test) from 4% to 9% on NDCG@1. This demonstrates that, among the vector representation based methods, our model can better capture the semantic similarity relation between queries and URLs. The main reason is that when learning the vectors, our unified model jointly considers different element types in the sessions such as websites and URLs as well as terms so that the learnt term vectors automatically encode the information of semantic similarity among the elements.

6.6. Results for entity recommendation

6.6.1. Task setup

Major search engines recommend some entities that are relevant to the query on the right of the result page. One such example is given in Fig. 7. In this example, the user issues a query "founder of Baidu". The founders of three other Internet companies in China are recommended. In this task, we employ the vector representations in ranking the candidate entities of a particular query. The comparison is conducted among s2v+.C and the baselines. The vector representation of each query is obtained in the same way as described in Section 6.5. For the entities with multiple terms, their vector representations are also generated in the same way.

6.6.2. Evaluation data

We collected top 100,000 hot queries in one period from Baidu search engine. The candidate entities are obtained as follows. For each query, its recommendation entities in the same period are employed as the candidates for ranking. To avoid noise, the candidates with impressions (the number of times a candidate was shown) less than 100 are filtered. The gold standard relevance score is automatically assigned based on the click-through rate (CTR) of the entities. CTR is defined as the number of clicks on an entity divided by its impression. We employ a Likert scale with five levels to assess the candidate entities. Specifically, 5 means strongly relevant and the responding entities have CTR $>$ 0.03, 4 means relevant and the responding entities have 0.02 $<$ CTR $\leq$ 0.03, 3 means somewhat relevant and the responding entities have 0.01 $<$ CTR $\leq$ 0.02, 2 means somewhat irrelevant and the responding entities have 0 $<$ CTR $\leq$ 0.01, and 1 means irrelevant and the responding entities have CTR $=$ 0.

6.6.3. Results

The results are given in Table 8. s2v+.C achieves the best results in all cases. Specifically, s2v+.C outperforms the baselines with 4% to 25% improvements (significant with $P < 0.01$ in paired t-test) on NDCG@1. The reasons for this outcome are similar to those for the above experiments. In addition, the training data of session graphs is more suitable to capture the entities' semantic meanings in Web search scenario. Note that the result values in this task are much lower than those of the previous tasks. The reason is that the candidate entities as well as their relevance scores are automatically collected with a click-through log of entity recommendation. Besides semantic similarity, other factors such as personal interest also affect the clicking behavior of users and these cannot be fully revealed by the similarity.

7. Related work

7.1. Query and URL vector learning

Researchers had observed the potential of generating semantic vectors for search queries and Web pages [19,25,46]. Deep Structured Semantic Model (DSSM) [25] and Convolutional Latent Semantic Model (CLSM) [46] employ deep neural network to map the raw term vector (i.e., with the bag-of-words representation) of a query or a document to its latent semantic vector. Both of them use the full text of pages as input, and CLSM also captures the text contextual information. The network architecture in our model is significantly different from them and it can be trained more efficiently. Furthermore, our framework also learns vectors of terms

Please cite this article as: L. Bing et al., Learning a unified embedding space of web search from large-scale query log, Knowledge-Based Systems (2018), https://doi.org/10.1016/j.knosys.2018.02.037

Fig. 7. An example of entity recommendation. When a user searches "founder of Baidu", a set of related entrepreneurs in Internet industry are given on the right of the result page, including Pony Ma (the founder of Tencent), Jack Ma (the founder of Alibaba), and Hongyi Zhou (the founder of Qihu).

and websites. Some other studies attempted to learn a binary vector for each query or URL. Each binary value shows whether the URL or query is relevant to the semantic dimension [39]. Obviously, binary value imposes limitations in real applications.

Gao et al. proposed Bi-Lingual Topic Model (BLTM) and linear Discriminative Projection Model (DPM) for query-document matching at the semantic level [19]. The models are trained on click-through data and the objectives are tailored for document ranking task. More specifically, BLTM is a generative model and it requires that a query and its clicked documents share the same distribution over topics and contain similar factions of words assigned to each topic. DPM is learnt using the S2Net algorithm that follows the pairwise learning-to-rank paradigm. Previous works also tried to learn query-document similarity from click-through data with implicit representation of semantics, such as click-through bipartite graph or translation models [14,18,52]. In contrast to these methods, we model the search procedure in a more general perspective by assuming that the search activities are governed by a hidden unified space of Web search semantics. This formulation allows us to incorporate more types of elements, and learn their representations uniformly.

7.2. Distributed representation learning

Another related area is the study of vector representation learning for words. A very popular model for estimating neural network language model was proposed in [6]. However, this model was computationally expensive. Word2vec [37] is a recent development with a simple architecture for efficient training. A development [30] of Word2vec by Le and Mikolov embeds paragraphs into the same space for words, which shares similar architecture with our session2vec. In this paper, we adapt the neural network based model on graph data. Moreover, we intend to learn an intent space that uniformly embeds the elements of different granularities, such as queries, websites, and terms. In comparison, our work focuses on modeling graph data, not bag of words or word sequences. In addition, the session vector is incorporated in the networks for modeling the information need of users. More importantly, the tailor-made enhanced learning model delicately embeds the terms in the same semantic space of the elements. Some other works employed neural networks to learn concept vector representations of input text objects for similarity calculation under a supervised setting [53], while our work learns vector representations for different elements in Web search under an unsupervised setting.

7.3. Query semantic identification

Query semantic identification aims at identifying predefined semantic classes or undefined clusters of queries. These works are related to our work from the perspective of query semantic discovery since our models represent each query as a vector with its dimension values showing the relevance of the query with hidden semantic topics. The investigated predefined query classes in previous works [5,17,45] include "Internet", "Autos", "NBA", "shoes", etc. Some works classified queries by search tasks such as "purchase computer", "plan a travel", "job-finding query" [27,31,47]. Classifying the queries into pre-defined classes is a challenging task since queries are short and ambiguous [33,43,49]. It is observed that users with similar information needs click the same group of URLs, even though the queries they issue vary [51]. Based on this observation, researchers hypothesize that queries within such a cluster express highly similar information needs. Click-through bipartite graph is commonly used in query clustering studies [4]. Some researchers studied the importance of query templates and structured patterns [1,13,32]. They concluded that a large fraction of queries follow some templates in most examined domains. Other studies identified different intents of an ambiguous query with its following refined queries or the clicked URLs [24,42]. Different from the above works, we do not explicitly assign a semantic label (e.g. "Internet" and "purchase computer") to queries or generate query clusters. Instead, a vector representation of each query is computed, which could serve as input features of other tasks.

8. Conclusions and future work

In this paper, we proposed a framework to uncover a unified semantic space for Web search. Two neural-network-based models are developed to learn continuous vectors for elements and terms. Compared with previous studies, our framework can perform hidden semantic learning for different types of elements uniformly, which enables the similarity calculations across element types. Moreover, our models perform vector representation learning on graph data by converting a graph vertex as a training example with its neighboring vertices as the context information. Experimental results show that the learnt vector representations are more effective than some standard methods (including Word2vec and PLSA) in four finer tasks of Web search.

For the future work, a few directions are worthwhile to explore. The first direction is to extend our framework to model the interest profile of use rs. One way is to employ the learn session vectors to derive the user interest profile. Another way is to directly involve the users into the learning procedure. The second direction is to enhance the session graph with the information of click order and dwell time. The third direction is to derive the real-time information need with the partial information of the current search session.

Please cite this article as: L. Bing et al., Learning a unified embedding space of web search from large-scale query log, Knowledge-Based Systems (2018), https://doi.org/10.1016/j.knosys.2018.02.037

References

[1] G. Agarwal, G. Kabra, K.C.-C. Chang, Towards rich query interpretation: walking back and forth for mining query templates, in: WWW, 2010, pp. 1–10.
[2] A. Anagnostopoulos, L. Becchetti, C. Castillo, A. Gionis, An optimization framework for query recommendation, in: WSDM, 2010, pp. 161–170.
[3] R. Baeza-Yates, L. Calderón-Benavides, C. González-Caro, The intention behind web queries, SPIRE, 2006.
[4] D. Beeferman, A. Berger, Agglomerative clustering of a search engine query log, in: KDD, 2000, pp. 407–416.
[5] S.M. Beitzel, E.C. Jensen, D.D. Lewis, A. Chowdhury, O. Frieder, Automatic classification of web queries using very large unlabeled query logs, ACM Trans. Inf. Syst. 25 (2) (2007).
[6] Y. Bengio, R. Ducharme, P. Vincent, C. Janvin, A neural probabilistic language model, J. Mach. Learn. Res. 3 (2003) 1137–1155.
[7] L. Bing, R. Guo, W. Lam, Z.-Y. Niu, H. Wang, Web page segmentation with structured prediction and its application in web page classification, in: SIGIR, 2014, pp. 767–776.
[8] L. Bing, W. Lam, T. Wong, Using query log and social tagging to refine queries based on latent topics, in: Proceedings of the 20th ACM Conference on Information and Knowledge Management, CIKM 2011, Glasgow, United Kingdom, October 24–28, 2011, 2011, pp. 583–592.
[9] L. Bing, W. Lam, T.-L. Wong, S. Jameel, Web query reformulation via joint modeling of latent topic dependency and term context, ACM Trans. Inf. Syst. 33 (2) (2015).
[10] L. Bing, Z. Niu, W. Lam, H. Wang, Learning a semantic space of web search via session data, in: Information Retrieval Technology - 12th Asia Information Retrieval Societies Conference, AIRS 2016, Beijing, China, November 30, - December 2, 2016, Proceedings, 2016, pp. 83–97.
[11] S. Brin, L. Page, The anatomy of a large-scale hypertextual web search engine, Comput. Netw. ISDN Syst. 30 (1–7) (1998) 107–117.
[12] A. Broder, A taxonomy of web search, SIGIR Forum 36 (2) (2002) 3–10.
[13] J.C.K. Cheung, X. Li, Sequence clustering and labeling for unsupervised query intent discovery, in: WSDM, 2012, pp. 383–392.
[14] N. Craswell, M. Szummer, Random walks on the click graph, in: SIGIR, 2007, pp. 239–246.
[15] A. Dong, Y. Chang, Z. Zheng, G. Mishne, J. Bai, R. Zhang, K. Buchner, C. Liao, F. Diaz, Towards recency ranking in web search, in: WSDM, 2010, pp. 11–20.
[16] H. Fang, T. Tao, C. Zhai, A formal study of information retrieval heuristics, in: SIGIR, 2004, pp. 49–56.
[17] A. Fuxman, P. Tsaparas, K. Achan, R. Agrawal, Using the wisdom of the crowds for keyword generation, in: WWW, 2008, pp. 61–70.
[18] J. Gao, X. He, J.-Y. Nie, Clickthrough-based translation models for web search: from word models to phrase models, CIKM, 2010.
[19] J. Gao, K. Toutanova, W.-t. Yih, Clickthrough-based latent semantic models for web search, in: SIGIR, 2011, pp. 675–684.
[20] Z. Gyöngyi, H. Garcia-Molina, J. Pedersen, Combating web spam with trustrank, in: VLDB, 2004, pp. 576–587.
[21] C. Hauff, V. Murdock, R. Baeza-Yates, Improved query difficulty prediction for the web, in: CIKM '08, 2008, pp. 439–448.
[22] B. He, I. Ounis, Query performance prediction, Inf. Syst. 31 (7) (2006) 585–594.
[23] T. Hofmann, Unsupervised learning by probabilistic latent semantic analysis, Mach. Learn. 42 (1–2) (2001) 177–196.
[24] Y. Hu, Y. Qian, H. Li, D. Jiang, J. Pei, Q. Zheng, Mining query subtopics from search log data, SIGIR, 2012.
[25] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, L. Heck, Learning deep structured semantic models for web search using clickthrough data, in: CIKM, 2013, pp. 2333–2338.
[26] K. Järvelin, J. Kekäläinen, Cumulated gain-based evaluation of ir techniques, ACM Trans. Inf. Syst. 20 (4) (2002) 422–446.
[27] M. Ji, J. Yan, S. Gu, J. Han, X. He, W.V. Zhang, Z. Chen, Learning search tasks in queries and web pages via graph regularization, in: SIGIR, 2011, pp. 55–64.
[28] T. Joachims, Optimizing search engines using clickthrough data, in: KDD, 2002, pp. 133–142.
[29] Y. Kim, A. Hassan, R.W. White, Y.-M. Wang, Playing by the rules: mining query associations to predict search performance, in: WSDM, 2013, pp. 133–142.
[30] Q.V. Le, T. Mikolov, Distributed representations of sentences and documents, in: ICML, 2014, pp. 1188–1196.
[31] X. Li, Y.-Y. Wang, A. Acero, Learning query intent from regularized click graphs, in: SIGIR, 2008, pp. 339–346.
[32] Y. Li, B.-J.P. Hsu, C. Zhai, Unsupervised identification of synonymous query intent templates for attribute intents, in: CIKM, 2013, pp. 2029–2038.
[33] C. Luo, Y. Liu, M. Zhang, S. Ma, Query ambiguity identification based on user behavior information, in: AIRS, 2014, pp. 36–47.
[34] Q. Mei, D. Zhou, K. Church, Query suggestion using hitting time, in: CIKM, 2008, pp. 469–478.
[35] T. Mikolov, Statistical Language Models Based on Neural Networks, 2012 Ph.D. thesis.
[36] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, CoRR abs/1301.3781 (2013).
[37] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: NIPS, 2013, pp. 3111–3119.
[38] F. Radlinski, T. Joachims, Query chains: learning to rank from implicit feedback, in: KDD, 2005, pp. 239–248.
[39] X. Ren, Y. Wang, X. Yu, J. Yan, Z. Chen, J. Han, Heterogeneous graph-based intent learning with queries, web pages and wikipedia concepts, in: WSDM, 2014, pp. 23–32.
[40] S.E. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, M. Gatford, Okapi at trec-3, in: TREC, 1994, pp. 109–126.
[41] D.E. Rose, D. Levinson, Understanding user goals in web search, in: WWW, 2004, pp. 13–19.
[42] E. Sadikov, J. Madhavan, L. Wang, A. Halevy, Clustering query refinements by user intent, WWW, 2010.
[43] M. Sanderson, Ambiguous queries: test collections need more sense, in: SIGIR, 2008, pp. 499–506.
[44] H. Schwenk, Continuous space language models, Comput. Speech Lang. 21 (3) (2007) 492–518.
[45] D. Shen, J.-T. Sun, Q. Yang, Z. Chen, Building bridges for web query classification, in: SIGIR, 2006, pp. 131–138.
[46] Y. Shen, X. He, J. Gao, L. Deng, G. Mesnil, A latent semantic model with convolutional-pooling structure for information retrieval, in: CIKM, 2014, pp. 101–110.
[47] Y. Shen, J. Yan, S. Yan, L. Ji, N. Liu, Z. Chen, Sparse hidden-dynamics conditional random fields for user intent understanding, in: WWW, 2011, pp. 7–16.
[48] A. Singhal, C. Buckley, M. Mitra, Pivoted document length normalization, in: SIGIR, 1996, pp. 21–29.
[49] R. Song, Z. Luo, J.-Y. Nie, Y. Yu, H.-W. Hon, Identification of ambiguous queries in web search, Inf. Process. Manage. 45 (2) (2009) 216–229.
[50] X. Wang, C. Zhai, Mining term association patterns from search logs for effective query reformulation, in: CIKM, 2008, pp. 479–488.
[51] J.-R. Wen, J.-Y. Nie, H.-J. Zhang, Clustering user queries of a search engine, in: WWW, 2001, pp. 162–168.
[52] W. Wu, H. Li, J. Xu, Learning query and document similarities from click-through bipartite graph with metadata, in: WSDM, 2013, pp. 687–696.
[53] W.-t. Yih, K. Toutanova, J.C. Platt, C. Meek, Learning discriminative projections for text similarity measures, in: CoNLL, 2011, pp. 247–256.

Please cite this article as: L. Bing et al., Learning a unified embedding space of web search from large-scale query log, Knowledge-Based Systems (2018), https://doi.org/10.1016/j.knosys.2018.02.037