Learning a Semantic Space of Web Search via Session Data*

Lidong Bing[1], Zheng-Yu Niu[2], Wai Lam[3], and Haifeng Wang[2]

[1] Tencent Inc. lyndonbing@tencent.com
[2] Baidu Inc. {niuzhengyu, wanghaifeng}@baidu.com
[3] Department of Systems Engineering and Engineering Management
The Chinese University of Hong Kong wlam@se.cuhk.edu.hk

Abstract. In Web search, a user first comes up with an information need and issues an initial query. Then some retrieved URLs are clicked and other queries are issued if he/she is not satisfied. We advocate that Web search is governed by a hidden semantic space, and each involved element such as query and URL has its projection, i.e., 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 semantic space of Web search that uniformly captures the hidden semantics of search queries, URLs and other elements. We propose session2vec and session2vec+ models to learn vectors in the space with search session data, where a search session is regarded as an instantiation of an information need and keeps the interaction information of queries and URLs. Vector learning is done on a large query log from a search engine, and the efficacy of learnt vectors is examined in a few tasks.

1 Introduction

In the study of word embedding, words are mapped into a vector space such that semantically relevant words are placed near each other [16, 17, 1]. Word vectors are helpful for a wide range of NLP tasks by better capturing syntactic and semantic information than simple lexical features [23, 8, 14]. In this work, we explore to apply embedding methodology to model the intrinsic hidden semantic space of Web search. Figure 1(a) gives an example to illustrate the intuition. The user has an information need in mind which can be represented as a 4-dimension vector, and each dimension indicates the relevance of his need with a particular semantic topic. Although the user intends to formulate queries conveying his need on the third dimension, the first two queries are not precise enough. Then the user issues the last query that well matches his need, and accordingly, the returned URLs satisfy him. To generalize, websites and query terms could also be involved and projected as vectors in the same space. Obviously, building such a space governing the search procedure could be useful for different tasks, such as query suggestion, result ranking, etc.

We conduct the semantic space learning using search session data since a search session can be regarded as an instantiation of a particular information need. The learning task is cast as a vertex embedding problem on a set of graphs (built from sessions), where the elements in a session are represented as vertices and related vertices

* This work is substantially supported by a grant from the Research Grant Council of the Hong Kong Special Administrative Region, China (Project Code: CUHK413510).

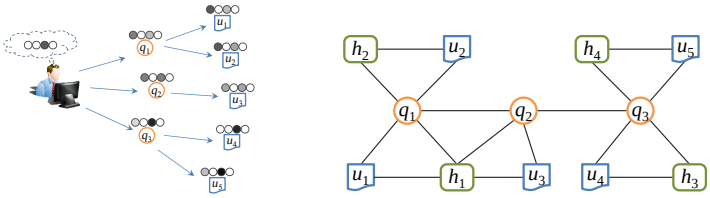


Fig. 1. Search session and session graph.

are connected by edges. The use of graph seems a suitable choice for session representation because it captures the semantic interactions among elements. Given user's information need, represented as a semantic vector, the probability of obtaining a session is jointly determined by semantic meaning of involved elements, i.e., vertices of the session graph. Then we perform vector learning for vertices via maximizing the log-likelihood of a training data of sessions.

Our main contributions are: (1) a framework is proposed to learn a semantic space of Web search, and different elements (such as queries, URLs, and terms) are projected as vectors in this space. Vectors of different elements are directly comparable for semantic similarity calculation, and our model has good applicability to unseen data; (2) We use graph structure to represent session data and develop an approach for vertex vector learning on session graphs. Our model can capture fine-grained structure information in click-through data. It is flexible to incorporate other types of elements. (3) Our model is trained on a large query log data from a 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.

2 Related Work

Researchers had observed the potential of generating semantic vectors for search queries and Web pages [7, 10, 21]. Deep Structured Semantic Model (DSSM) [10] and Convolutional Latent Semantic Model (CLSM) [21] employ deep neural network to map the raw term vector 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 contextual information. The network architecture in our model is different from them and it can be trained more efficiently. Furthermore, our framework also learns vectors of terms and websites and can be easily extended to include other elements, e.g. users. The learnt term vectors enable our model to tackle unseen data. Some other studies attempted to learn binary vectors for queries or URLs and binary values show the relevance to semantic dimensions [18].

[7] proposed Bi-Lingual Topic Model (BLTM) and linear Discriminative Projection Model (DPM) for query-document matching at the semantic level. 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 semantic representation, such as bipartite graph or translation models [4, 6, 24]. [8] proposed to learn query and term vectors for query rewriting in sponsored search. Here our framework performs vector learning for a more comprehensive setting, i.e. including URLs, queries, terms, and websites.

Another related area is the study of word embedding. A popular model for estimating neural network language model was proposed in [2]. Word2vec [16] is a development with a simple architecture for efficient training. A development of word2vec maps paragraphs into the same space of words [13], which shares similar architecture with our basic model. In comparison, our work focuses on modeling session graph data, and the session vector is incorporated in the networks to model users' information need. More importantly, our tailor-made enhanced model elegantly projects terms into the same semantic space of search elements. Some other works employed neural networks to learn concept vectors of input text objects for similarity calculation under a supervised setting [25].

3 Problem Formulation

Given a set of search sessions $S = \{s_i\}_{i=1}^{n}$ as training data, we aim at finding a semantic space to model Web search scenario so that the probability of observing the sessions in S is maximized. Let θ denote the parameters of the space (i.e., the semantic vectors of elements in sessions). The log-likelihood objective is defined as follows:

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

where $P(s_i; \theta)$ is the probability of observing s_i in the space.

Let e_j denote an element such as a query or a URL in the session s_i, and $C(e_j)$ denote the context elements of e_j in s_i. Let $\mathbf{v}(e_j)$ denote the vector of e_j, and $\mathbf{v}(s_i)$, having the same dimensionality, denote the information need of the user corresponding to s_i. $\mathbf{v}(s_i)$ is also called session vector. We assume that the probability of e_j only depends on $C(e_j)$ and $\mathbf{v}(s_i)$, and it is denoted as $P(e_j; C(e_j), \mathbf{v}(s_i))$. Therefore, $P(s_i; \theta)$ is 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 element vectors of $C(e_j)$ and $\mathbf{v}(s_i)$ (described later). To summarize, our task is to learn vectors for elements in search sessions so that the objective in Equation 1 is maximized. To do so, we should transform each session into training instances of the form $(e_j, C(e_j), s_i)$ for calculating $P(e_j; C(e_j), \mathbf{v}(s_i))$, and a major task is to define the context $C(e_j)$ of e_j in s_i. For better capturing the structure information in click-through data, we introduce a graph representation of session data. Then we develop two models for our learning task by extending an algorithm of word2vec [16]. [4]

4 Basic Model for Vector Learning

4.1 Session Graph and Training Instances

In a search session, there are several types of elements. A user first issues a query, and some URLs are clicked in the result list. To obtain better results, she may issue more

[4] Existing methods for vector representation learning [2, 15, 20, 16] cannot be readily applied here due to: (1) our training data is a set of sessions and each of them is represented as a graph, while the training data of existing methods is a set of word sequences; (2) a vector capturing users' information need is incorporated into our learning procedure. Moreover, we intend to learn a space that uniformly embeds elements of different types such as queries and URLs.

queries. During browsing a clicked page, the user may also browse other pages in the same website. Thus, the website is also involved as an element of the session.

Session Graph. *A search session graph $\mathcal{G} = \{V, E\}$ is defined as an undirected graph. The vertex set V includes search query, clicked URL, and website. The edges are added bewteen (1) two successive queries; (2) a clicked page and the corresponding query; (3) a website and pages from it; (4) a website and a query that results in pages of this website clicked.*

An example is given in Figure 1(b). With a query q_1, the user clicked two URLs, u_1 and u_2. Thus, the edges (q_1, u_1) and (q_1, u_2) are added. The websites h_1 and h_2 of u_1 and u_2 are involved. 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 q_2 and q_3 and clicked more URLs. $C(e_j)$ is defined as adjacent elements of e_j. For example, we have $C(q_1) = \{q_2, u_1, u_2, h_1, h_2\}$. Each training instance $(e_j, C(e_j), s_i)$ means that the target e_j comes from session s_i with context $C(e_j)$.

4.2 Basic Learning Model

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

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

The network, called **session2vec** (s2v for short), for calculating $P(e_j; C(e_j), \mathbf{v}(s_i))$ is given in Figure 2(a), which basically introduces an auxiliary vector into CBOW model [16], as previously did in [13, 17]. The input layer takes the element vectors of $C(e_j)$ and session vector $\mathbf{v}(s_i)$. In the projection layer, the average of the element vectors [5] is summed with $\mathbf{v}(s_i)$ to get $\mathbf{x}_j$. The output layer contains a Huffman tree with each distinct element in training sessions as a leaf. The more frequent an element is, the shorter its Huffman code is.

Let p denote the path from the root to a leaf e_j and L denote the length of p. Let $v^p_{1:L}$ denote the vertices on p and we have $v^p_L = e_j$. Let $c_{1:L-1}$ be the sequence of binary codewords on p. Let $\boldsymbol{\gamma}_{1:L-1}$ denote the vectors associated with the inner vertices $v^p_{1:L-1}$ on p, each of them has 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 p (going through $L-1$ binary selections). Specifically, at vertex v^p_l, we select the branch having the codeword c_l with probability $P(c_l; \boldsymbol{\gamma}_l, \mathbf{x}_j)$, which is defined with the sigmoid function σ:

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

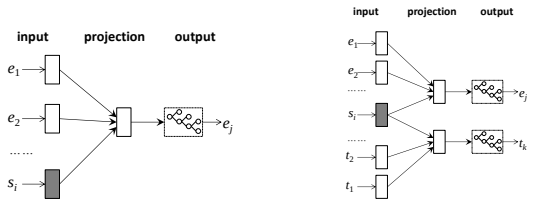
$P(e_j; C(e_j), \mathbf{v}(s_i))$ is calculated as:

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

Thus, combining Equations 5 and 3, the objective can be calculated with the network in Figure 2(a).

We use the SGD algorithm to learn the vectors of elements, inner nodes of Huffman tree, and sessions. During learning, each instance generated in Section 4.1 is fed into the network and its related parameters are updated. The learning procedure is performed by scanning all training instances one or a few times depending on efficiency requirement.

[5] The number of contextual elements varies, so we calculate the average of contextual vectors.



(a) Session2vec (b) Session2vec+

Fig. 2. Our models.

5 Enhanced Model for Vector Learning

With session2vec, each element (i.e., query, URL, and website) in training data is projected as a vector. However, s2v cannot deal with unseen elements in new data. To solve this problem, we propose an enhanced learning model, **session2vec+** (s2v+), as depicted in Figure 2(b). The upper part of s2v+ is the same as s2v. The lower part, having the same architecture, incorporates the term-based training instances in the form of $(t_k, C(t_k), s_i)$ (t_k is a term in s_i). The session vector is shared by two parts as a bridge to align the dimensions of element vectors and term vectors, learnt by the upper and lower parts, respectively. Thus, terms and elements are embedded in the same space, and term vectors can be utilized to build vectors for new elements such as unseen queries. Another advantage of s2v+ is that these term vectors can help solve the sparsity issue in s2v, since the vectors of sparse elements learnt in s2v might be unreliable.

5.1 Training Instances for Term Vector Learning

We build term-based training instances by post-processing element-based instances. Specifically, if e_j of $(e_j, C(e_j), s_i)$ is a query or a URL, it is transformed into a set of term-based instances. Let t_k denote a term in query e_j or the URL title of e_j. Each t_k corresponds to one term-based instance $(t_k, C(t_k), s_i)$ where $C(t_k)$ is the context of t_k containing all terms of queries or URL titles in $C(e_j)$. Noun phrase chunking is done and a single term here may refer to a phrase, e.g. "New York Times". Because t_k could also come from URL titles, our model is augmented to handle unseen query terms with title terms.

5.2 Enhanced Learning Model

For s2v+, we define a new objective function as follows:

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

where $P'(s_i; \theta)$ is the probability of s_i calculated with the term-based instances:

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

where $P(t_k; C(t_k), \mathbf{v}(s_i))$ is the probability of t_k in s_i. Then, $\ell'(\theta; S)$ is written as:

$$\ell'(\theta; S) = \sum_{s_i \in S} \{ \sum_{e_j \in s_i} \sum_{l=1}^{L^e - 1} \log P(c_l^e; \boldsymbol{\gamma}_l^e, \mathbf{x}_j^e) + \sum_{t_k \in s_i} \sum_{l=1}^{L^t - 1} \log P(c_l^t; \boldsymbol{\gamma}_l^t, \mathbf{x}_k^t) \},$$

where superscripts e and t indicate the calculations with element instances and term instances respectively. [6]

Now we derive the gradient of parameters for a single training instance. Two types of training instances from one session are processed separately in each iteration. We first proceed with $(e_j, C(e_j), s_i)$ and let $\ell(j,l) = \log P(c_l^e; \boldsymbol{\gamma}_l^e, \mathbf{x}_j^e)$. After combined with Equation 4, $\ell(j,l)$ is written as:

$$\ell(j,l) = (1 - c_l^e) \log \{\sigma(\mathbf{x}_j^e \boldsymbol{\gamma}_l^e)\} + c_l^e \log \{1 - \sigma(\mathbf{x}_j^e \boldsymbol{\gamma}_l^e)\}. \tag{8}$$

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

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

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

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

$$\boldsymbol{\gamma}_l^e \leftarrow \boldsymbol{\gamma}_l^e + \eta \{1 - c_l^e - \sigma(\mathbf{x}_j^e \boldsymbol{\gamma}_l^e)\} \mathbf{x}_j^e, \tag{11}$$

where η is the learning rate. $\mathbf{x}_j^e$ is an intermediate vector Our aim is to learn $\mathbf{v}(e')$ for $e' \in C(e_j)$, to do so, $\mathbf{v}(e')$ is updated with the partial derivative of $\mathbf{x}_j^e$:

$$\mathbf{v}(e') \leftarrow \mathbf{v}(e') + \eta \sum_{l=1}^{L^e - 1} \{1 - c_l^e - \sigma(\mathbf{x}_j^e \boldsymbol{\gamma}_l^e)\} \boldsymbol{\gamma}_l^e. \tag{12}$$

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 update formulae are:

$$\boldsymbol{\gamma}_l^t \leftarrow \boldsymbol{\gamma}_l^t + \eta \{1 - c_l^t - \sigma(\mathbf{x}_k^t \boldsymbol{\gamma}_l^t)\} \mathbf{x}_k^t, \tag{13}$$

$$\mathbf{v}(t') \leftarrow \mathbf{v}(t') + \eta \sum_{l=1}^{L^t - 1} \{1 - c_l^t - \sigma(\mathbf{x}_k^t \boldsymbol{\gamma}_l^t)\} \boldsymbol{\gamma}_l^t, \tag{14}$$

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

$$\mathbf{v}(s_i) \leftarrow \mathbf{v}(s_i) + \eta \sum_{l=1}^{L^e - 1} \frac{\partial \ell(j,l)}{\partial \mathbf{x}_j^e} + \eta \sum_{l=1}^{L^t - 1} \frac{\partial \ell(k,l)}{\partial \mathbf{x}_k^t}. \tag{15}$$

The learning procedure for s2v can be easily derived by simplifying that of s2v+.

[6] One may notice that both $P(s_i; \theta)$ and $P'(s_i; \theta)$ are defined as probability of s_i and they may be unequal. In fact, refer to Equations 2, 4, and 5, the probability of a session is calculated from element vectors and parameter vectors associated with the Huffman tree. Therefore, it is possible that different types of input vectors, term-based or element-based, output different values. We would not restrict $P(s_i; \theta) = P'(s_i; \theta)$ since such constraint will make the model less flexible in learning vectors for different elements. On the other hand, the session vector $\mathbf{v}(s_i)$, as an intermediary, softly aligns the dimensions of element vectors and term vectors.

Table 1. Salient elements and terms in three dimensions.

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

6 Training Data And Case Study

6.1 Training Data

We employ a query log data set from Baidu search engine, including 10,413,491 unique queries, 13,126,252 URLs, 1,006,352 websites, and 3,965,539 terms (coming from queries and URL titles). Session boundaries are detected with a hybrid method of time-gap-based detection and task-based detection [3, 12]: the interval of two consecutive queries is no more than 15 minutes; and the similarity between two consecutive queries is no less than a threshold. To calculate this similarity, we employ term vectors trained in a baseline system (CBOW of word2vec, described later) to represent query terms and the sum of them is used as query vector. The cosine similarity threshold is 0.5. In total, we collected 23,676,669 sessions, each session contains 2.1 queries and 2.3 clicks on average.

6.2 Case Study

Semantic dimensions. We show salient elements and terms of three dimensions (manually entitled "Star", "Movie resource" and "Software resource"), generated by s2v+, in Table 1. These terms and elements have the largest values in these dimensions, meanwhile the frequency is ≥ 100. For "Star", five singers/actors from mainland China and Hong Kong are output as salient terms. The queries mainly search for the personal information of stars. For websites, the entertainment homepages of five top websites are listed. In "Movie resource", popular movie titles are output as salient terms and the queries are about movies' showtime and scheme song. Interestingly, although "Star" and "Movie resource" are related, our model generates different salient term sets and query sets for them, focusing on different aspects. Presumably, it is because searching stars and searching movies are two different information needs. The element-based and term-based training instances are generated from individual sessions, thus the two information needs are well identified in learning. The websites involved in these two needs are also different and can help differentiate them to some extent.

Learnt vectors. The term vector "北京大学(Peking University)" and the query vector "Peking University" learnt by s2v+ are given in Figure 3. The two vectors are generally correlated well (cosine similarity is 0.591). Thus, we can reasonably derive the vector of an unseen query with term vectors. The two vectors also show some differences. The reason is that "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 query "Peking University", the dominant information need is to find the university's homepage or encyclopedia page.

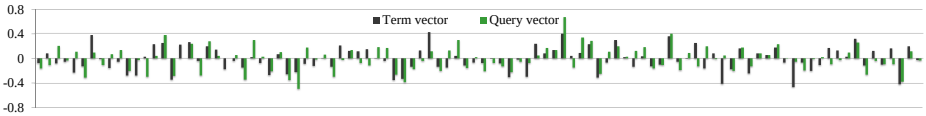


Fig. 3. The term vector of "Peking University" and the query vector of "Peking University".

7 Quantitative Experimental Results

7.1 Settings

Variants of our framework. S2v+ can generate vectors of elements and terms (from queries and URL titles). According to how to use these vectors, we have three variants: **S2v+.A** directly uses element vectors; **S2v+.B** interpolates an element vector and the term vectors from this element. For instance, for query q, we first calculate the sum of its term vectors, then the sum is summed with $\mathbf{v}(q)$, and the result is used as the final vector of q; **S2v+.C** uses the sum of term vectors of an element as its vector, and it is applicable for both existing elements in the training data and new elements.

Comparison systems. We employ the CBOW algorithm of word2vec [7] (w2v for short) as a baseline and run it on a corpus containing 1 billion Chinese Web pages (much larger than the training data used in our model), and a vector is generated for each term. PLSA [9] is another baseline, and we run it on a pseudo-document corpus generated from our training data. Each pseudo-document is composed of queries and URL titles of a training session, and topic vectors of terms are learnt.

7.2 Results for Query Similarity Prediction

We analyze our framework with a similar query ranking task to illustrate the behaviors of its variants. NDCG [11] is employed as the metric and 100 dimensions are used for all systems.

Task Description and Evaluation Data Each testing query has 5 candidate queries, and the task is to rank the candidates according to their similarity with the testing query. Cosine similarity is calculated with the learnt vectors.

We employ an annotated data set containing 500 testing queries, each of which has 5 candidate queries. A Likert scale with three levels is employed to assess the candidates. Specifically, 3 means strongly relevant (e.g. "Bill Gates" and "founder of Microsoft"), 2 means relevant (e.g. "Bill Gates" and "Steve Jobs"), and 1 means irrelevant, (e.g. "Bill Gates" and "Spider-Man"). Each candidate is assessed by 3 assessors and the average score is rounded to the nearest relevance level. On average, each testing query has 1.7 strongly relevant candidates and 1.2 relevant candidates. These 500 testing queries are divided into observed part (**Q_obs**) and unobserved part (**Q_unobs**). Q_obs has 129 testing queries, each testing query and its candidate queries are observed in our training data. Q_unobs has 371 testing queries.

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

Table 2. Results of query similarity prediction on Q_obs.

	w2v	PLSA	s2v	s2v+.A	s2v+.B	s2v+.C
NDCG@1	0.769	0.727	0.784	0.792	0.797	0.799
NDCG@3	0.804	0.786	0.824	0.830	0.834	0.835
NDCG@5	0.833	0.810	0.838	0.841	0.849	0.853

Analysis of S2v Results on Q_obs For s2v, query vectors are directly learnt, and for the baselines, the vector of a query is obtained by summing its term vectors. The results of different methods are given in the left part of Table 2. S2v can outperform the baselines. Specifically, on NDCG@1, s2v outperforms PLSA and w2v by about 8% (significant with $P < 0.01$ in paired t-test) and 2% ($P < 0.05$), respectively. The reasons might be: (1) 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 a training instance are semantically more cohesive. Such training instances bring in less noise; (2) PLSA and w2v generate query vectors by summing the term vectors, however, their term vectors are learnt without considering query and session semantics and cannot well derive query vector. In contrast, s2v directly generates query vectors; (3) s2v maintains a session vector, and the semantic of a session is normally less ambiguous than a query. Thus, the session vector is helpful to guide vector learning for queries by deriving more precise information need. W2v also performs well, and its large training corpus helps deal with sparse queries more effectively.

Analysis of S2v+ Results Sparsity will hinder the effectiveness of learnt embeddings by s2v. In addition, if a query was not observed in the training data, s2v cannot learn a vector for it. S2v+ conducts vector learning for terms in a unified model. The learnt term vectors can be used in different variants as described in Section 7.1.

Results on Q_obs. To examine the effectiveness of s2v+, we first compare its variants with s2v on Q_obs and the results are given in the right part of Table 2. S2v+.A outperforms s2v by 1% on NDCG@1 ($P < 0.05$). This shows that the unified learning in s2v+ generates better vector representation for queries. It is because the lower part of the network in Figure 2(b) for term vector learning can help overcome the sparsity problem 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. As a result, accurate session vectors will help learn better query vectors. S2v+.B slightly outperforms s2v+.A, which shows using term vectors to interpolate the query vector can further solve the sparsity problem.

S2v+.C is the most effective one. It shows that the sum of term vectors generated by s2v+ can better derive the query vector. It is probably because the unified learning in s2v+ can better align the semantic meanings of queries and terms with the session vector as bridge. The term vectors from the baselines are not as effective as ours for deriving query vectors. S2v+.C performs better than s2v+.A and s2v+.B. The reason is that s2v+.A and s2v+.B use query vectors, but the sparsity problem affects the reliability of vectors of low-frequency queries. To have a closer look at the sparsity problem, we divide Q_obs into 5 equal buckets, A, B, C, D, and E, according to the descending order of frequency. Similarly, candidates queries are also divided into 5 buckets, A', B', C', D', and E'. We evaluate the variants in different intervals and the results are shown

Table 3. 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**

in Table 3. In each cell, the results of s2v+.A, s2v+.B, and s2v+.C are given in the upper, middle, and lower positions. The largest value is underscored, in bold and green, the smallest value is in italic and red. As shown in the upper left of Table 3, S2v+.A and s2v+.B perform better for more frequent queries, When the queries become less frequent, moving toward the lower right corner, 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 Q_unobs and compare it with w2v and PLSA baselines. The results are given in Table 4. S2v+.C achieves 8% and 4% improvements ($P < 0.05$) on NDCG@1 compared with PLSA and w2v, respectively. This demonstrates term vectors generated with our model are more effective due to the unified learning and introducing the session vector. Combining the results in Tables 2 and 4, s2v+.C is the most effective system.

7.3 Results for URL Ranking

Setup. Here we examine the performance of our model in the task of URL ranking. The relevance between a query and its candidate URLs is computed as cosine similarity of their vectors. Still, a query vector is obtained by summing the vectors of its terms. For each URL, its vector is obtained by summing the vectors of terms in its title. We introduce another baseline BM25 [19] which is a ranking function commonly used to rank documents according to their relevance to a search query. Specifically, our BM25 baseline is a revision of the original BM25 formula to conduct normalization of term frequency according to [22] and revise inverse document frequency according to [5]. As discussed above, s2v+.C is the most effective variant and it also has better adaptability for unseen data. In addition, URL vectors also face the sparsity problem. Therefore, we conduct the comparison between s2v+.C and baselines.

Evaluation data. This data set has 1,000 queries of various length and popularity. On average, each query has 19.8 marked URLs retrieved by the query. A Likert scale with five levels is employed to assess the relevance of each URL.

Results. The results are given in Table 5. All vector-based methods can outperform BM25. Our model achieves the best results in all cases. Specifically, it outperforms the baselines by about 4% to 9% on NDCG@1 ($P < 0.01$). Recall that we train s2v+ with term-based training instances (together with element-based) from both URL titles and queries. Presumably, such mixed instances make the learnt term vectors more capable for capturing the similarity between queries and URLs. Another reason might be that

Table 4. Results of query similarity on Q_unobs.

	s2v+.C	w2v	PLSA
NDCG@1	0.798	0.766	0.736
NDCG@3	0.836	0.812	0.787
NDCG@5	0.852	0.837	0.815

Table 5. Results for ranking the result URLs.

	s2v+.C	w2v	PLSA	BM25
NDCG@1	0.611	0.587	0.576	0.559
NDCG@3	0.632	0.615	0.607	0.582
NDCG@5	0.640	0.631	0.630	0.616

Table 6. Results for the prediction of website similarity.

	s2v+.S	s2v+.T	s2v.S	w2v	PLSA
NDCG@1	0.794	0.786	0.791	0.772	0.719
NDCG@3	0.855	0.843	0.849	0.832	0.763
NDCG@5	0.883	0.880	0.881	0.870	0.794

s2v+ jointly considers different types of elements (such as queries and URLs) in learning, thus the learnt term vectors can implicitly encode the semantic similarity among these elements to some extent.

7.4 Results for Website Similarity Prediction

Setup. In this task, the vectors from different systems are employed to calculate website similarity. For PLSA and w2v, the vector of a website is obtained by summing the terms vectors of its homepage title. Our model has three variants, namely, s2v.S, s2v+.S, and s2v+.T. S2v.S and s2v+.S use the learnt website vectors directly. S2v+.T uses website vectors by summing term vectors, as is done for baselines.

Evaluation data. This data set contains 500 testing websites with different popularity. Each testing website has 5 candidate websites. A Likert scale with three levels is employed to assess the candidate websites. Specifically, 3 means strongly relevant (e.g. "sports.sina.com.cn" and "sports. qq.com"), 2 means relevant (e.g. "sports.sina.com.cn" and "www.sina.com.cn"), and 1 means irrelevant, (e.g. "sports. sina.com.cn" and "mil.qq.com"). On average, each testing website has 1.6 strongly relevant candidates, and 1.4 relevant candidates. All the testing and candidate websites are covered by our training data set.

Results. The results are given in Table 6. The variants of our model outperform the baselines. Specifically, s2v+.S achieves 3% to 10% improvements ($P < 0.05$) on NDCG@1 compared with baselines. Among the variants, s2v+.S and s2v.S perform better than s2v+.T. It shows that the directly learnt website vectors are more effective than summing term vectors of titles for similarity prediction. This observation is different from that of queries. One reason might be that the sparsity problem for websites is not severe in training data. Another possible reason is that homepage titles, such as "The best car website in China", contain irrelevant terms.

8 Conclusions and Future Work

In this paper, we proposed a framework to uncover a semantic space for Web search. We develop two neural-network-based models, i.e. session2vec and session2vec+, to learn vectors for elements and terms. Compared with previous studies, our framework can perform hidden semantic learning for different types of elements. Moreover, our models enable the learning of vector representation on graph data. Experimental results indicate that the learnt vectors are helpful for a few tasks in Web search. For the future work, one direction is to extend our framework to model the interest profile of users. Another direction is to enhance the session graph with the information of click order and dwell time. A third direction is to derive the real-time information need with the partial information of the current session.

References

1. Baroni, M., Dinu, G., Kruszewski, G.: Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In: ACL. pp. 238–247 (2014)
2. Bengio, Y., Ducharme, R., Vincent, P., Janvin, C.: A neural probabilistic language model. J. Mach. Learn. Res. 3, 1137–1155 (2003)
3. Bing, L., Lam, W., Wong, T.L., Jameel, S.: Web query reformulation via joint modeling of latent topic dependency and term context. ACM Trans. Inf. Syst. 33(2) (2015)
4. Craswell, N., Szummer, M.: Random walks on the click graph. In: SIGIR. pp. 239–246 (2007)
5. Fang, H., Tao, T., Zhai, C.: A formal study of information retrieval heuristics. In: SIGIR. pp. 49–56 (2004)
6. Gao, J., He, X., Nie, J.Y.: Clickthrough-based translation models for web search: from word models to phrase models. In: CIKM (2010)
7. Gao, J., Toutanova, K., Yih, W.t.: Clickthrough-based latent semantic models for web search. pp. 675–684. SIGIR (2011)
8. Grbovic, M., Djuric, N., Radosavljevic, V., Silvestri, F., Bhamidipati, N.: Context- and content-aware embeddings for query rewriting in sponsored search. In: SIGIR '15 (2015)
9. Hofmann, T.: Unsupervised learning by probabilistic latent semantic analysis. Mach. Learn. 42(1-2), 177–196 (2001)
10. Huang, P.S., He, X., Gao, J., Deng, L., Acero, A., Heck, L.: Learning deep structured semantic models for web search using clickthrough data. In: CIKM. pp. 2333–2338 (2013)
11. Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of ir techniques. ACM Trans. Inf. Syst. 20(4), 422–446 (2002)
12. Jones, R., Klinkner, K.L.: Beyond the session timeout: Automatic hierarchical segmentation of search topics in query logs. In: CIKM '08. pp. 699–708 (2008)
13. Le, Q.V., Mikolov, T.: Distributed representations of sentences and documents. In: ICML. pp. 1188–1196 (2014)
14. Lee, S., Hu, Y.: Joint embedding of query and ad by leveraging implicit feedback. In: EMNLP. pp. 482–491 (2015)
15. Mikolov, T.: Statistical Language Models Based on Neural Networks. Ph.D. thesis (2012)
16. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: NIPS. pp. 3111–3119 (2013)
17. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: EMNLP. pp. 1532–1543 (2014)
18. Ren, X., Wang, Y., Yu, X., Yan, J., Chen, Z., Han, J.: Heterogeneous graph-based intent learning with queries, web pages and wikipedia concepts. In: WSDM. pp. 23–32 (2014)
19. Robertson, S.E., Walker, S., Jones, S., Hancock-Beaulieu, M., Gatford, M.: Okapi at trec-3. In: TREC. pp. 109–126 (1994)
20. Schwenk, H.: Continuous space language models. Comput. Speech Lang. 21(3), 492–518 (2007)
21. Shen, Y., He, X., Gao, J., Deng, L., Mesnil, G.: A latent semantic model with convolutional-pooling structure for information retrieval. In: CIKM. pp. 101–110 (2014)
22. Singhal, A., Buckley, C., Mitra, M.: Pivoted document length normalization. In: SIGIR. pp. 21–29 (1996)
23. Socher, R., Bengio, Y., Manning, C.D.: Deep learning for nlp (without magic). In: Tutorial Abstracts of ACL 2012. pp. 5–5. ACL '12 (2012)
24. Wu, W., Li, H., Xu, J.: Learning query and document similarities from click-through bipartite graph with metadata. In: WSDM. pp. 687–696 (2013)
25. Yih, W.t., Toutanova, K., Platt, J.C., Meek, C.: Learning discriminative projections for text similarity measures. In: CoNLL. pp. 247–256 (2011)