Expert Systems With Applications 123 (2019) 178–194

Contents lists available at ScienceDirect

Expert Systems With Applications

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

An integrated service recommendation approach for service-based system development

Fang Xie[a,b], Jian Wang[a,*], Ruibin Xiong[a], Neng Zhang[a], Yutao Ma[a], Keqing He[a]

[a] School of Computer Science, Wuhan University, Wuhan, China
[b] School of Computer Science, Hubei University of Technology, Wuhan, China

ARTICLE INFO

Article history:
Received 2 June 2018
Revised 1 November 2018
Accepted 6 January 2019
Available online 7 January 2019

Keywords:
Service recommendation
Service-based system
Heterogeneous information network
Word embedding
Collaborative filtering

ABSTRACT

With the wide adoption of service-oriented computing and cloud computing, service-based systems (SBSs), a kind of software systems that can offer certain functionalities by leveraging one or more Web services, become increasingly popular. A challenging issue in SBS development is to find suitable services from a variety of available (semantics different) services. Towards this issue, we propose a new service recommendation approach that can integrate diverse information of SBSs and their component services. In this research, SBSs, services, their respective attributes (e.g. content and categories) and SBS-service composition relations are modeled as a heterogeneous information network (HIN); and several semantic similarities between SBSs are measured on a set of meta-paths in the HIN. Particularly, a word embedding technique is used to learn word vectors from the content of SBSs and services, which contribute to better functional similarities between SBSs. Afterwards, the combinational weights of different similarities are optimized using a Bayesian personalized ranking algorithm. Services are finally recommended based on collaborative filtering. We identify two recommendation scenarios with different SBS requirements. By conducting a series of experiments on a real-world dataset crawled from the ProgrammableWeb, we validate the effectiveness of our approach and find out the optimal combinations of SBS similarities for those two scenarios.

1. Introduction

Service-Oriented Computing (SOC) is becoming a significant paradigm for designing, developing, delivering, and consuming software applications (He et al., 2017b; Hu, Peng, Hu, & Yang, 2015; Song & Jacobsen, 2018), which leads to the rapid growth of Web services on the Internet. The popular "everything as a service" model promoted by cloud computing in recent years further speeds up the provisioning of services (He et al., 2017b). For example, as of Jan. 20, 2018, over 19,000 and 8000 services have been published at the public service registries ProgrammableWeb[1] (PW) and Mashape,[2] respectively. A lot of companies have also built their service marketplaces, such as Baidu's APIStore,[3] Amazon's AWS,[4] and Microsoft's Azure.[5] These publically accessible services can be leveraged by software developers to reduce the time and efforts of development and improve the quality of applications (Bano, Zowghi, Ikram, & Niazi, 2014). Service-based systems (SBSs) (He et al., 2017b; Spanoudakis & Zisman, 2010), a kind of software systems that can offer certain functionalities by leveraging one or more Web services (or Web APIs), are thus gained increasing attention in recent years.

A great amount of research works on service discovery, matching, recommendation, and composition have been proposed to facilitate the reuse of services in SBS development. Service discovery (also referred to as service matching/matchmaking) aims to discover services that are functionally similar to user requirements or a specific service (Aznag, Quafafou, & Jarir, 2014; Cassar, Barnaghi, & Moessner, 2014; Chen, Lu, Wu, & Li, 2017a; Lampe, Schulte, Siebenhaar, Schuller, & Steinmetz, 2010; Schulte, Lampe, Eckert, & Steinmetz, 2010; Stavropoulos, Andreadis, Bassiliades, Vrakas, & Vlahavas, 2016; Wang, Gao, Ma, He, & Hung, 2017). Service recommendation intends to recommend services that may probably be

* Corresponding author.
E-mail addresses: fangxie@whu.edu.cn (F. Xie), jianwang@whu.edu.cn (J. Wang), ruibinxiong@whu.edu.cn (R. Xiong), nengzhang@whu.edu.cn (N. Zhang), ytma@whu.edu.cn (Y. Ma), hekeqing@whu.edu.cn (K. He).

[1] http://www.programmableweb.com/.
[2] http://www.mashape.com/.
[3] http://apistore.baidu.com/.
[4] https://aws.amazon.com/marketplace/.
[5] http://azuremarketplace.microsoft.com/.

https://doi.org/10.1016/j.eswa.2019.01.025
0957-4174/© 2019 Elsevier Ltd. All rights reserved.

preferred by a user (Al-hassan, Lu, & Lu, 2015; Hu et al., 2015; Liu, Tang, Zheng, Liu, & Lyu, 2016a; Meng, Dou, Zhang, & Chen, 2014) or services that may probably be composed together to fulfil complex user requirements (Chen, Wu, Jian, Deng, & Wu, 2014; Gao, Chen, Wu, & Bouguettaya, 2016; Huang, Fan, & Tan, 2014; Xia et al., 2015b; Yu, Zhou, Zhang, Wei, & Wang, 2015) by leveraging users' explicit or implicit preferences (e.g. invocation and subscription) on services or historical service composition records (i.e. composite services). Content (including structured or unstructured descriptions represented in texts and WSDL documents) of services and user requirements, if provided, can also be employed for service recommendation (Al-hassan et al., 2015; Gao et al., 2016; Xia et al., 2015a). It is worthy to mention that service recommendation differs from service discovery in two major aspects: 1) unlike service discovery that mainly utilizes the content of services and user requirements, service recommendation further exploits the valuable historical user-service preferences and service compositions; and 2) service discovery focuses on retrieving functional similar services, while service recommendation can find potentially collaborative services. Since user requirements in real scenarios tend to be too complex to be achieved by a single service, service composition that aims to satisfy both functional and non-functional requirements by composing a set of services, becomes another fundamental topic in SBS development (Chen et al., 2014; He et al., 2017a,b; Liang, Chen, Wu, Dong, & Bouguettaya, 2016). As a typical representative of SBSs, mashups (a kind of Web applications developed based on Web services and data sources) become increasingly popular (Xia et al., 2015a). For example, over 7900 mashups have been published at PW. However, building an SBS is generally an intractable task for inexperienced developers because it is hard to find suitable services from a large amount of available services. This challenging issue calls for techniques that can effectively recommend services for SBS development.

In the past two decades, many service recommendation approaches have been proposed. The basic idea of them is to measure the similarities between existing services and user requirements (or preferences) and then generate a service recommendation list based on a set of most similar services. According to the resources employed for similarity measurement, those approaches can be roughly categorized as three groups, namely content-based (Al-hassan et al., 2015; Aznag et al., 2014; Cassar et al., 2014; Chen et al., 2017a; Crasso, Zunino, & Campo, 2011; He et al., 2017a,b; Meng et al., 2014), QoS (Quality-of-Services)-based (Hu et al., 2015; Liu et al., 2016a; Xu, Yin, Deng, Xiong, & Huang, 2016; Zheng, Ma, Lyu, & King, 2013), and social-based (Chen, Paik, & Yen, 2017b; Gao et al., 2016; Huang et al., 2014; Liang et al., 2016). Content-based approaches focus on measuring functional similarities between the content of services and user queries using keyword search or semantics-aware search. Keyword search methods usually have many limitations due to the insufficiencies in identifying semantically relevant keywords. Semantics-aware search methods can be further divided to two subgroups: *logical* ones based on ontologies and *non-logical* ones based on latent factor models (also called topic models) such as LDA (Latent Dirichlet Allocation) (Blei, Ng, & Jordan, 2003). Logical semantics-aware methods require well-defined ontologies and semantic annotation of services and user queries, which makes them hard to apply; while the non-logical methods are generally not very effective due to the coarse-grained semantics captured by topic models. QoS-based approaches measure similarities between services and users' non-functional requirements (e.g. availability and throughput) based on the historical QoS information of services. They are limited by the fact that the QoS of services is difficult to collect (Liang et al., 2016). Social-based approaches focus on measuring the similarities between SBSs, services, and users by analyzing the social relationships among them, e.g. the composite-service network (Huang et al., 2014) and global social service network (Chen et al., 2017b). A major limitation of these approaches is that they mainly exploit the SBS-service composition relations or users' interests on SBSs and services, while neglecting other social information in existing registries, e.g. the providers and user labeled categories of services. In addition, although some of these approaches (Gao et al., 2016) incorporate the content of SBSs and services, they do not provide effective measurement mechanisms on functional similarities.

In this paper, we propose a new service recommendation approach for SBS development, which can integrate diverse information of existing SBSs and services. Specifically, we model the SBSs, services, their attributes (e.g. content, categories, and providers), and SBS-service composition relations in a service registry as a heterogeneous information network (HIN), which is a technique for organizing multiple types of objects and attributes (Sun & Han, 2012). Several semantic similarities between SBSs are then measured on a set of meta-paths in the HIN. In particular, two kinds of functional similarities between SBSs are measured based on the word vectors learned by applying the Word2vec technique (Mikolov, Chen, Corrado, & Dean, 2013) to the content of SBSs and services. Considering that different SBS similarities may have different contributions in service recommendation, we optimize the combinational weights of all measured SBS similarities using a Bayesian personalized ranking (BPR) algorithm (Rendle, Freudenthaler, Gantner, & Schmidt-Thieme, 2009). Finally, given user requirements of SBS development (SBS requirements for simplicity), services are recommended using a collaborative filtering (CF) method. Three main contributions of this research are outlined below:

1) An integrated service recommendation approach is proposed by exploiting diverse information of SBSs and services using the HIN technique. Two typical recommendation scenarios are identified for the first time according to different SBS requirements: a) SBS requirements that contain functional descriptions and some possible predefined categories, and b) SBS requirements that contain functional descriptions, some possible predefined categories, and some component services. Both of these two scenarios are supported by the approach.
2) We propose an effective method for measuring functional similarities between SBSs based on the distributed vector representations of words learned using word embedding techniques.
3) Experiments conducted on a real-world dataset demonstrate the effectiveness of the proposed approach. Moreover, we find out the combinations of SBS similarities that can achieve optimal performance for the two service recommendation scenarios.

The rest of the paper is organized as follows. Section 2 reviews the related work. Section 3 gives the problem definition and an overview of the proposed approach. Two modules of the approach: offline data processing and online service recommendation, are detailed in Sections 4 and 5, respectively. Section 6 presents the experimental results. Conclusions and limitations are discussed in Section 7, as well as the future work.

2. Related work

As stated previously, existing studies on service recommendation can be mainly divided into three groups: content-based, QoS-based, and social-based, which are discussed separately in the following three subsections.

2.1. Content-based service recommendation

Content-based approaches (Al-hassan et al., 2015; Aznag et al., 2014; Cassar et al., 2014; Crasso et al., 2011; He et al., 2017a,b; Meng et al., 2014) focus on estimating functional similarities between the content of services and given user requirements; and then recommending services based on several most similar services of the requirements. A lot of functional similarity measurement methods have been devised. For example, He et al. (2017b) proposed a keyword search method named KS3 to retrieve candidate services for each task that needs to be performed by an SBS. The efficiency of KS3 was further improved by modeling keyword queries as dynamic programming problems (He et al., 2017a). Meng et al. (2014) developed a personalized service recommendation approach, in which users' preferences were indicated by the keywords extracted from their reviews. These *keyword-based* approaches are easy to implement, however they are unable to recommend semantically relevant services.

To overcome the limitation of keyword-based approaches, many semantics-aware service recommendation approaches have been proposed, which can be divided into two subgroups: *logical* and *non-logical*. Logical approaches (Al-hassan et al., 2015; Yu et al., 2015; Subbulakshmi, Ramar, Shaji, & Prakash, 2018) are implemented by describing services and user requirements using ontology-based languages (e.g. SAWSDL[6] and OWL-S[7]) and then recommending services by measuring their semantic similarities using logical reasoning algorithms. For instance, Yu et al. (2015) proposed a recommendation approach for services described by OWL-S. Ontologies and the TF-IDF (Term Frequency-Inverse Document Frequency) technique were combined to compute service similarities. Al-hassan et al. (2015) introduced a method for measuring semantic similarities of terms in an ontology. A service recommendation approach was then proposed by combining the method and item-based CF. Although these logical approaches have shown to be able to obtain relatively good performance, they are hard to apply due to the following reasons (Aznag et al., 2014; Cassar et al., 2014; Crasso et al., 2011): a) there is usually unavailable ontologies at hand and the construction of ontologies may be difficult, and b) considerable efforts are required to manually annotate the content of services and queries.

Non-logical semantics-aware approaches (Gao et al., 2016; Hao, Fan, Tan, & Zhang, 2017; Xia et al., 2015a) mainly use topic models (e.g. LDA) to capture the underlying semantics of services and queries. Based on the learned latent topics, semantic similarities between services and queries can be measured to some extent, which can help recommend semantically relevant services. The performance of these approaches is limited since they cannot accurately measure semantic similarities between services and queries due to the coarseness of learned topics.

Compared with existing keyword-based and non-logical semantics-aware approaches, we devise an effective method for evaluating functional similarities between services and user queries by leveraging the semantic word vectors learned using Word2vec.

2.2. QoS-based service recommendation

QoS-based approaches (Hu et al., 2015; Liu et al., 2016a; Zheng et al., 2013) aim at helping users find services that can fulfil their non-functional requirements by leveraging QoS information of services. Since the QoS of services is hard to collect, a key topic of these approaches is to predict unknown QoS values according to some observations. CF is widely adopted for this task. For instance, Zheng et al. (2013) developed a neighborhood integrated matrix factorization method for predicting QoS of services. Hu et al. (2015) developed a random walk algorithm for evaluating indirect similarities between users (and services), and proposed a time-aware CF approach for QoS prediction. In Liu et al. (2016a), QoS of services was predicted using a personalized location-aware CF approach. It determines similar neighbors of a given user (or service) by exploiting users' (or services') locations.

Due to the fact that there is no available QoS information in existing service registries, e.g. PW, we do not consider the QoS of services in the proposed approach.

2.3. Social-based service recommendation

Social-based approaches (Chen et al., 2017b; Gao et al., 2016; Huang et al., 2014; Liang et al., 2016) perform service recommendation by analyzing social relationships among SBSs, services, and users. For example, a three-phase approach was proposed in (Huang et al., 2014) for service recommendation. Two networks, i.e. composition-service network and service-service network, were firstly built from the historical service composition information. A method was then introduced for predicting the evolution of those networks by studying a number of network features. Finally, three kinds of recommendations, e.g. potential composition, were presented based on the predicted networks. Chen et al. (2017b) designed a method for linking distributed services, resulting in a global service network. A social influence-aware approach was then proposed to facilitate service recommendation based on the network. Gao et al. (2016) presented a service recommendation approach by jointly modeling users' historical preferences, mashups' and services' functionalities, as well as mashup-service compositions. Similar to the content-based approaches, a major limitation of these approaches is that they do not provide effective and easy-to-use ways for measuring functional similarities between services and user queries. In addition, they neglect some useful social attributes related to SBSs and services, e.g. user labeled categories and providers.

Table 1 summarizes the state-of-the-art on service recommendation. To overcome those limitations of existing content-based and social-based approaches, we propose in this research a novel integrated service recommendation approach based on the HIN and word embedding techniques. Specifically, HIN is employed for modeling diverse social relationships of SBSs and services; and different semantic similarities between SBSs are measured based on the HIN. A popular word embedding technique, i.e. Word2vec, is used for learning semantic vector representations of words from the content of SBSs and services; afterwards the functional similarities between SBSs and user requirements are measured based on the learned word vectors, which can contribute to better performance of service recommendation than those keyword-based and topic model-based approaches (as evaluated in Section 6.4).

Moreover, this work is inspired from the work (Liang et al., 2016), which also developed a service recommendation approach by modeling mashups, services, and their attributes as a HIN. Our work improves theirs in four aspects. First, unlike they measured functional similarities between mashups based on the latent topics learned from the content of mashups and services, we devise a more effective functional similarity measurement method based on the word vectors learned using Word2vec. Second, we cluster SBSs using topic models, such that the time required for service recommendation can be greatly reduced by only seeking similar SBSs of an SBS requirement in several most similar clusters of it. Third, we identify two practical service recommendation scenarios and describe in detail how to adapt our approach to address each scenario. Fourth, we conduct extensive experiments for testing various combinations of the SBS similarities measured on six

[6] https://www.w3.org/TR/sawsdl/.
[7] https://www.w3.org/Submission/OWL-S/.

Table 1
Summary of the state-of-the-art on service recommendation.

Approaches	Description	Recommended service type	Typical techniques	Example references	Limitations
Content-based	Measure functional similarities between the content of services and user queries	Functional similar services	Keyword-based matching	He et al. (2017a), Meng et al. (2014)	Cannot recommend semantically relevant services
			Logical semantics-aware matching based on ontologies	Yu et al. (2015) Alhassan et al. (2015)	Usually lack of well-defined ontologies; Much efforts are required for manual semantic annotation
			Non-logical semantics-aware matching using topic models	Gao et al. (2016), Hao et al. (2017)	Low accuracy
QoS-based	Compare the QoS of services with users' non-functional requirements	Non-functional similar services	CF with the context of users and services	Hu et al. (2015), Liu et al. (2016a)	QoS of services are usually unavailable
Social-based	Analyze social relationships among the SBSs, services, and users	A set of services that could be composed; User preferred services	Composite-service network, global social service network, etc.	Chen et al. (2017b), Gao et al. (2016)	Lack of effective and easy-to-use methods for measuring functional similarities; Neglect some useful social attributes

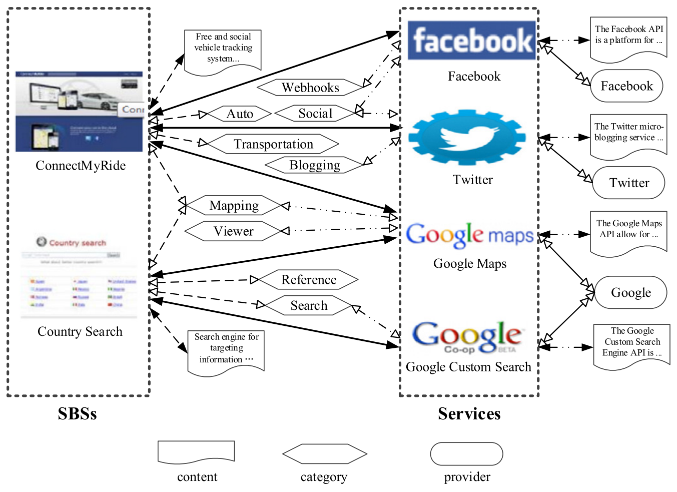

Free and social vehicle tracking system...
Webhooks
Auto
Social
Transportation
Blogging
Mapping
Viewer
Reference
Search
Search engine for targeting information ···
ConnectMyRide
Country Search
SBSs
Facebook
Twitter
Google Maps
Google Custom Search
Services
The Facebook API is a platform for ...
Facebook
The Twitter micro-blogging service ...
Twitter
The Google Maps API allow for ...
Google
The Google Custom Search Engine API is ...
content
category
provider

Fig. 1. A part of HMSN in PW.

meta-paths. Based on the results, we determine the optimal combinations of SBS similarities for the two scenarios.

3. Problem definition and approach overview

In this section, we define the problem of service recommendation for SBS development based on a service registry and present the overview of our approach.

3.1. Problem definition

In current service registries, there exist various types of objects (e.g. SBSs and services) and rich relationships among them, which can be modeled using the HIN technique (Liang et al., 2016; Sun & Han, 2012). Fig. 1 shows a part of the heterogeneous SBS-service network (denoted by HMSN) in PW, which involves two SBSs, four services, three kinds of attributes (i.e. content, categories, and providers) related to them, and the composition relations among them. For example, both SBSs "Google Maps" and "Google Custom Search" are provided by *Google*; SBSs "ConnectMyRide" and "Country Search" share the same category *Mapping*.

3.1.1. Definition (HMSN)

Given a service registry *SR*, the HMSN of it can be essentially defined as $HMSN = \langle O, R\rangle$, where O denotes the entire set of objects in *SR* and *R* denotes the entire set of relations in *SR*. Specifically, $O = \langle M, S, Co, Ca, Pro\rangle$ and $R = \langle MS, MCo, MCa, SCo, SCa, SPro\rangle$, where M, S, Co, Ca, and Pro denote five main sets of objects, i.e. SBSs, services, content, categories, and providers, respectively; MS, MCo, MCa, SCo, SCa, and $SPro$ denote six main sets of relations, i.e. the *composing*/*composed-by* relations between SBSs and services, the *describing*/*described-by* relations between SBSs and their content, the *labeling*/*labeled-with* relations between SBSs and their categories, the *describing*/*described-by* relations between services and their content, the *labeling*/*labeled-with* relations between services and their categories, and the *providing*/*provided-by* relations between services and their providers. For example, $(m_i, s_j) \in MS$ means that service s_j is a component service of SBS m_i, and $(m_i, c_k) \in MCo$ means that m_i has content c_k.

A concept of "network schema" (Sun & Han, 2012) is introduced for describing the meta-structure (i.e. object types and relationship types) of a HIN. For example, Fig. 2 depicts the network schema of the HMSN in Fig. 1.

As defined above, we investigate four main types of information (namely the composition relations and three kinds of attributes)

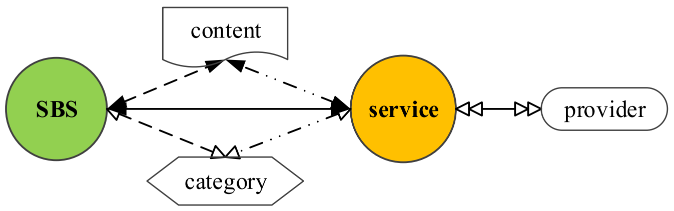


Fig. 2. Network schema of the HMSN in Fig. 1.

associated with SBSs and services in existing service registries. Other technical attributes of services like the architectural style (e.g. REST and SOAP) and request/response formats (e.g. XML and JSON) are not taken into account because we do not want to discuss the technical factors in service recommendation.

Using the definition above, the service recommendation problem is defined as follows.

3.1.2. Problem (Service recommendation)

Given a service registry *SR* and its HMSN, for an SBS requirement *MReq*, the objective of service recommendation is to generate a ranked service list from *S*. A service ranked higher in the list will be more likely to be used for creating the target SBS.

In reality, there can be two types of SBS requirements. Let us consider the process of creating an SBS. At first, the developer may only have a description of the functionalities to be implemented by the SBS. Note that if the service registry to be explored has predefined categories for services, the developer can add some relevant categories as a part of the SBS requirements. By sending the requirements to the service recommendation engine, a ranked service list will be returned. From the recommendations, the developer may select some services of interest that could be used to create the SBS. If the selected services are insufficient, the developer can send another service recommendation request to the engine. At this time, the selected services can be included in the new SBS requirements. This process stops when the selected services are sufficient or the developer chooses to quit the iteration.

According to the process of SBS development described above, two service recommendation scenarios with different SBS requirements can be identified as follows.

1) *SRec*-1: Recommend services for SBS requirements with functional description (i.e. textual description that depicts the functionality requirements of an SBS) and some possible categories;
2) *SRec*-2: Recommend services for SBS requirements with functional description, some possible categories, and some services.

The second scenario *SRec*-2 will also occur when an existing SBS needs to be extended by adding new services or updated by replacing some out-of-date component services. Both *SRec*-1 and *SRec*-2 are handled and evaluated in this work, as detailed in Sections 5 and 6.

3.2. Approach overview

Our proposed service recommendation approach consists of two modules, as illustrated in Fig. 3. In the offline data processing module, given a service registry *SR*, the content of SBSs *M* and services *S* are preprocessed at first. Three steps are then conducted. First, we apply LDA to the SBSs and group them into clusters according to their relevant latent topics. Second, we construct the HMSN of *SR* by modeling the SBSs, services, their attributes, and SBS-service composition relations using the HIN technique. Third, Word2vec is applied to the content of SBSs and services. Based on the HMSN and the learned word vectors, we measure several semantic similarities between the SBSs on a set of meta-paths in the HMSN. Af-

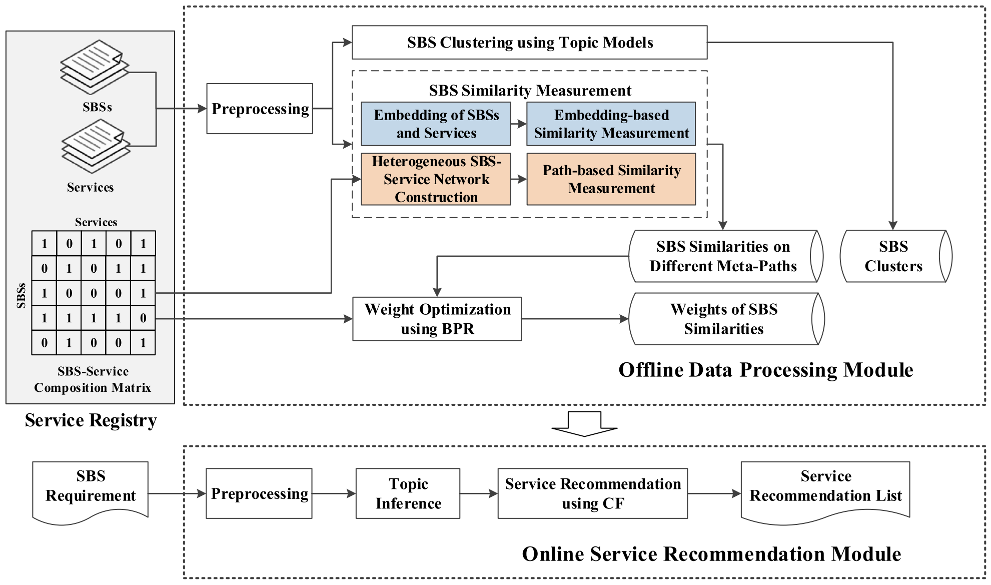

SBSs
Services
Services
SBSs
SBS-Service Composition Matrix
Service Registry
Preprocessing
SBS Clustering using Topic Models
SBS Similarity Measurement
Embedding of SBSs and Services
Embedding-based Similarity Measurement
Heterogeneous SBS-Service Network Construction
Path-based Similarity Measurement
SBS Similarities on Different Meta-Paths
SBS Clusters
Weight Optimization using BPR
Weights of SBS Similarities
Offline Data Processing Module
SBS Requirement
Preprocessing
Topic Inference
Service Recommendation using CF
Service Recommendation List
Online Service Recommendation Module

Fig. 3. The proposed service recommendation approach.

ter that, the combinational weights of those similarities for service recommendation are optimized using a BPR algorithm.

In the online module, when an SBS requirement *MReq* is sent to the service recommendation engine, the content of *MReq* is firstly preprocessed. Then, the topic distribution of *MReq* is inferred using a technique called *Folding-in* (Cassar et al., 2014) based on the LDA model trained for SBSs; and the candidate set of SBSs is restricted to the clusters that correspond to several most relevant topics of *MReq*, so as to improve the time efficiency. Afterwards, several similarities between *MReq* and each candidate SBS *m* are measured as in the offline module. As stated before, there can be two types of SBS requirements (in *SRec*-1 and *SRec*-2). The available set of similarities that can be measured relies on the type of *MReq*. An overall similarity between *MReq* and *m* is computed by integrating the different similarities with respect to their weights. A service recommendation list is finally generated by predicting the ratings of *MReq* on all services in *S* using a CF method.

Note that the techniques and algorithms, e.g. LDA, Word2vec, BPR, and *Folding-in*, adopted in this work are selected according to their popularity in the literature. Other similar techniques and algorithms can also be used in the proposed approach, e.g. the topic models PLSA (Probabilistic Latent Semantic Analysis) (Hofmann, 1999) and BTM (Bi-term Topic Model) (Chen et al., 2014), the Glove word embedding technique (Pennington, Socher, & Manning, 2014), the adaptive BPR (Pan, Zhong, Xu, & Ming, 2015), and the topic inference algorithm proposed in (Yao, Mimno, & Mccallum, 2009). Moreover, some existing methods for functional similarity measurement can also be employed or integrated in our approach, as demonstrated in the experiments.

4. Offline data processing

This section introduces the offline data processing module for a given service registry, which comprises four main phases, namely preprocessing, SBS clustering using topic models, SBS similarity measurement, and weight optimization using BPR.

4.1. Preprocessing

At first, we preprocess the content of each SBS (and service) in the service registry using the following three steps.

4.1.1. Word segmentation

Words contained in the content are extracted using the NLTK,[8] a popular Python toolkit developed for natural language processing.

4.1.2. Lemmatization

Inflected words are normalized to their basic form using the WordNet Lemmatizer in NTLK. For example, “create,” “creates,” “created,” and “creating” will be transformed into “create.”

4.1.3. Stop word removal

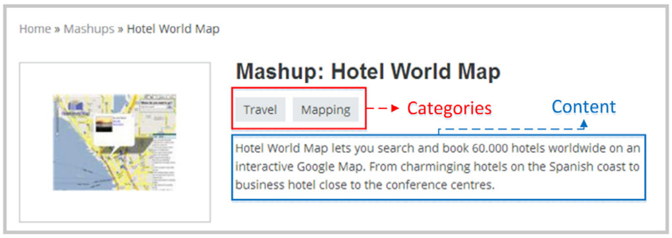
Stop words are removed based on the built-in stop word list in NLTK.

4.2. SBS clustering using topic models

CF, a technique widely used in recommendation systems, is adopted for service recommendation in this work. Referring to the idea of CF, a set of similar SBSs needs to be obtained for an SBS requirement *MReq*. However, there can be a large number of SBSs in the service registry; and it will require much time to compute the similarity between each available SBS and *MReq*. To improve the efficiency, a general solution is to group the SBSs into clusters in advance and then reduce the search space to several most similar clusters of *MReq*. Although SBSs in service registries are usually organized by predefined categories, a problem should be pointed out is that: there can be hundreds of predefined categories (e.g. PW consists of about 460 predefined categories like *Travel*, *Hotels*, *Mapping*, and *Weather*) and the categories of SBSs and services are manually labeled by users (registry managers or SBS providers); thus, it is often difficult to label an SBS (or a service) with all relevant categories. Fig. 4 shows the content and categories of an SBS “Hotel World Map” published in the PW. As can be seen from the content, apart from the two labeled categories *Travel* and *Mapping*, the SBS is also highly relevant to the category *Hotels* that has not been assigned to it. Therefore, it is not suitable to restrict the SBS search space of SBS requirements according to the predefined categories, which may lead to the missing of some similar SBSs.

Recently, topic models have been widely employed in service clustering (Aznag et al., 2014; Chen, Wang, Yu, Zheng, & Wu, 2013;

[8] http://www.nltk.org/.


Home » Mashups » Hotel World Map

Mashup: Hotel World Map

Travel Mapping

Hotel World Map lets you search and book 60.000 hotels worldwide on an interactive Google Map. From charminging hotels on the Spanish coast to business hotel close to the conference centres.

Fig. 4. Example SBS (Mashup) published in the PW.

Wang et al., 2017). To solve the above issue, we propose to cluster SBSs using the LDA topic model. The produced SBS clusters are subsequently used to reduce the search space. LDA (Blei et al., 2003) is a generative probabilistic model developed for a corpus (e.g. a collection of documents). Each item in the corpus is modeled as a finite mixture over a set of latent topics, while each topic is characterized by a distribution over words.

Here, we apply the LDA algorithm to the preprocessed content of SBSs. The parameters of LDA, i.e. topic-word distribution and SBS-topic distribution, can be estimated using several methods such as the variational Expectation-Maximization and Gibbs sampling (Blei et al., 2003). Take the Gibbs sampling method as an example, it starts with randomly assigning topics to all words in the corpus, denoted by $\mathbf{w} = \{w_1, w_2, ..., w_N\}$ (word w_i belongs to some SBS document d_i), and then continues resampling the topic of each word w_i in $\mathbf{w}$ by

$$p(z_i = t|\mathbf{z}_{-i}, \mathbf{w}) \propto \frac{n_{-i,t}^{(w_i)} + \beta}{n_{-i,t}^{(\cdot)} + |V|\beta} \frac{n_{-i,t}^{(d_i)} + \alpha}{n_{-i,\cdot}^{(d_i)} + T\alpha}, \ \forall t \in \{1, \ 2, \ldots, T\}, \tag{1}$$

where z_i denotes the topic of w_i; $\mathbf{z}_{-i}$ denotes the topics of all words except w_i; $n_{-i,t}^{(w_i)}$ is the number of instances of w_i assigned to topic t, excluding the current instance; V denotes the set of distinct words in $\mathbf{w}$; and $n_{-i,t}^{(d_i)}$ is the number of words in d_i assigned to t, except w_i. T is the number of topics. α and β are two hyperparameters.

After passing the "burn-in" period (by resampling the topics of all words for a number of iterations) (Wang, Barnaghi, & Bargiela, 2010), the probability of each topic z in SBS document d, denoted by $\theta_z^{(d)}$, and the probability of each word $w \in V$ under z, denoted by $\varphi_w^{(z)}$, can be estimated as

$$\hat{\theta}_z^{(d)} = \frac{n_z^{(d)} + \alpha}{n_{\cdot}^{(d)} + T\alpha}, \tag{2}$$

$$\hat{\phi}_w^{(z)} = \frac{n_z^{(w)} + \beta}{n_z^{(\cdot)} + |V|\beta}, \tag{3}$$

where $n_z^{(d)}$ is the number of words in d assigned to z and $n_z^{(w)}$ is the number of instances of w assigned to z.

A key issue of LDA is that the topic number T should be pre-determined. The study conducted in Griffiths and Steyvers (2004) reveals that as T increases from a small value, the LDA model can fit the corpus more accurately until an optimal point is reached; thereafter the model's performance degrades as it becomes more complex than necessary. Several methods have been proposed to determine the best T for a given corpus. For example, Griffiths and Steyvers (2004) proposed a Bayesian model selection method for estimating the posterior probabilities $p(\mathbf{w}|\mathbf{z}, T)$ of LDA models trained using a range of T values; and the best T is chosen according to the model that leads to the maximum posterior probability. In Blei et al. (2003) and Liu, Agarwal, Ding, and Yu (2016b), the best T was determined by measuring the likelihood on a held-out test set using the trained models.

In our experiments, the best T is determined using the Bayesian model selection method. Based on the learned topics and SBS-topic distributions, we cluster the SBSs by creating T SBS clusters (a cluster is created for each topic) and assigning each SBS to the clusters that correspond to its relevant topics. For this task, we need to determine the relevant topics of each SBS. However, through observation, the topic distributions of SBSs can be notably different, and it is hard (or even impossible) to set a proper probability threshold to obtain the relevant topics of all SBSs. A simple solution adopted in the literature of topic model-based clustering is to assign each item to the clusters corresponding to its top k^{clu} most relevant topics (Blei et al., 2003). Here, we also adopt the top k^{clu} cluster assignment for SBS clustering. Note that the value of k^{clu} has an impact on the quality of SBS clusters. Specifically, a small k^{clu} will lead to the missing of assigning SBSs to some clusters that correspond to their relevant topics. A large k^{clu} will help assign each SBS to the clusters that correspond to all its relevant topics, but some dissimilar SBSs may be grouped together.

4.3. SBS similarity measurement

This subsection introduces the methods for measuring several kinds of semantic similarities between SBSs by leveraging the HIN and Word2vec techniques.

4.3.1. Heterogeneous SBS-service network construction

As described in Section 3.1, the diverse objects and relations in a service registry can be modeled as a HIN, referred to as HMSN. We can see from the network schema of HMSN (as shown in Fig. 2) that there are multiple types of paths (called meta-paths) between any two objects. For example, two SBSs can be connected via six meta-paths (Liang et al., 2016), as presented in Table 2. Specifically, a meta-path P can be formally denoted as $O_1 \xrightarrow{R_1} O_2 \xrightarrow{R_2} \ldots \xrightarrow{R_l} O_{l+1}$, $O_i \in \mathcal{O}$ and $R_j \in \mathcal{R}$ ($\mathcal{O}$ is the object type set and $\mathcal{R}$ is the relationship type set). Given a path $p = o_1 o_2 \ldots o_{l+1}$, we say p is an instance of meta-path P (denoted as $p \in P$) if $\forall i$, the type of o_i is O_i and each relation $e_i = o_i o_{i+1}$ belongs to R_i. For example, path "ConnectMyRide-Google Maps-Country Search" in Fig. 1 is an instance of meta-path "SBS-service-SBS" in Fig. 2.

The semantics of meta-path P is indicated by its relationship chain $R = R_1 \circ R_2 \circ \ldots \circ R_l$. In this work, six different types of semantic similarities between SBSs are evaluated on the six meta-paths in Table 2, as detailed in Sections 4.3.2 and 4.3.4.

4.3.2. Path-based similarity measurement

Generally speaking, SBSs in a registry are composed by the local services; the set of categories and the set of providers are shared by the SBSs and services. Therefore, semantic similarities between two SBSs, e.g. m_i and m_j, on the four meta-paths {P$_1$, P$_3$, P$_4$, P$_6$} in Table 2 can be measured using the following path-based method (Sun, Han, Yan, Yu, & Wu, 2011).

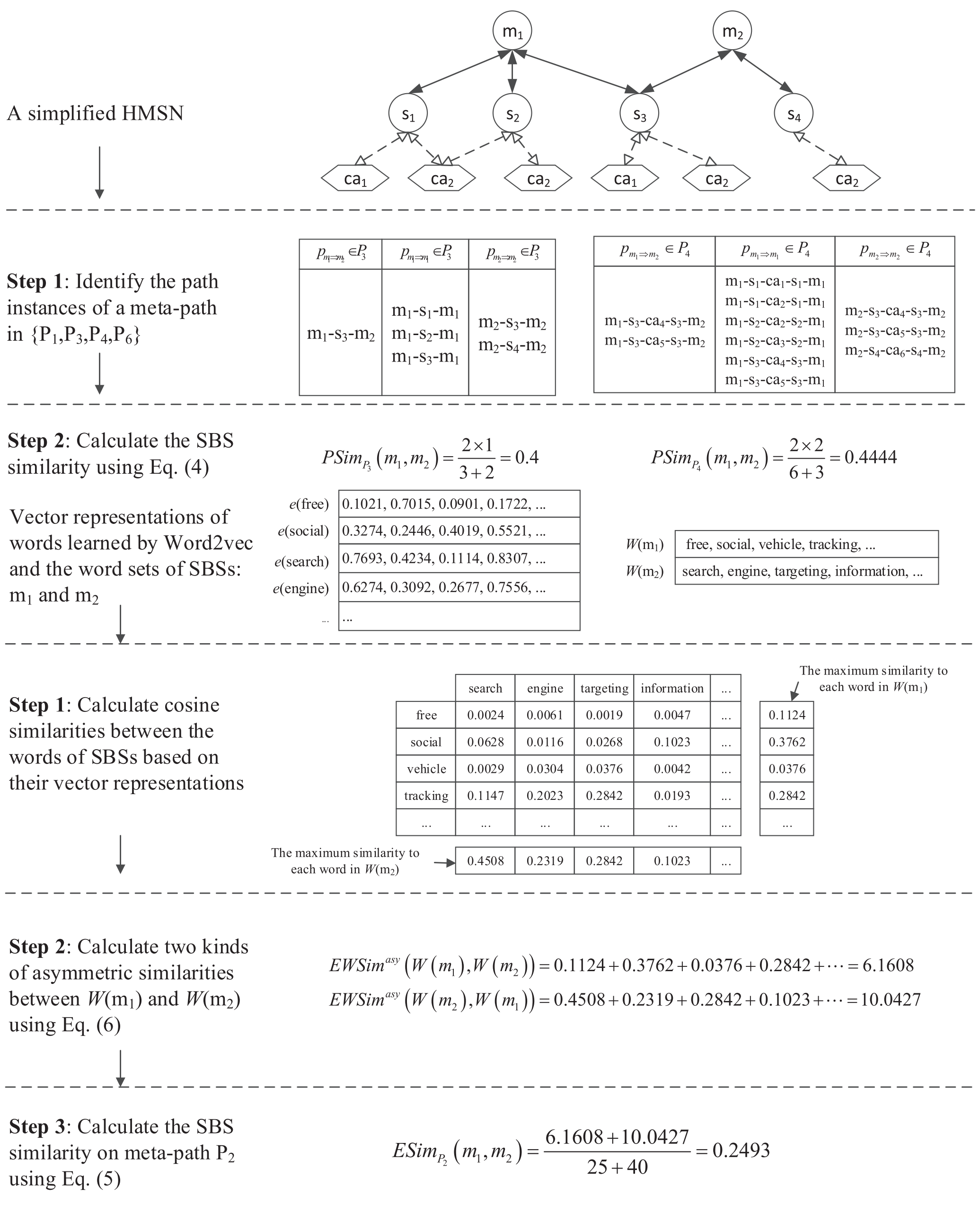
$$PSim_P(m_i, m_j) = \frac{2 \times \left|\left\{p_{m_i \Rightarrow m_j} : p_{m_i \Rightarrow m_j} \in P\right\}\right|}{\left|\left\{p_{m_i \Rightarrow m_i} : p_{m_i \Rightarrow m_i} \in P\right\}\right| + \left|\left\{p_{m_j \Rightarrow m_j} : p_{m_j \Rightarrow m_j} \in P\right\}\right|}, \tag{4}$$

where P is a specific meta-path and $\{p_{x \Rightarrow y}: p_{x \Rightarrow y} \in P\}$ denotes the set of path instances between SBSs x and y following P.

Fig. 5(a) illustrates the computation of the two kinds of SBS similarities on P$_3$ and P$_4$ using Eq. (4), given a simplified HMSN that contains only two SBSs $\{m_1, m_2\}$, four services $s_1 \sim s_4$, and six categories $ca_1 \sim ca_6$.

Table 2
Six meta-paths connecting SBSs.

ID	Meta-path	Semantics
P_1	SBS-category-SBS	Two SBSs are labeled with the same category.
P_2	SBS-content-SBS	Two SBSs are described using the same content.
P_3	SBS-service-SBS	Two SBSs are composed by the same service.
P_4	SBS-service-category-service-SBS	Two SBSs are composed by services that share the same category.
P_5	SBS-service-content-service-SBS	Two SBSs are composed by services that share the same content.
P_6	SBS-service-provider-service-SBS	Two SBSs are composed by services provided by the same provider.



Step 1: Identify the path instances of a meta-path in {P1,P3,P4,P6}

$p_{m_1\Rightarrow m_2} \in P_3$	$p_{m_1\Rightarrow m_1} \in P_3$	$p_{m_2\Rightarrow m_2} \in P_3$
m_1-s_3-m_2	m_1-s_1-m_1 m_1-s_2-m_1 m_1-s_3-m_1	m_2-s_3-m_2 m_2-s_4-m_2

$p_{m_1\Rightarrow m_2} \in P_4$	$p_{m_1\Rightarrow m_1} \in P_4$	$p_{m_2\Rightarrow m_2} \in P_4$
m_1-s_3-ca_4-s_3-m_2 m_1-s_3-ca_5-s_3-m_2	m_1-s_1-ca_1-s_1-m_1 m_1-s_1-ca_2-s_1-m_1 m_1-s_2-ca_2-s_2-m_1 m_1-s_2-ca_3-s_2-m_1 m_1-s_3-ca_4-s_3-m_1 m_1-s_3-ca_5-s_3-m_1	m_2-s_3-ca_4-s_3-m_2 m_2-s_3-ca_5-s_3-m_2 m_2-s_4-ca_6-s_4-m_2

Step 2: Calculate the SBS similarity using Eq. (4)

$$PSim_{P_3}(m_1,m_2)=\frac{2\times 1}{3+2}=0.4 \qquad PSim_{P_4}(m_1,m_2)=\frac{2\times 2}{6+3}=0.4444$$

Vector representations of words learned by Word2vec and the word sets of SBSs: m1 and m2

e(free)	0.1021, 0.7015, 0.0901, 0.1722, ...
e(social)	0.3274, 0.2446, 0.4019, 0.5521, ...
e(search)	0.7693, 0.4234, 0.1114, 0.8307, ...
e(engine)	0.6274, 0.3092, 0.2677, 0.7556, ...
...	...

W(m1)	free, social, vehicle, tracking, ...
W(m2)	search, engine, targeting, information, ...

Step 1: Calculate cosine similarities between the words of SBSs based on their vector representations

	search	engine	targeting	information	...	The maximum similarity to each word in *W*(m1)
free	0.0024	0.0061	0.0019	0.0047	...	0.1124
social	0.0628	0.0116	0.0268	0.1023	...	0.3762
vehicle	0.0029	0.0304	0.0376	0.0042	...	0.0376
tracking	0.1147	0.2023	0.2842	0.0193	...	0.2842
...	...	...	...	...	...	...
The maximum similarity to each word in *W*(m2)	0.4508	0.2319	0.2842	0.1023	...	

Step 2: Calculate two kinds of asymmetric similarities between *W*(m1) and *W*(m2) using Eq. (6)

$$EWSim^{asy}\left(W(m_1),W(m_2)\right)=0.1124+0.3762+0.0376+0.2842+\cdots=6.1608$$
$$EWSim^{asy}\left(W(m_2),W(m_1)\right)=0.4508+0.2319+0.2842+0.1023+\cdots=10.0427$$

Step 3: Calculate the SBS similarity on meta-path P2 using Eq. (5)

$$ESim_{P_2}(m_1,m_2)=\frac{6.1608+10.0427}{25+40}=0.2493$$

Fig. 5. (a). Illustration of the computation of two kinds of SBS similarities on P_3 and P_4 using Eq. (4). (b). Illustration of the computation of SBS similarity on P_2 using Eq. (5).

4.3.3. *Word embedding of SBSs and services*

Since the content of each SBS (or service) is unique, the two types of SBS similarities on meta-paths P_2 and P_5 cannot be measured using the path-based method. Although several methods have been developed for evaluating functional similarities between the content of services (and queries) in existing service recommendation approaches, they all have their own limitations, as discussed in Section 2.1. Here, we devise an effective method for measuring functional similarities between SBSs by utilizing the Word2vec technique (Mikolov et al., 2013), a neural network-based technique for learning semantic vector representations of words from a large amount of text data. In recent years, Word2vec has been widely used to boost the performance of various applications, e.g. information retrieval (Ganguly, Roy, Mitra, & Jones, 2015), clustering (Shi, Liu, Zhou, Tang, & Cao, 2017), and semantic similarity measurement between terms or texts (Kenter & Rijke, 2015; Zhang, Jatowt, & Tanaka, 2016).

We apply Word2vec to the preprocessed content of SBSs and services. The vector representation learned for word w is denoted by $e(w)$.

4.3.4. Embedding-based similarity measurement

Based on the learned word vectors, we adopt a method for measuring the SBS similarities on P$_2$ and P$_5$. Specifically, the similarity between SBSs m_i and m_j on P$_2$ is computed using Eq. (5) (Kenter & Rijke, 2015).

$$ESim_{P_2}\left(m_i,\ m_j\right) = \frac{EWSim^{asy}\left(W(m_i),\ W(m_j)\right) + EWSim^{asy}\left(W(m_j),\ W(m_i)\right)}{\left|W(m_i)\right| + \left|W(m_j)\right|}, \tag{5}$$

where $W(m)$ represents the set of words contained in the preprocessed content of SBS m, and $EWSim^{asy}(W_1,\ W_2)$ calculates an asymmetric similarity of word sets W_1 to W_2:

$$EWSim^{asy}(W_1,\ W_2) = \sum\nolimits_{w_i \in W_1} \max\nolimits_{w_j \in W_2} \frac{e(w_i)\cdot e\left(w_j\right)}{\left\|e(w_i)\right\|\cdot\left\|e\left(w_j\right)\right\|}. \tag{6}$$

Fig. 5(b) illustrates the computation of SBS similarity on P$_2$ using Eq. (5).

As for P$_5$, we collect the content of all component services of each SBS m. The resulting new content is denoted by m'. Similarity between m_i and m_j on P$_5$ is then calculated as $ESim_{P_5}(m_i',\ m_j')$.

4.4. Weight optimization using BPR

Until now, we have introduced how to measure the SBS similarities on all meta-paths in Table 2. Intuitively, the semantic of each meta-path may have its own contribution in service recommendation; and the importance of the six meta-paths can be different, e.g. the content of a service may be more importance than its provider. It is reasonable to assume that better service recommendations might be produced by integrating diverse SBS similarities.

To optimize the combinational weights of different SBS similarities for service recommendation, we adopt a learning to rank algorithm called BPR (Rendle et al., 2009) based on the implicit feedback of SBSs on services (i.e. the SBS-service composition relations MS). By assuming that a service is more likely to be preferred by the SBSs that have used it, the objective of BPR is to generate the correct order of SBSs M for each service $s_i \in S$ according to MS. Specifically, M can be divided to two subsets M_1 and M_2 with respect to s_i. M_1 is the set of SBSs that have used s_i, i.e. $M_1 = \{m_j: (m_j, s_i) \in MS\}$, while M_2 is the set of SBSs that have not used s_i, i.e. $M_2 = M - M_1$. Each SBS in M_1 should be ranked higher than all SBSs in M_2. In order to rank the SBSs, we need to estimate the rating of each SBS $m_g \in M$ on s_i. A CF method is used for this task. Firstly, an overall similarity between m_g and any other SBS in M, e.g. m_h ($h \neq g$), is obtained by integrating the six similarities between them:

$$Sim_{\mathcal{P}}(m_g,\ m_h) = \sum\nolimits_{P_l \in \mathcal{P}} \theta_{P_l}\cdot Sim_{P_l}(m_g,\ m_h), \tag{7}$$

where $\mathcal{P}$ denotes the set of meta-paths in Table 2, $Sim_{P_l}(m_g,\ m_h)$ is the similarity between m_g and m_h on meta-path P_l, and θ_{P_l} is the weight coefficient of P_l.

Based on the overall similarities, a set of top k^{cf} most similar SBSs is obtained for m_g, denoted by $SimM_{\mathcal{P}}(m_g, k^{cf})$. The rating of m_g on s_i is then estimated as

$$\hat{r}_{\mathcal{P}}(m_g,\ s_i) = \sum\nolimits_{m_h \in SimM_{\mathcal{P}}\left(m_g, k^{cf}\right)} Sim_{\mathcal{P}}(m_g, m_h)\cdot r(m_h, s_i), \tag{8}$$

where $r(m_h,\ s_i)$ is the implicit rating of m_h on s_i, which can be obtained from MS, i.e. $r(m_h,\ s_i) = 1$ if $(m_h,\ s_i) \in MS$, otherwise 0.

Given the training data MS, the weight parameters of different meta-paths in Eq. (7), i.e. $\theta = \{\theta_{P_1},\ \theta_{P_2},\ \ldots,\ \theta_{P_6}\}$, can be learned by maximizing the posterior probability $p(\theta|MS) \propto p(MS|\theta)p(\theta)$. Under the assumptions that services are selected independently by an SBS and all SBS pairs of a service are also independent, the likelihood $p(MS|\theta)$ can be represented as

$$p(MS|\theta) = \prod\nolimits_{s_i \in S}\prod\nolimits_{m_j \in M_1,\ m_k \in M_2} p\left(m_j \succ_{s_i} m_k\right), \tag{9}$$

where $p(m_j \succ_{s_i} m_k)$ denotes the probability that SBS m_j is ranked higher than SBS m_k with respect to service s_i, which is defined as $\tau(\hat{r}_{\mathcal{P}}(m_j, s_i) - \hat{r}_{\mathcal{P}}(m_k, s_i))$, τ refers to the logistic sigmoid function.

Then, the objective function of maximizing $p(\theta|MSCM)$ is derived as Eq. (10) with $\theta \sim \mathcal{N}(0, \lambda\boldsymbol{I})$.

$$F = \min_{\theta} - \sum\nolimits_{s_i \in S}\sum\nolimits_{m_j \in M_1, m_k \in M_2} \ln\tau\left(\hat{r}_{\mathcal{P}}\left(m_j, s_i\right) - \hat{r}_{\mathcal{P}}(m_k, s_i)\right) + \frac{\lambda}{2}\|\theta\|_2^2, \tag{10}$$

where $\frac{\lambda}{2}\|\theta\|_2^2$ is the L_2 regularization term.

As the objective function F is differentiable, we use stochastic gradient descent to learn the parameters.

5. Online service recommendation

Once the offline data processing is completed, the online service recommendation module in Fig. 3 will be initialized when an SBS requirement $MReq$ is sent to the service recommendation engine.

5.1. Preprocessing

The content of $MReq$ is firstly preprocessed using the three steps described in Section 4.1.

5.2. Topic inference

As stated before, compare $MReq$ to each SBS in the service registry will be a time-consuming task. Based on the SBS clusters, the time efficiency can be largely improved by reducing the SBS search space to several clusters that correspond to the relevant topics of $MReq$. To achieve this, the topic distribution of $MReq$ is inferred using the *Folding-in* (Cassar et al., 2014; Wang et al., 2010) (a technique developed for fitting new documents into a trained topic model) based on the LDA model trained for SBSs. The “fold-in” process begins with randomly assigning topics to the words contained in the preprocessed content of $MReq$, denoted by $W(MReq)$; and then continues resampling topics for each word in $W(MReq)$ (with the topic assignments of words in the SBSs fixed).

Given enough iterations, the topic distribution of $MReq$ is estimated using Eq. (2). The top k^{clu} topics with maximum probabilities can be chosen as the relevant topics of $MReq$. Afterwards, the SBS search space of $MReq$, denoted by $MSP(MReq)$, is restricted to the clusters that correspond to those relevant topics.

5.3. Service recommendation using CF

We adopt the CF technique for service recommendation. The key is to find a set of similar SBSs for $MReq$ from the reduced SBS search space, i.e. $MSP(MReq)$. For this task, we need to evaluate the similarity between $MReq$ and each candidate SBS m in $MSP(MReq)$. As explained in Section 3.2, there are two service recommendation scenarios, i.e. $SRec$-1 and $SRec$-2, with different SBS requirements. Different similarities between $MReq$ and m can be measured according to the type of $MReq$:

Table 3
Statistics of dataset.

# SBSs	# Services	# SBS-service relations	# SBS-category relations
5769	1105	10,950	17,330
# service-category relations	**# SBS-content relations**	**# service-content relations**	**# service-provider relations**
3096	5769	1103	207

1) If *MReq* belongs to the type of SBS requirements in *SRec*-1, one or two similarities can be measured on meta-path P_2 or meta-paths $\{P_1, P_2\}$, respectively. Specifically, the similarity between the content of *MReq* and *m* on P_2 is computed using Eq. (5) based on the learned word vectors. If *MReq* contains some categories, the similarity between *MReq* and *m* on P_1 is computed using Eq. (4).
2) If *MReq* belongs to the type of SBS requirements in *SRec*-2, five or six similarities can be measured on meta-paths $P_2 \sim P_6$ or $P_1 \sim P_6$, respectively. The methods for measuring similarities between *MReq* and *m* on P_1 and P_2 are described above. Similarities between *MReq* and *m* on P_3, P_4, and P_6 are computed using Eq. (4). As for P_5, the content of services included in *MReq* are collected and preprocessed. Afterwards, the similarity between *MReq* and *m* on P_5 is computed using Eq. (6).

After measuring similarities between *MReq* and *m* on a set of meta-paths $\mathcal{P}'$, the overall similarity between *MReq* and *m* is calculated as

$$Sim_{\mathcal{P}'}(MReq, m) = \sum\nolimits_{P_l \in \mathcal{P}'} \theta_{P_l} \cdot Sim_{P_l}(MReq, m), \qquad (11)$$

where $Sim_{P_l}(MReq,\ m)$ is the similarity between *MReq* and *m* on meta-path P_l, and θ_{P_l} is the corresponding weight of P_l referring to Eq. (7), which has been learned using BPR.

Based on the overall similarities, a set of top k^{cf} most similar SBSs is obtained for *MReq*, denoted by $SimM_{\mathcal{P}'}(MReq, k^{cf})$. Then, the rating of *MReq* on each service $s_i \in S$ is estimated as

$$\hat{r}(MReq, s_i) = \sum\nolimits_{m_j \in SimM_{\mathcal{P}'}(MReq, k^{cf})} Sim_{\mathcal{P}'}\left(MReq, m_j\right) \cdot r\left(m_j, s_i\right). \qquad (12)$$

A service recommendation list is finally generated for *MReq* by sorting services *S* according to the estimated ratings.

6. Experiments

This section evaluates our proposed approach by conducting a series of experiments on a real-world dataset collected from PW. We describe in Section 6.1 the statistics of the dataset, the construction of three pairs of experimental training and test sets, as well as the application of our approach to each pair of the training and test sets. In Sections 6.2 and 6.3, we introduce nine competing approaches (including five baseline approaches and four variations of our approach) and two metrics. Evaluation results of three experiments are presented in Section 6.4. Programs were implemented in Java and ran on a PC with 2.40 GHz CPU, 4GB RAM, and Win 7 OS.

6.1. Dataset and preparation

On July 25, 2016, we crawled 5769 SBSs and 1105 services used by those SBSs from PW. For each SBS, we collected its name, content, and categories. For each service, we collected its name, content, categories (including the primary category and secondary categories), and provider. Statistics of the dataset is presented in Table 3. Fig. 6 shows the numbers of SBSs that contain different numbers of component services. As can be seen, nearly 90% of the SBSs have less than four component services.

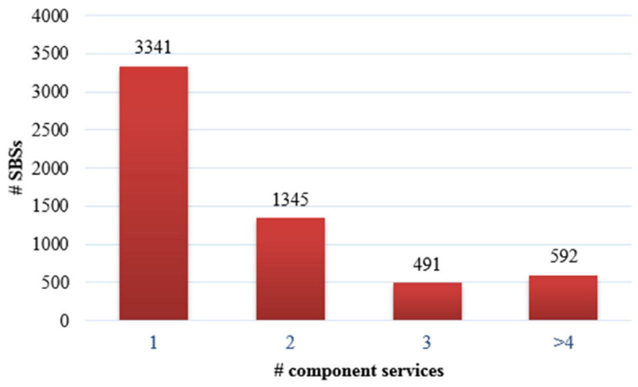


Fig. 6. Numbers of SBSs that contain different numbers of component services.

Recall that we distinguish two service recommendation scenarios *SRec*-1 and *SRec*-2 with different SBS requirements. To evaluate the proposed approach on both scenarios, we constructed three pairs of training and test sets: one pair for *SRec*-1 and two pairs for *SRec*-2, as follows.

1) For *SRec*-1, we built the training set by randomly selecting 80% of the SBSs; and the rest 20% was used as the test set.
2) For *SRec*-2, to evaluate our approach on SBS requirements with different numbers of specified services, we built two pairs of training and test sets by retaining n ($=1$ or 2) component services of each SBS in the test set, respectively. More specifically, for each specific n, we randomly selected 30% of the SBSs that have more than n component services as the test set and the rest SBSs as the training set; then, for each SBS in the test set, n services were randomly selected from the component services of that SBS. The selected services would be included in SBS requirements for service recommendation, while the remaining services were used for evaluation.

Table 4 presents the numbers of SBSs included in three pairs of training and test sets. "*SRec*-2(n)" represents the case of *SRec*-2 with n services selected from each SBS (requirement) in the test set.

We conducted our integrated service recommendation approach (referred to as *iSRec*) on each pair of training and test sets as follows.

6.1.1. Offline data processing for the training set

First, we performed the three preprocessed steps described in Section 4.1 on the content of SBSs and services. Second, the LDA algorithm was applied to the SBSs with a range of T values [10, 100] (step size is 5). For each specific T, we set $\alpha = 50/T$ and $\beta = 0.1$ as in Zhang, Wang, He, Li, and Huang (2018) and Zhou, Lyu, King, and Lou (2015) and ran Gibbs sampling for 1000 iterations. Afterwards, the best T was determined using the Bayesian model selection method proposed in Griffiths and Steyvers (2004). Table 4 presents the best T identified for each of the three training sets. The three

Table 4
Three pairs of training and test sets and the best *T* of LDA for each training set.

Scenario	# SBSs in the training set	# SBSs in the test set	Best *T* of LDA for the training set
SRec-1	4615	1154	55
SRec-2(1)	4555	1214	55
SRec-2(2)	5228	541	60

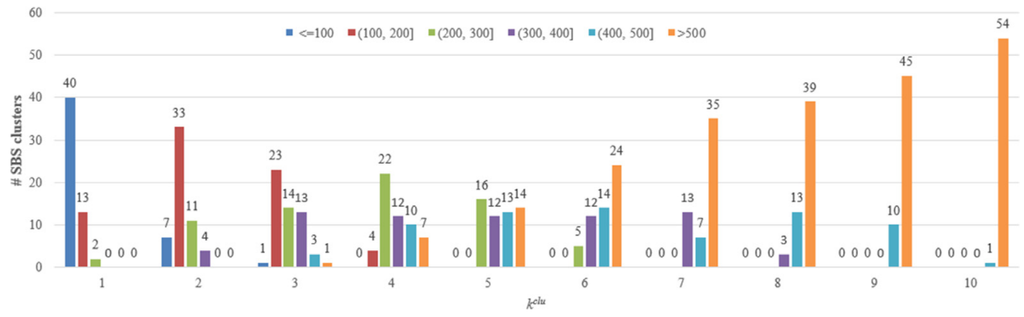


Fig. 7. Size distribution of SBS clusters generated with different k^{clu} (for *SRec*-1).

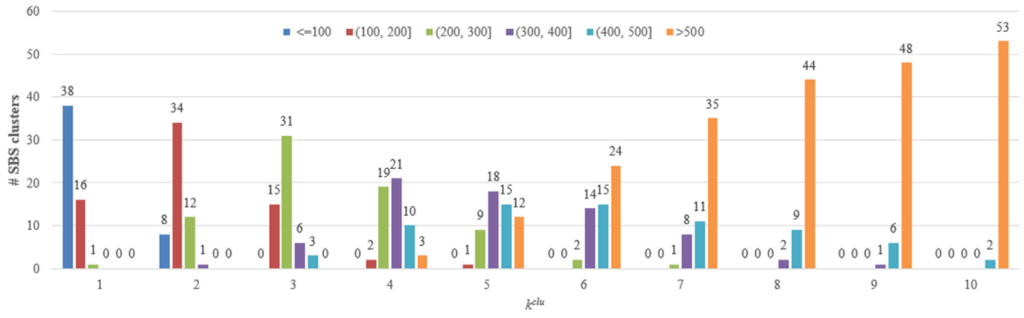


Fig. 8. Size distribution of SBS clusters generated with different k^{clu} (for *SRec*-2(1)).

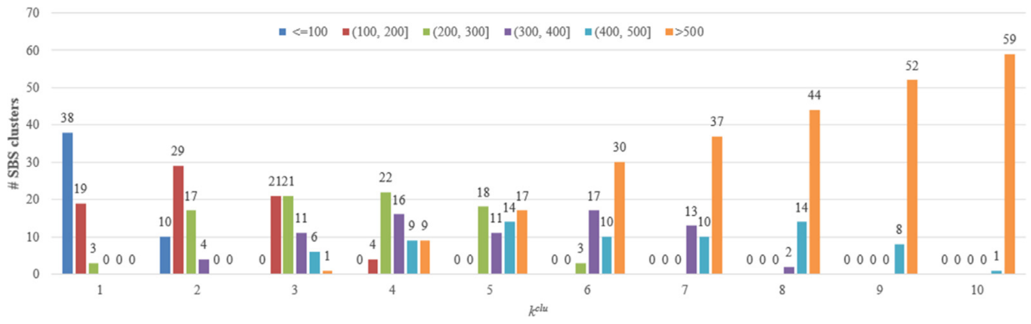


Fig. 9. Size distribution of SBS clusters generated with different k^{clu} (for *SRec*-2(2)).

LDA models trained with the best *T* values were used for experiments. Based on the SBS-topic distributions, we assigned each SBS to the clusters that correspond to its top k^{clu} most relevant topics. To evaluate the impact of k^{clu}, we built different SBS clusters by varying k^{clu} from 1 to 10. Figs. 7–9 depict the distributions of the size of SBS clusters produced with different k^{clu} values for the three training sets, respectively. It can be observed that SBS clusters become larger as k^{clu} increases from 1 to 10. Third, we constructed the HMSN and learned word vectors by applying Word2vec to the content of SBSs and services. The Word2vec algorithm was implemented by using the Skip-Gram network architecture (Mikolov et al., 2013) and hierarchical softmax function. Moreover, we set the window width as 5 and the vector dimensionality as 80. Six types of SBS similarities were then measured using Eqs. (4)–(6). Afterwards, the combinational weights of those SBS similarities were optimized using BPR based on the SBS-service composition relations in the training set. According to Liang et al. (2016), we set the parameter λ in Eq. (10) as 0.001. In addition, we set the parameter k^{cf} in Eq. (8) as 15 (i.e. the top 15 most similar SBSs of an SBS requirement were used for service recommendation) and ran stochastic gradient descent for 1000 iterations.

6.1.2. Online service recommendation for each SBS requirement in the test set

The content of the SBS requirement *MReq* was preprocessed first. Then, the topic distribution of *MReq* was inferred by folding-in it to the LDA model of training SBSs. As described above, we constructed different sets of SBS clusters with different k^{clu} values. Here, based on each set of SBS clusters, we reduced the SBS search space of *MReq* to the clusters that correspond to its top k^{clu} most relevant topics (k^{clu} was set as the same value used for creating the SBS clusters. We measured different similarities between *MReq* and each candidate SBS according to the type of *MReq*, as detailed in Section 5.3; and calculated the overall similarity using Eq. (11). Afterwards, we estimated the rating of *MReq* on each service based on the top 15 SBSs with maximum overall similarities. Finally, a ranking list of services was generated and recommended for *MReq*.

6.2. Competing approaches

For comparison, we further conducted the following service recommendation approaches on the three pairs of training and test sets.

CF (Xu, Cao, Hu, Wang, & Li, 2013): This is a technique widely used in recommendation systems. We implemented two types of CF: *item-based CF* and *user-based CF*.

SVD (Singular Value Decomposition) (Paterek, 2007): This is a classical matrix factorization technique used for recommendation.

BPR-SVD (Rendle et al., 2009): This approach learned a service recommendation model based on SVD using BPR.

PaSRec (Liang et al., 2016): This approach is similar to ours. As discussed in Section 2.3, a major difference is that it adopted Eq. (4) to measure the SBS similarities on meta-paths P_2 and P_5 based on the top three most relevant topics of SBSs and services learned using LDA.

iSRec (TFIDFCS), *iSRec* (TFCS), and *iSRec* (LDACS): These are three variants of our proposed *iSRec* that adopted three different methods for measuring SBS similarities on meta-paths P_2 and P_5, respectively. As for *iSRec* (TFIDFCS), the content of each SBS and the content collected from its component services were represented as vectors using TF-IDF. Similarly, the content of each SBS requirement and the content collected from its retained services were also represented as vectors using TF-IDF. Afterwards, SBS similarities as well as the similarities between SBSs and SBS requirements on P_2 and P_5 were calculated using the Cosine similarity measure. *iSRec* (TFCS) represented the above four types of content as vectors according to their term frequencies and computed Cosine similarities between the vectors. *iSRec* (LDACS) represented SBSs, SBS requirements, and the two types of collected content as vectors based on their topic distributions learned using LDA or inferred using the *Folding-in* technique; and then also computed Cosine similarities between the vectors.

Note that the two types of CF, *SVD*, and *BPR-SVD* were only applied to the two pairs of training and test sets constructed for *SRec*-2 since they cannot handle SBS requirements with no specified service.

6.3. Metrics

Two popular metrics were adopted for evaluating the ranked service list produced for an SBS requirement *MReq*, i.e. MAP@N and NDCG@N (Ganguly et al., 2015).

$$MAP@N = \frac{1}{|CS_{MReq}|} \sum_{i=1}^{N} \left(\frac{n_i}{i} \cdot I(i) \right), \tag{13}$$

where CS_{MReq} is the set of component services of *MReq*; n_i represents the number of the top i services that exist in CS_{MReq}; and $I(i)$ indicates whether the service at the ranking position i is in CS_{MReq}.

$$NDCG@N = \frac{1}{IDCG_N} \sum_{i=1}^{N} \frac{2^{I(i)} - 1}{\log_2(1+i)}, \tag{14}$$

where $IDCG_N$ denotes the ideal DCG score of the top N services that can be achieved for *MReq*.

For a service recommendation approach, its overall MAP@N and NDCG@N on a test set were measured as the average performance achieved for all SBS requirements in the test set.

6.4. Evaluation results and analysis

This subsection reports evaluation results from three aspects: a) impact of parameter k^{clu}; b) comparison of service recommendation approaches; and c) various combinations of SBS similarities.

6.4.1. Impact of k^{clu}

In our proposed *iSRec*, to improve the efficiency, we group SBSs into clusters using topic models and then reduce the SBS search space of SBS requirements based on the clusters. Parameter k^{clu} is used to determine the relevant topics of an SBS or an SBS requirement. In the experiments, we tested a range of k^{clu} values [1, 10].

Tables 5–7 present the performance results (i.e. overall MAP@N and NDCG@N) of *iSRec* on three test sets under different k^{clu} values, respectively. The results in the last row "all" of each table are obtained by setting $k^{clu} = \text{all}$ (that is, each SBS requirement is compared to the SBSs contained in all clusters). We can see that as k^{clu} increases from 1 to 10, the performance on every specific metric (e.g. MAP@5 and MAP@10) experiences a similar process (excluding some fluctuations): it increases relatively fast at the beginning until reaches an optimal point and thereafter it degrades slightly to a stable value. The low performance obtained with a small k^{clu} (e.g. 1) is mainly due to the fact that some similar SBSs of an SBS requirement are not contained in the top k^{clu} most similar clusters, which in turn fails to recommend some relevant services. The degraded performance achieved with a large k^{clu} (e.g. 8) is due to some unsuitable similar SBSs obtained for SBS requirements by considering more candidate SBSs.

For each specific k^{clu}, we recorded the total time of *iSRec* for all SBS requirements in the three test sets as well as the average time for each SBS requirement, as presented in the tables. It can be observed that much more time is required for service recommendation as k^{clu} increases from 1 to 10, due to the increasing number of candidate SBSs. For example, as for the test set of *SRec*-2(1), the average time for each SBS requirement is only 0.156 seconds when $k^{clu} = 1$ (which is good enough), while the average time is 3.576 seconds when $k^{clu} = \text{all}$ (which may probably be unacceptable in practice). Moreover, with the same setting of k^{clu}, the average time on the test set of *SRec*-1 is obviously less than those on the test sets of *SRec*-2(1) and *SRec*-2(2). This is caused by the fact that more types of similarities between the candidate SBSs and an SBS requirement need to be computed in *SRec*-2.

6.4.2. Comparison of service recommendation approaches

To validate our proposed *iSRec*, we conducted five existing service recommendation approaches and three variants of *iSRec* on three constructed datasets. More specifically, four approaches among them, i.e. *user-based CF, item-based CF, SVD*, and *BPR-SVD*, had not been applied to the test set of *SRec*-1, as explained in Section 6.2.

Figs. 10–15 show the performance results of five or nine service recommendation approaches on three test sets of *SRec*-1, *SRec*-2(1), and *SRec*-2(2). Several observations and analysis are given below:

As for *SRec*-1, *iSRec* and the three variants of it, i.e. *iSRec* (TFCS), *iSRec* (TFIDFCS), and *iSRec* (LDACS), are all superior to *PaSRec* in terms of both metrics. This demonstrates that the methods adopted in the four types of *iSRec* for measuring functional similarities between SBSs, namely the ESim (embedding-based similarity measure), TFCS (term frequency vectorization with Cosine similarity measure (CS for short)), TFIDFCS (TF-IDF vectorization with CS), and LDACS (LDA-based vectorization with CS), are better than

Table 5
Performance results of *iSRec* with different k^{clu} (for *SRec*-1).

k^{clu}	MAP@N				NDCG@N				Total time (s)	Average time (s) for each SBS requirement
	N = 5	N = 10	N = 20	N = 30	N = 5	N = 10	N = 20	N = 30		
1	0.5304	0.5427	0.5472	0.5481	0.585	0.6035	0.6138	0.6166	109.448	0.095
2	0.5775	0.5901	0.5962	0.597	0.6322	0.6501	0.6639	0.6668	114.53	0.099
3	0.5913	0.6033	0.609	0.61	0.6477	0.6634	0.6762	0.6795	137.577	0.119
4	0.5957	0.6081	0.6128	0.614	0.6519	0.6685	0.6792	0.6833	157.329	0.136
5	0.6024	0.6138	0.6186	0.6198	0.6574	0.6722	0.6836	0.6874	182.943	0.159
6	0.6028	0.6142	0.6193	0.6205	0.6586	0.6736	0.6855	0.6893	215.903	0.187
7	0.6029	0.6146	0.6195	0.6207	0.658	0.6739	0.6856	0.6896	260.498	0.226
8	0.6025	0.6143	0.6191	0.6204	0.6577	0.674	0.6852	0.6894	294.735	0.255
9	0.6029	0.6147	0.6195	0.6207	0.6579	0.6743	0.6856	0.6894	348.907	0.302
10	0.6024	0.6143	0.6192	0.6204	0.6575	0.6743	0.6857	0.6895	395.478	0.343
all	0.6023	0.6144	0.6191	0.6203	0.6578	0.6746	0.6856	0.6894	1508.203	1.307

Table 6
Performance results of *iSRec* with different k^{clu} (for *SRec*-2(1)).

k^{clu}	MAP@N				NDCG@N				Total time (s)	Average time (s) for each SBS requirement
	N = 5	N = 10	N = 20	N = 30	N = 5	N = 10	N = 20	N = 30		
1	0.3617	0.3752	0.3817	0.3832	0.4261	0.4479	0.4646	0.4697	189.765	0.156
2	0.3882	0.4069	0.4151	0.4174	0.4569	0.4875	0.5081	0.515	240.768	0.198
3	0.3841	0.403	0.4116	0.4136	0.4583	0.4887	0.51	0.5161	279.214	0.23
4	0.3985	0.4161	0.4248	0.4271	0.4695	0.4987	0.5203	0.528	304.945	0.251
5	0.4034	0.4214	0.4294	0.4319	0.4742	0.5045	0.5245	0.5331	457.866	0.377
6	0.4055	0.4238	0.4317	0.4342	0.4772	0.5067	0.527	0.5349	512.812	0.422
7	0.397	0.4138	0.4227	0.4251	0.4701	0.4974	0.5204	0.5282	572.745	0.472
8	0.3974	0.4143	0.4232	0.4255	0.4712	0.4988	0.5215	0.529	690.22	0.569
9	0.3944	0.4121	0.4209	0.4234	0.4682	0.4968	0.5197	0.5277	830.835	0.684
10	0.3927	0.4101	0.4188	0.4213	0.4668	0.495	0.5176	0.5258	1019.272	0.84
all	0.3928	0.4097	0.4188	0.4214	0.4664	0.494	0.5174	0.5261	4341.415	3.576

Table 7
Performance results of *iSRec* with different k^{clu} (for *SRec*-2(2)).

k^{clu}	MAP@N				NDCG@N				Total time (s)	Average time (s) for each SBS requirement
	N = 5	N = 10	N = 20	N = 30	N = 5	N = 10	N = 20	N = 30		
1	0.2849	0.3	0.3093	0.3111	0.3642	0.384	0.4066	0.4115	88.1	0.163
2	0.303	0.3264	0.3378	0.3412	0.3889	0.4235	0.4496	0.4591	101.589	0.188
3	0.3066	0.327	0.3381	0.3417	0.3949	0.4253	0.4498	0.4607	120.139	0.222
4	0.2934	0.3165	0.3285	0.3317	0.3831	0.4162	0.4448	0.4547	148.838	0.275
5	0.2902	0.3132	0.3262	0.3288	0.3813	0.4132	0.4445	0.4524	180.828	0.334
6	0.2995	0.3234	0.3359	0.3389	0.3865	0.4205	0.4512	0.4599	234.007	0.433
7	0.301	0.3232	0.3356	0.3387	0.3896	0.4208	0.4506	0.4604	281.383	0.52
8	0.299	0.3222	0.333	0.3368	0.3867	0.4216	0.447	0.4592	358.669	0.663
9	0.2971	0.3187	0.33	0.3344	0.3828	0.4153	0.4422	0.4557	423.596	0.783
10	0.3003	0.3211	0.3318	0.3358	0.3868	0.4181	0.4439	0.4565	481.409	0.89
all	0.2958	0.3164	0.3275	0.3312	0.3824	0.4124	0.4388	0.4505	2426.219	4.485

the PSim (path-based similarity measure) adopted in *PaSRec*. Moreover, in most cases, *iSRec* > *iSRec* (TFIDFCS) > *iSRec* (TFCS) > *iSRec* (LDACS), which indicates that ESim is more effective than the other three methods. Based on our analysis, the best performance achieve by *iSRec* is mainly because of the high-quality word vectors learned using Word2vec, which contribute to better functional similarities. In addition, *iSRec* (TFIDFCS) is better than *iSRec* (TFCS), as TF-IDF can more accurately measure the weights of words in a piece of content by considering both the global feature (i.e. IDF) and the local feature (i.e. TF) of a word. The relatively low performances achieved by the two LDA-based approaches, i.e. *iSRec* (LDACS) and *PaSRec*, can be explained by the fact that the learned topics are generally very coarse, which cannot help measure functional similarities accurately. Moreover, since some semantics of SBSs and services may probably be discarded by considering only the top three most relevant topics of them, *PaSRec* is worse than *iSRec* (LDACS).

As for *SRec*-2(1) and *SRec*-2(2), *iSRec* obviously outperforms all the other approaches. The performances of *iSRec, iSRec* (TFCS), and *iSRec* (TFIDFCS) are better than those of *PaSRec*. These results also demonstrate that the three methods, i.e. ESim, TFCS, and TFIDFCS, can more accurately measure functional similarities between SBSs than PSim; and ESim is the best. However, unlike for *SRec*-1, the performances of the three variants of *iSRec* are very close in *SRec*-2(1), which may be caused by the fact that as more types of similarities on meta-paths $P_3 \sim P_6$ are aggregated, the differences among the functional similarities calculated by TFCS, TFIDFCS, and LDACS are obscured. Moreover, the five HIN-based approaches that exploit heterogeneous information of SBSs and services are much better than the four classic approaches that purely depend on the SBS-service composition relations.

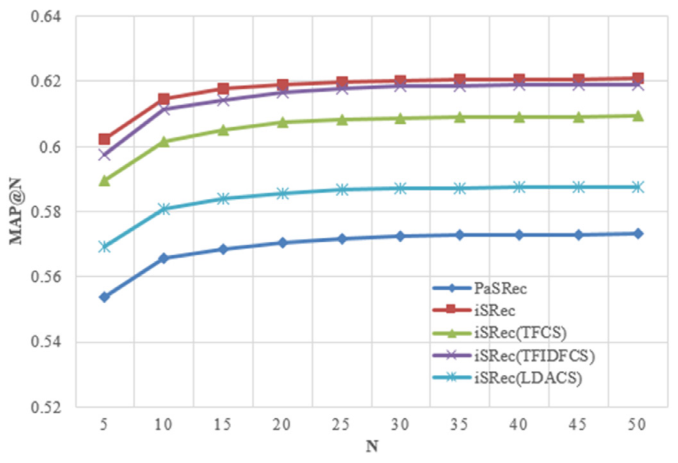


Fig. 10. MAP@N of five service recommendation approaches for *SRec*-1.

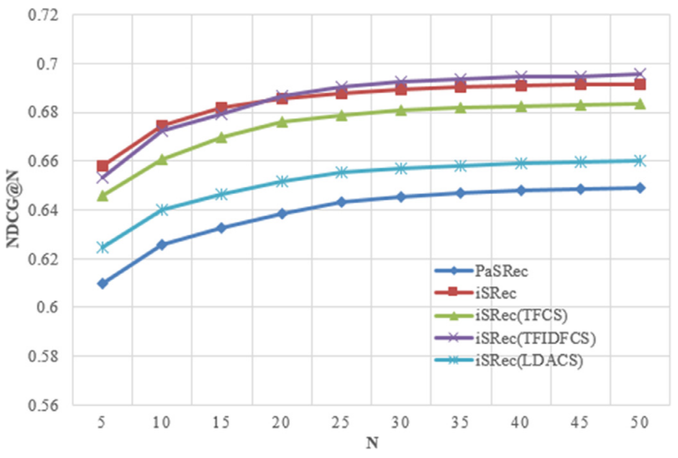


Fig. 11. NDCG@N of five service recommendation approaches for *SRec*-1.

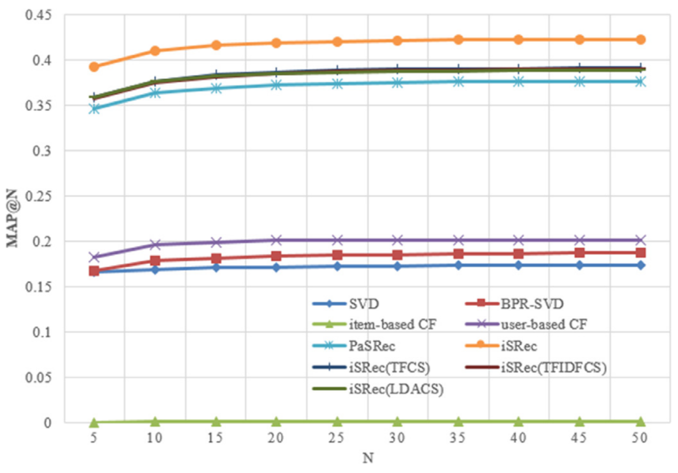


Fig. 12. MAP@N of nine service recommendation approaches for *SRec*-2(1).

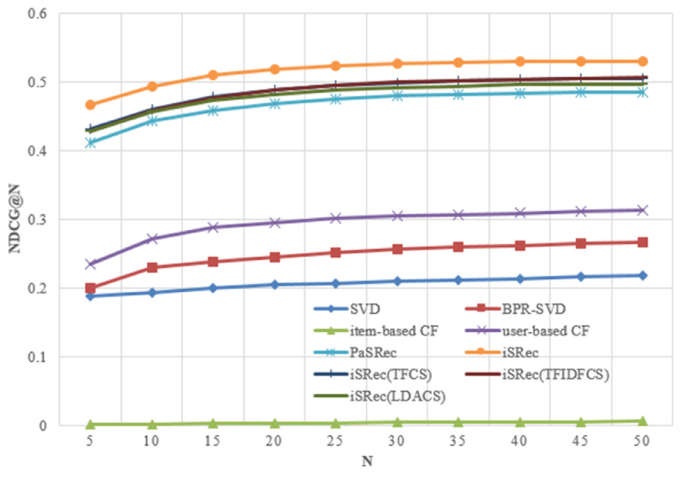


Fig. 13. NDCG@N of nine service recommendation approaches for *SRec*-2(1).

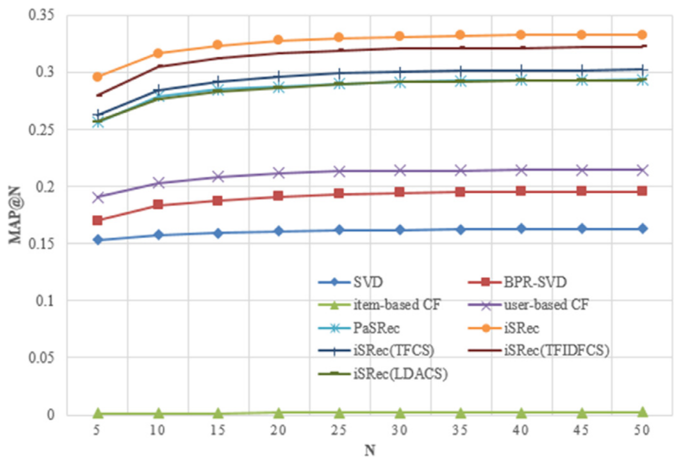


Fig. 14. MAP@N of nine service recommendation approaches for *SRec*-2(2).

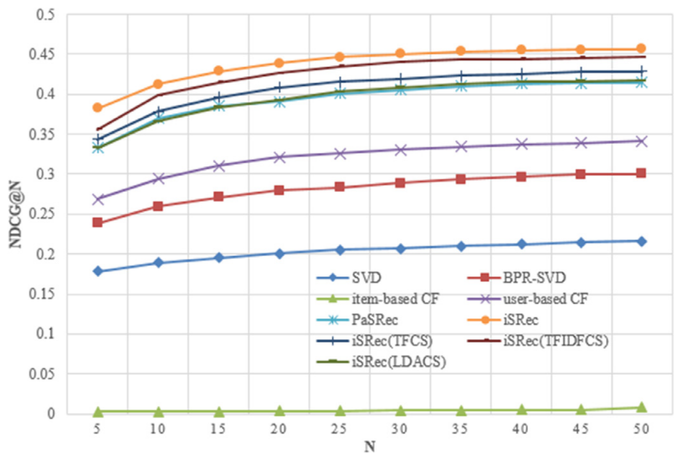


Fig. 15. NDCG@N of nine service recommendation approaches for *SRec*-2(2).

6.4.3. Various combinations of SBS similarities

In *iSRec*, we integrate six SBS similarities on the six meta-paths in Table 2, under the assumption that better service recommendations might be produced by integrating diverse semantics of those meta-paths. Although the effectiveness of *iSRec* has been validated by the performance results shown in Figs. 10–15, we still conducted another experiment to verify the assumption by testing various combinations of SBS similarities for *SRec*-1 and *SRec*-2, which are presented in Tables 8 and 9. More specifically, five kinds of functional similarities on meta-paths P_2 and P_5 measured using PSim, ESim, TFCS, TFIDFCS, and LDACS are considered, as well as several combinations of them. The similarities on P_2 and P_5 measured using a specific method *sm* are denoted by "$P_2 + sm$" and "$P_5 + sm$," respectively.

Tables 10–12 present the performance results of the top ten best compositions for *SRec*-1, *SRec*-2(1), and *SRec*-2(2), respectively. The values in bold are the best performance results achieved on

Table 8
Compositions of SBS similarities for *SRec*-1.

ID	Composition	ID	Composition
C_1	$\{P_1\}$	C_9	$\{P_1, P_2+\text{LDACS}\}$
C_2	$\{P_2+\text{PSim}\}$	C_{10}	$\{P_1, P_2+\text{TFIDFCS}\}$
C_3	$\{P_2+\text{TFCS}\}$	C_{11}	$\{P_1, P_2+\text{ESim}\}$
C_4	$\{P_2+\text{LDACS}\}$	C_{12}	$\{P_1, P_2+\text{PSim}, P_2+\text{ESim}\}$
C_5	$\{P_2+\text{TFIDFCS}\}$	C_{13}	$\{P_1, P_2+\text{TFCS}, P_2+\text{ESim}\}$
C_6	$\{P_2+\text{ESim}\}$	C_{14}	$\{P_1, P_2+\text{LDACS}, P_2+\text{ESim}\}$
C_7	$\{P_1, P_2+\text{PSim}\}$	C_{15}	$\{P_1, P_2+\text{TFIDFCS}, P_2+\text{ESim}\}$
C_8	$\{P_1, P_2+\text{TFCS}\}$	C_{16}	$\{P_1, P_2+\text{PSim}, P_2+\text{TFCS}, P_2+\text{LDACS}, P_2+\text{TFIDFCS}, P_2+\text{ESim}\}$

Table 9
Compositions of SBS similarities for *SRec*-2.

ID	Composition	ID	Composition
C_1	$\{P_1\}$	C_{23}	$\{P_1, P_2+\text{TFIDFCS}, P_2+\text{ESim}\}$
C_2	$\{P_2+\text{PSim}\}$	C_{24}	$\{P_1, P_2+\text{PSim}, P_2+\text{TFCS}, P_2+\text{LDACS}, P_2+\text{TFIDFCS}, P_2+\text{ESim}\}$
C_3	$\{P_2+\text{TFCS}\}$	C_{25}	$\{P_1, P_2+\text{PSim}, P_3, P_4\}$
C_4	$\{P_2+\text{LDACS}\}$	C_{26}	$\{P_1, P_2+\text{PSim}, P_3, P_4, P_5+\text{PSim}\}$
C_5	$\{P_2+\text{TFIDFCS}\}$	C_{27}	$\{P_1, P_2+\text{PSim}, P_3, P_4, P_5+\text{PSim}, P_6\}$
C_6	$\{P_2+\text{ESim}\}$	C_{28}	$\{P_1, P_2+\text{TFCS}, P_3, P_4\}$
C_7	$\{P_3\}$	C_{29}	$\{P_1, P_2+\text{TFCS}, P_3, P_4, P_5+\text{TFCS}\}$
C_8	$\{P_4\}$	C_{30}	$\{P_1, P_2+\text{TFCS}, P_3, P_4, P_5+\text{TFCS}, P_6\}$
C_9	$\{P_5+\text{PSim}\}$	C_{31}	$\{P_1, P_2+\text{LDACS}, P_3, P_4\}$
C_{10}	$\{P_5+\text{TFCS}\}$	C_{32}	$\{P_1, P_2+\text{LDACS}, P_3, P_4, P_5+\text{LDACS}\}$
C_{11}	$\{P_5+\text{LDACS}\}$	C_{33}	$\{P_1, P_2+\text{LDACS}, P_3, P_4, P_5+\text{LDACS}, P_6\}$
C_{12}	$\{P_5+\text{TFIDFCS}\}$	C_{34}	$\{P_1, P_2+\text{TFIDFCS}, P_3, P_4\}$
C_{13}	$\{P_5+\text{ESim}\}$	C_{35}	$\{P_1, P_2+\text{TFIDFCS}, P_3, P_4, P_5+\text{TFIDFCS}\}$
C_{14}	$\{P_6\}$	C_{36}	$\{P_1, P_2+\text{TFIDFCS}, P_3, P_4, P_5+\text{TFIDFCS}, P_6\}$
C_{15}	$\{P_1, P_2+\text{PSim}\}$	C_{37}	$\{P_1, P_2+\text{ESim}, P_3, P_4\}$
C_{16}	$\{P_1, P_2+\text{TFCS}\}$	C_{38}	$\{P_1, P_2+\text{ESim}, P_3, P_4, P_5+\text{ESim}\}$
C_{17}	$\{P_1, P_2+\text{LDACS}\}$	C_{39}	$\{P_1, P_2+\text{ESim}, P_3, P_4, P_5+\text{ESim}, P_6\}$
C_{18}	$\{P_1, P_2+\text{TFIDFCS}\}$	C_{40}	$\{P_1, P_2+\text{PSim}, P_2+\text{ESim}, P_3, P_4, P_5+\text{PSim}, P_5+\text{ESim}, P_6\}$
C_{19}	$\{P_1, P_2+\text{ESim}\}$	C_{41}	$\{P_1, P_2+\text{TFCS}, P_2+\text{ESim}, P_3, P_4, P_5+\text{TFCS}, P_5+\text{ESim}, P_6\}$
C_{20}	$\{P_1, P_2+\text{PSim}, P_2+\text{ESim}\}$	C_{42}	$\{P_1, P_2+\text{LDACS}, P_2+\text{ESim}, P_3, P_4, P_5+\text{LDACS}, P_5+\text{ESim}, P_6\}$
C_{21}	$\{P_1, P_2+\text{TFCS}, P_2+\text{ESim}\}$	C_{43}	$\{P_1, P_2+\text{TFIDFCS}, P_2+\text{ESim}, P_3, P_4, P_5+\text{TFIDFCS}, P_5+\text{ESim}, P_6\}$
C_{22}	$\{P_1, P_2+\text{LDACS}, P_2+\text{ESim}\}$	C_{44}	$\{P_1, P_2+\text{PSim}, P_2+\text{TFCS}, P_2+\text{LDACS}, P_2+\text{TFIDFCS}, P_2+\text{ESim}, P_3, P_4, P_5+\text{PSim}, P_5+\text{TFCS}, P_5+\text{LDACS}, P_5+\text{TFIDFCS}, P_5+\text{ESim}, P_6\}$

Table 10
MAP@N and NDCG@N of the top ten best compositions for *SRec*-1.

Rank	MAP@5		MAP@10		MAP@20		MAP@30		NDCG@5		NDCG@10		NDCG@20		NDCG@30	
	C_i	value	C_i	value	C_i	value	C_i	value	C_i	value	C_i	value	C_i	value	C_i	value
1	C_{15}	**0.6138**	C_{15}	**0.6264**	C_{15}	**0.6316**	C_{15}	**0.6335**	C_{15}	**0.6698**	C_{15}	**0.6872**	C_{15}	**0.7**	C_{15}	**0.7064**
2	C_{13}	0.6092	C_{13}	0.6227	C_{13}	0.6277	C_{13}	0.6295	C_{13}	0.6642	C_{13}	0.6839	C_{13}	0.6959	C_{13}	0.7009
3	C_6	0.6044	C_{16}	0.6163	C_{16}	0.6218	C_{16}	0.6238	C_6	0.6606	C_{16}	0.679	C_{16}	0.6922	C_{16}	0.6989
4	C_{11}	0.6023	C_6	0.6147	C_{11}	0.6191	C_{11}	0.6203	C_{11}	0.6578	C_{11}	0.6746	C_{10}	0.6866	C_{10}	0.6926
5	C_{16}	0.6002	C_{11}	0.6144	C_6	0.6188	C_6	0.6194	C_{16}	0.6572	C_6	0.6746	C_{11}	0.6856	C_{14}	0.6916
6	C_{14}	0.5986	C_{14}	0.6121	C_{14}	0.6174	C_{14}	0.6192	C_{14}	0.6535	C_{10}	0.6725	C_{14}	0.6854	C_{11}	0.6894
7	C_{10}	0.5976	C_{10}	0.6115	C_{10}	0.6167	C_{10}	0.6184	C_{10}	0.6532	C_{14}	0.6721	C_6	0.683	C_6	0.685
8	C_{12}	0.5908	C_{12}	0.6035	C_{12}	0.6089	C_{12}	0.6108	C_{12}	0.6472	C_{12}	0.6645	C_{12}	0.6777	C_{12}	0.6845
9	C_8	0.5897	C_8	0.6014	C_8	0.6074	C_8	0.6087	C_8	0.6457	C_8	0.6608	C_8	0.676	C_8	0.6809
10	C_3	0.5838	C_3	0.5962	C_3	0.6003	C_3	0.601	C_3	0.639	C_3	0.6565	C_3	0.6666	C_3	0.6688

all specific metrics. It can be seen that as for *SRec*-1, C_{15}, i.e. $\{P_1, P_2+\text{TFIDFCS}, P_2+\text{ESim}\}$, achieves the best performances on all specific metrics, which indicates that: a) better service recommendations can be produced by exploiting both functional description and categories of SBS requirements; b) more accurate functional similarities can be measured by combining ESim with TFIDFCS (Note that TFCS can also be combined with ESim but it is less effective than TFIDFCS, according to the performance order $C_{15} > C_{13} > C_6$); and c) it is not suitable to combine all the five functional similarities, as the performance of C_{16} is worse than those of C_{15} and C_{13}, and even worse than C_6 in some cases.

As for *SRec*-2(1) and *SRec*-2(2), C_{23}, i.e. $\{P_1, P_2+\text{TFIDFCS}, P_2+\text{ESim}\}$ (which is the same composition as C_{15} in *SRec*-1), achieves the optimal performances on almost all specific metrics, except NDCG@30. We can also see that only two compositions, i.e. C_{43} and C_{44}, contain the similarities on meta-paths $\{P_3, P_4, P_5, P_6\}$ (where SBSs are connected via services) appear in the bottom rows of Tables 11 and 12. Apart from the three indications explained above for *SRec*-1, these results further indicate that the similarities

Table 11
MAP@N and NDCG@N of the top ten best compositions for *SRec*-2(1).

Rank	MAP@5		MAP@10		MAP@20		MAP@30		NDCG@5		NDCG@10		NDCG@20		NDCG@30	
	C_i	value	C_i	value	C_i	value	C_i	value	C_i	value	C_i	value	C_i	value	C_i	value
1	C_{23}	**0.5038**	C_{23}	**0.5185**	C_{23}	**0.5237**	C_{23}	**0.5255**	C_{23}	**0.5712**	C_{23}	**0.5903**	C_{23}	**0.603**	C_{24}	**0.6096**
2	C_{21}	0.4979	C_{24}	0.5127	C_{24}	0.5195	C_{24}	0.5216	C_{24}	0.5657	C_{24}	0.5855	C_{24}	0.6023	C_{23}	0.6091
3	C_{24}	0.4972	C_{21}	0.5113	C_{21}	0.5162	C_{21}	0.518	C_{21}	0.5656	C_{21}	0.5827	C_{21}	0.5944	C_{21}	0.6003
4	C_6	0.4902	C_{22}	0.5049	C_{22}	0.5107	C_{22}	0.5124	C_{22}	0.556	C_{22}	0.5773	C_{22}	0.5917	C_{22}	0.5973
5	C_{19}	0.4893	C_{19}	0.5025	C_{19}	0.507	C_{19}	0.5083	C_{19}	0.556	C_{19}	0.5731	C_{20}	0.584	C_{20}	0.5884
6	C_{22}	0.4892	C_6	0.5018	C_6	0.5055	C_6	0.5059	C_6	0.5553	C_{20}	0.5688	C_{19}	0.5836	C_{19}	0.5873
7	C_{20}	0.4837	C_{20}	0.497	C_{20}	0.5034	C_{20}	0.5048	C_{20}	0.5511	C_6	0.5688	C_6	0.5771	C_6	0.5786
8	C_{18}	0.4684	C_{18}	0.4838	C_{18}	0.4897	C_{18}	0.4913	C_{18}	0.5348	C_{18}	0.5566	C_{18}	0.5718	C_{18}	0.5771
9	C_{16}	0.4659	C_{16}	0.4794	C_{16}	0.4855	C_{16}	0.487	C_{16}	0.5315	C_{16}	0.5518	C_{16}	0.5658	C_{16}	0.5709
10	C_3	0.4573	C_3	0.469	C_3	0.4731	C_3	0.4737	C_3	0.5216	C_3	0.5367	C_{44}	0.551	C_{44}	0.5613

Table 12
MAP@N and NDCG@N of the top ten best compositions for *SRec*-2(2).

Rank	MAP@5		MAP@10		MAP@20		MAP@30		NDCG@5		NDCG@10		NDCG@20		NDCG@30	
	C_i	value	C_i	value	C_i	value	C_i	value	C_i	value	C_i	value	C_i	value	C_i	value
1	C_{23}	**0.3829**	C_{23}	**0.4026**	C_{23}	**0.4121**	C_{23}	**0.4144**	C_{23}	**0.4661**	C_{23}	**0.4897**	C_{23}	**0.511**	C_{23}	**0.5179**
2	C_{21}	0.3802	C_{21}	0.3982	C_{21}	0.407	C_{21}	0.4092	C_{21}	0.4619	C_{24}	0.4834	C_{24}	0.5038	C_{24}	0.5138
3	C_{22}	0.3786	C_{24}	0.3947	C_{24}	0.4035	C_{24}	0.4063	C_{22}	0.4612	C_{21}	0.482	C_{21}	0.5017	C_{21}	0.5092
4	C_{19}	0.3753	C_{22}	0.3946	C_{22}	0.4034	C_{22}	0.4063	C_{24}	0.4572	C_{22}	0.478	C_{22}	0.4978	C_{22}	0.5072
5	C_{24}	0.373	C_{19}	0.3913	C_{19}	0.4002	C_{19}	0.4019	C_{19}	0.4552	C_{19}	0.4731	C_{19}	0.4923	C_{20}	0.5027
6	C_{20}	0.3702	C_{20}	0.388	C_{20}	0.3964	C_{20}	0.3998	C_{20}	0.4524	C_{20}	0.472	C_{20}	0.4912	C_{19}	0.4973
7	C_6	0.3675	C_6	0.3844	C_6	0.3898	C_6	0.3902	C_6	0.4472	C_6	0.4678	C_6	0.4787	C_{18}	0.485
8	C_{18}	0.3613	C_{18}	0.3766	C_{18}	0.3854	C_{18}	0.3875	C_{18}	0.4401	C_{18}	0.4578	C_{18}	0.4778	C_6	0.4795
9	C_{16}	0.3539	C_{16}	0.367	C_{16}	0.3766	C_{16}	0.3787	C_{16}	0.4315	C_{16}	0.4455	C_{16}	0.4664	C_{16}	0.4734
10	C_{17}	0.333	C_{17}	0.3467	C_{17}	0.3548	C_{17}	0.3571	C_{17}	0.4119	C_{43}	0.4311	C_{44}	0.4552	C_{43}	0.4695

on those meta-paths cannot be integrated to improve the performance of service recommendation.

In summary, as suggested by the results in the three tables, {P_1, P_2 + TFIDFCS, P_2 + ESim} is the optimal combination for both of the two service recommendation scenarios *SRec*-1 and *SRec*-2.

7. Conclusions and future work

This research proposes a new integrated HIN-based service recommendation approach (named *iSRec*) for SBS development, which improves the state-of-the-art in two main aspects. First, we devise an effective method for measuring functional similarities between SBSs and user requirements based on the semantic word vectors learned using Word2vec. Second, we identify two service recommendation scenarios in practice, explain how they can be handled by our approach, and finally determine the optimal combination of SBS similarities on six meta-paths for each of them by conducting extensive experiments on a real-world dataset crawled from the ProgrammableWeb.

Although experiment results have demonstrated the superiority of our *iSRec* compared with several popular and recent approaches, as well as three variations of *iSRec*, there are some imitations related to *iSRec*. One limitation is about the experimental dataset. To make the evaluation results credible, we use the dataset from the most popular and publicly accessible service registry, PW, however it still needs to evaluate our approach on more datasets to get more reliable results. Moreover, in this work, the recommended services are only ranked by the scores predicted based on the top k^{cf} most similar SBSs of a given requirement and the historical SBS-service compositions. We observe that there are many functional similar services, e.g. "Google Maps" and "Bing Maps", contained in the top of the recommendation lists, which may probably be redundant and not desired by the user. This limitation could be alleviated by utilizing other types of information in service registries, e.g. the collaboration relations between services and the *following/followed-by* relations between services and users. These limitations will be investigated in our future work.

Author contributions

Fang Xie, Neng Zhang, and Jian Wang designed the proposed approach; Fang Xie, Ruibing Xiong and Neng Zhang carried out experiments; Neng Zhang, Fang Xie, and Jian Wang analyzed experiment results. All the authors participated in preparing the manuscript. Jian Wang is the corresponding author.

Competing interests statement

The authors declare that they have no competing interests.

Acknowledgments

This research was supported by the National Key Research and Development Program of China (No. 2017YFB1400602), National Science Foundation of China (Nos. 61832014, 61672387, and 61572371), and Natural Science Foundation of Hubei Province of China (No. 2018CFB511).

References

Al-hassan, M., Lu, H., & Lu, J. (2015). A semantic enhanced hybrid recommendation approach: A case study of e-Government tourism service recommendation system. *Decision Support Systems, 72*, 97–109. doi:10.1016/j.dss.2015.02.001.

Aznag, M., Quafafou, M., & Jarir, Z. (2014). Leveraging formal concept analysis with topic correlation for service clustering and discovery. (pp. 153–160). doi:10.1109/ICWS.2014.33.

Bano, M., Zowghi, D., Ikram, N., & Niazi, M. (2014). What makes service oriented requirements engineering challenging? A qualitative study. *IET Software, 8*(4), 154–160. doi:10.1049/iet-sen.2013.0131.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research*, (3), 993–1022.

Cassar, G., Barnaghi, P. M., & Moessner, K. (2014). Probabilistic matchmaking methods for automated service discovery. IEEE Transactions on Services Computing, 7(4), 654–666. doi:10.1109/TSC.2013.28.

Chen, F., Lu, C., Wu, H., & Li, M. (2017a). A semantic similarity measure integrating multiple conceptual relationships for web service discovery. *Expert Systems with Applications, 67*, 19–31. doi:10.1016/j.eswa.2016.09.028.

Chen, L., Wang, Y., Yu, Q., Zheng, Z., & Wu, J. (2013). *WT-LDA: User Tagging Augmented LDA for Web Service Clustering*.

Chen, L., Wu, J., Jian, H., Deng, H., & Wu, Z. (2014). Instant recommendation for web services composition. IEEE Transactions on Services Computing, *7*(4), 586–598. doi:10.1109/TSC.2013.32.

Chen, W., Paik, I., & Yen, N. Y. (2017b). Discovering internal social relationship for influence-aware service recommendation. Multimedia Tools and Applications, , *76*(18), 18193–18220. doi:10.1007/s11042-016-3437-8.

Crasso, M., Zunino, A., & Campo, M. (2011). A survey of approaches to web service discovery in service-oriented architectures. Journal of Database Management, *22*(1), 102–132. doi:10.4018/jdm.2011010105.

Ganguly, D., Roy, D., Mitra, M., & Jones, G. J. F. (2015). Word embedding based generalized language model for information retrieval. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 795–798). doi:10.1145/2766462.2767780.

Gao, W., Chen, L., Wu, J., & Bouguettaya, A. (2016). Joint modeling users, services, mashups, and topics for service recommendation. In *Proceedings of the 2016 IEEE International Conference on Web Services (ICWS)* (pp. 260–267).

Griffiths, T. L., & Steyvers, M. (2004). Finding scientific topics. *Proceedings of the National Academy of Sciences, 101*(suppl 1), 5228–5235. doi:10.1073/pnas.0307752101.

Hao, Y., Fan, Y., Tan, W., & Zhang, J. (2017). Service recommendation based on targeted reconstruction of service descriptions. In *Proceedings of the 2017 IEEE International Conference on Web Services* (pp. 285–292). http://doi.ieeecomputersociety.org/10.1109/ICWS.2017.44. doi:10.1109/ICWS.2017.44.

He, Q., Zhou, R., Zhang, X., Wang, Y., Ye, D., & Chen, F. et al. E(2017a). Efficient keyword search for building service-based systems based on dynamic programming. (pp. 462–470). doi: 10.1007/978-3-319-69035-3_33.

He, Q., Zhou, R., Zhang, X., Wang, Y., Ye, D., Chen, F., et al. (2017b). Keyword search for building service-based systems. IEEE Transactions on Software Engineering, *43*(7), 658–674. doi:10.1109/TSE.2016.2624293.

Hofmann, T. (1999). Probabilistic latent semantic analysis. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence* (pp. 289–296).

Hu, Y., Peng, Q., Hu, X., & Yang, R. (2015). Time aware and data sparsity tolerant web service recommendation based on improved collaborative filtering. IEEE Transactions on Services Computing, *8*(5), 782–794. doi:10.1109/TSC.2014.2381611.

Huang, K., Fan, Y., & Tan, W. (2014). Recommendation in an evolving service ecosystem based on network prediction. IEEE Transactions on Automation Science and Engineering, *11*(3), 906–920. doi:10.1109/TASE.2013.2297026.

Kenter, T., & Rijke, M. d. (2015). Short text similarity with word embeddings. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management* (pp. 1411–1420).

Lampe, U., Schulte, S., Siebenhaar, M., Schuller, D., & Steinmetz, R. (2010). Adaptive matchmaking for RESTful services based on hRESTS and MicroWSMO. (pp. 10–17). doi: 10.1145/1883133.1883136.

Liang, T., Chen, L., Wu, J., Dong, H., & Bouguettaya, A. (2016). Meta-path based service recommendation in heterogeneous information networks. In *Proceedings of the 14th International Conference on Service-Oriented Computing* (pp. 371–386). doi:10.1007/978-3-319-46295-0_23.

Liu, J., Tang, M., Zheng, Z., Liu, X., & Lyu, S. (2016a). Location-aware and personalized collaborative filtering for web service recommendation. *IEEE Transactions on Services Computing, 9*(5), 686–699. doi:10.1109/TSC.2015.2433251.

Liu, X., Agarwal, S., Ding, C., & Yu, Q. (2016b). An LDA-SVM active learning framework for web service classification. *Paper presented at the 2016 IEEE International Conference on Web Services (ICWS)*.

Meng, S., Dou, W., Zhang, X., & Chen, J. (2014). KASR: A keyword-aware service recommendation method on MapReduce for big data applications. *IEEE Transactions on Parallel and Distributed Systems, 25*(12), 3221–3231. doi:10.1109/TPDS.2013.2297117.

Mikolov, T., Chen, K., Corrado, G., & Dean, J.. *Efficient estimation of word representations in vector space ArXiv e-prints, 1301*. Accessed January 1, 2013 http://adsabs.harvard.edu/abs/2013arXiv1301.3781M.

Pan, W., Zhong, H., Xu, C., & Ming, Z. (2015). Adaptive Bayesian personalized ranking for heterogeneous implicit feedbacks. *Knowledge-Based Systems, 73*(1), 173–180.

Paterek, A. (2007). Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of the KDD Cup & Workshop* (pp. 39–42).

Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1532–1543).

Rendle, S., Freudenthaler, C., Gantner, Z., & Schmidt-Thieme, L. (2009). BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence* (pp. 452–461).

Schulte, S., Lampe, U., Eckert, J., & Steinmetz, R. (2010). LOG4SWS.KOM: self-adapting semantic web service discovery for SAWSDL. *Paper presented at the 2010 6th World Congress on Services*.

Shi, M., Liu, J., Zhou, D., Tang, M., & Cao, B. (2017). WE-LDA: A word embeddings augmented LDA model for web services clustering. In *Proceedings of the International Conference on Web Services* (pp. 9–16). doi:10.1109/ICWS.2017.9.

Song, W., & Jacobsen, H. (2018). Static and dynamic process change. *IEEE Transactions on Services Computing, 11*(1), 215–231. doi:10.1109/TSC.2016.2536025.

Spanoudakis, G., & Zisman, A. (2010). Discovering services during service-based system design using UML. *IEEE Transactions on Software Engineering, 36*(3), 371–389. doi:10.1109/TSE.2009.88.

Stavropoulos, T. G., Andreadis, S., Bassiliades, N., Vrakas, D., & Vlahavas, I. (2016). The tomaco hybrid matching framework for SAWSDL semantic web services. *IEEE Transactions on Services Computing, 9*(6), 954–967. doi:10.1109/TSC.2015.2430328.

Subbulakshmi, S., Ramar, K., Shaji, A., & Prakash, P. (2018). Web Service Recommendation Based on Semantic Analysis of Web Service Specification and Enhanced Collaborative Filtering. In *Proceedings of the International Symposium on Intelligent Systems Technologies and Applications* (pp. 54–65).

Sun, Y., & Han, J. (2012). Mining heterogeneous information networks: A structural analysis approach. *SIGKDD Explorations, 14*(2), 20–28. doi:10.1145/2481244.2481248.

Sun, Y., Han, J., Yan, X., Yu, P. S., & Wu, T. (2011). PathSim: meta path-based top-K similarity search in heterogeneous information networks. *PVLDB, 4*(11), 992–1003.

Wang, J., Gao, P., Ma, Y., He, K., & Hung, P. C. K. (2017). A web service discovery approach based on common topic groups extraction. *IEEE Access, 5*, 10193–10208. doi:10.1109/ACCESS.2017.2712744.

Wang, W., Barnaghi, P. M., & Bargiela, A. (2010). Probabilistic topic models for learning terminological ontologies. IEEE Transactions on Knowledge and Data Engineering, , *22*(7), 1028–1040. doi:10.1109/TKDE.2009.122.

Xia, B., Fan, Y., Tan, W., Huang, K., Zhang, J., & Wu, C. (2015a). Category-aware API clustering and distributed recommendation for automatic mashup creation. *IEEE Transactions on Services Computing, 8*(5), 674–687. doi:10.1109/TSC.2014.2379251.

Xia, B., Fan, Y., Tan, W., Huang, K., Zhang, J., & Wu, C. (2015b). Category-aware API clustering and distributed recommendation for automatic Mashup creation. IEEE Transactions on Services Computing, *8*(5), 674–687. doi:10.1109/TSC.2014.2379251.

Xu, W., Cao, J., Hu, L., Wang, J., & Li, M. (2013). A social-aware service recommendation approach for Mashup creation. In *Proceedings of the 2013 IEEE 20th International Conference on Web Services* (pp. 107–114).

Xu, Y., Yin, J., Deng, S., Xiong, N. N., & Huang, J. (2016). Context-aware QoS prediction for web service recommendation and selection. *Expert Systems with Applications, 53*, 75–86. doi:10.1016/j.eswa.2016.01.010.

Yao, L., Mimno, D., & Mccallum, A. (2009). Efficient methods for topic model inference on streaming document collections. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 937–946).

Yu, L., Zhou, J., Zhang, J., Wei, F., & Wang, J. (2015). Time-aware semantic web service recommendation. (pp. 664–671). doi: 10.1109/SCC.2015.95.

Zhang, N., Wang, J., He, K., Li, Z., & Huang, Y. (2018). Mining and clustering service goals for RESTful service discovery. *Knowledge and Information Systems*. doi:10.1007/s10115-018-1171-4.

Zhang, Y., Jatowt, A., & Tanaka, K. (2016). Towards understanding word embeddings: Automatically explaining similarity of terms. In *Proceedings of the 2016 IEEE International Conference on Big Data (Big Data)* (pp. 823–832). doi:10.1109/BigData.2016.7840675.

Zheng, Z., Ma, H., Lyu, M. R., & King, I. (2013). Collaborative web service QoS prediction via neighborhood integrated matrix factorization. *IEEE Transactions on Services Computing, 6*(3), 289–299. doi:10.1109/TSC.2011.59.

Zhou, T. C., Lyu, M. R.-T., King, I., & Lou, J. (2015). Learning to suggest questions in social media. *Knowledge and Information Systems, 43*(2), 389–416. doi:10.1007/s10115-014-0737-z.