Learning Context-Aware Service Representation for Service Recommendation in Workflow Composition

Xihao Xie, Jia Zhang
Department of Computer Science
Southern Methodist University
Dallas, USA
{xihaox; jiazhang}@smu.edu

Rahul Ramachandran
Marshall Space Flight Center
NASA
Huntsville, USA
rahul.ramachandran@nasa.gov

Tsengdar J. Lee
Science Mission Directorate
NASA Headquarters
Washington, USA
tsengdar.j.lee@nasa.gov

Seungwon Lee
Jet Propulsion Laboratory
NASA
Pasadena, USA
seungwon.lee@jpl.nasa.gov

2022 IEEE/ACIS 22nd International Conference on Computer and Information Science (ICIS) | 978-1-6654-9463-2/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICIS54925.2022.9882442

***Abstract*—As increasingly more software services have been published onto the Internet, it becomes critical yet highly challenging to recommend suitable services to facilitate scientific workflow composition. This paper proposes a novel Natural Language Processing (NLP)-inspired approach to recommending services throughout a workflow development process, based on incrementally learning latent service representation from workflow provenance. A workflow composition process is formalized as a step-wise, context-aware service selection procedure, which is mapped to next-word prediction in a natural language sentence generation. Historical service dependencies are extracted from workflow provenance to build and enrich a knowledge graph. Each path in the knowledge graph reflects a scenario in a data analytics experiment, which is analogous to a sentence in a conversation. All paths are thus formalized as composable service sequences and are mined, using various patterns, from the established knowledge graph to construct a corpus. Service embeddings are then learned by applying deep learning model from the NLP field. Extensive experiments on the real-world dataset demonstrate the effectiveness and efficiency of the approach.**

***Index Terms*—service representation, service recommendation, workflow composition**

I. Introduction

In recent years, increasingly more software programs have been deployed and published onto the Internet as reusable web services, so-called APIs or services for short. Scientific researchers can thus leverage and compose existing services to build new data analytics experiments, so-called scientific workflow or workflow in short [1]. However, studies over the life science field [2] have revealed that the reusability rate of life science services in workflows remains rather low. Earlier studies believe one of the major obstacles making researchers unwilling to reuse existing services is the data shimming problem [5], meaning preparing and transforming data types to feed in the required inputs of a downstream service. Unless a user fully understands each of the parameters, both of its syntactic and semantic meaning and requirements, he/she may not feel comfortable to reuse the service.

Our research project moves one step further, to argue that the adoption of a software service is not only dependent on its direct upstream service, but also the context of all selected services in the current workflow under construction. In order to add into consideration of such workflow-contextual information for more precise service recommendation in a recommend-as-you-go manner, this research presents a novel machine learning technique over service provenance knowledge graph. Inspired by the latest advancements in Natural Language Processing (NLP), we formalize the problem of service recommendation as a problem of next service prediction, where services and workflows are mapped to "tokens" and "sentences" in NLP, respectively. Note that some services used together (unit of work [4]) are analogous to "phrases" in NLP, thus are mapped to "tokens" as well. In this way, our goal has turned into predicting and recommending the next suitable services that might be used for a user during the process of workflow composition.

Inspired by Word2Vec [9], we leverage the skip-gram model to learn latent representation of services and their relationships for context-aware workflow recommendation. A corpus of "sentences" (i.e., service chains extracted from historical workflow provenance) are generated from the knowledge graph. Our rationale is that, each path (i.e., a sequence of services) in a workflow reflects a scenario of data analytics experiment, which is analogous to a sentence in a conversation. Thus, all paths are extracted from the knowledge graph, analogous to all sentences are accumulated to train a computational model from NLP. In other words, a service sequence carries a context when a service was invoked in the past.

To the best of our knowledge, we make the first attempt to seamlessly exploit the state-of-the-art machine learning techniques in both NLP and knowledge graph to facilitate service recommendation and workflow composition. Our extensive experiments over real-world dataset have demonstrated the effectiveness of our approach. In summary, our main contributions are three-fold:

- We formalize the service recommendation problem as a problem of context-aware next service prediction.
- We develop a technique to generate a corpus of service-oriented-sentences (service sequences) by traversing over the knowledge graph established from workflow and service usage provenance.
- We develop an approach to learning service representations offline based on sequential context information in accumulated knowledge graph and then recommending potential services real-time.

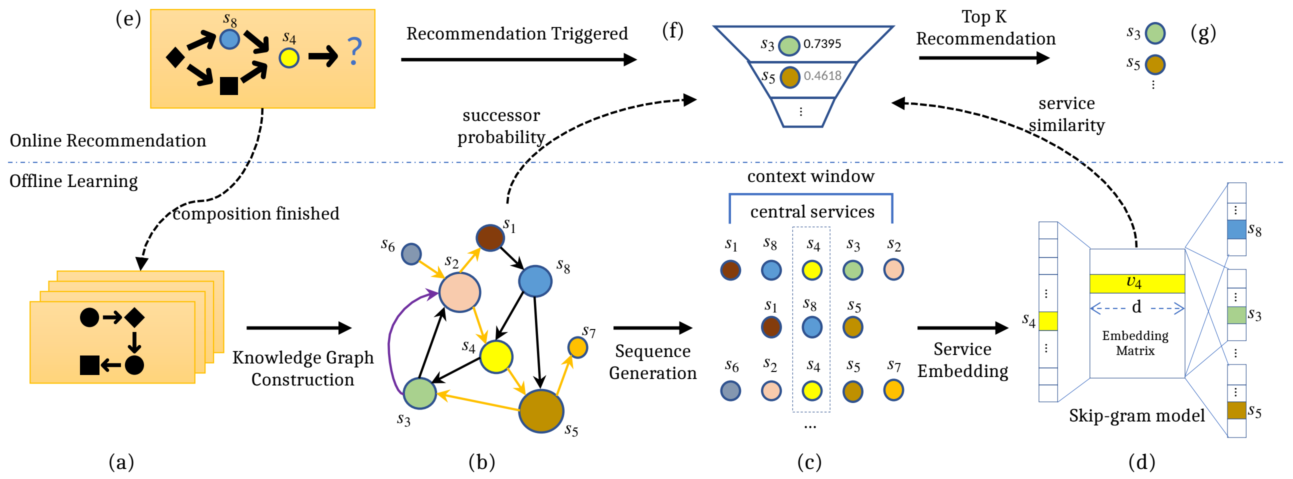

Fig. 1. Blueprint of proposed approach.

II. Context-Aware Service Recommendation Approach

Fig. 1 presents the high-level blueprint of our methodology. First, based on workflow provenance (a), we construct a knowledge graph (b) by extracting service dependencies from workflow structure. Second, we generate sequences of service tokens (c) from the knowledge graph. Third, we use language modeling techniques to learn latent representations of services (d). As shown in Fig. 1, the above steps are conducted offline. The results of the offline learning phase will support the online recommendation at run-time. Given a workflow which is under composition online (e), we rank the potential services respect to their semantic similarities in the context (f), and recommend top-K of them (g) to the user at real time. Note that once a new workflow is completely composed, it will be saved into the workflow repository thus to trigger incremental offline learning. This section will present the procedure in detail.

A. Problem Definition

Let $\mathcal{W} = \{w_1, w_2, ..., w_{|\mathcal{W}|}\}$ denote a set of workflows, $\mathcal{C} = \{c_1, c_2, ..., c_{|\mathcal{C}|}\}$ be a set of available software services, $\mathcal{T} = \{t_1, t_2, ..., t_{|\mathcal{T}|}\}$ be a set of service tokens. Note that each service token represents either an individual service or a subset of services, where an example may be a unit of work (several services always used together) [4]. Let list $S_{w'} = \left[s_1^{w'}, ..., s_k^{w'}, ..., s_{|S_{w'}|}^{w'}\right]$ denote a service sequence with length as $|S_{w'}|$ for workflow $w' \notin \mathcal{W}$ which has not been completed yet, where $s_k^{w'}$ is the service token $t_k \in \mathcal{T}$ generated at time step k and $t_k \subseteq \mathcal{C}$ is a subset of services. Given a specific existing service sequence $S_{w'}$ of a workflow w' under construction, our goal is to rank the probability over all service tokens $t \in \mathcal{T}$ at next time step $|S_{w'}| + 1$:

$$p_t = p(t_{|S_{w'}|+1}^{w'} = t | S_{w'}) \quad (1)$$

and recommend top-n tokens list $T \subseteq \mathcal{T}$, where n is the size of the list T.

B. Knowledge Graph Construction

Similar to our earlier work [6], we define web service knowledge graph (WSKG) as a directed graph (digraph) carrying historical service invocation dependencies extracted from workflow repository:

$$WSKG = \langle \mathcal{C}, \mathcal{R} \rangle \quad (2)$$

where each software service $c \in \mathcal{C}$ is regarded as an entity, and $\mathcal{R} = \{r_{i,j}^w\}$ is a set of relationships between service entities. $r_{i,j}^w = \langle c_i, c_j, w \rangle$ refers to a relationship that c_i is an upstream service of c_j in workflow w. We can regard the relationship $r_{i,j}^w$ as an edge starting from c_i and ending at c_j with label w. Note that there might be multiple edges between two nodes in the knowledge graph with different labels, meaning that such a service invocation dependency happens in multiple workflows.

C. Offline Service Representation Learning

The offline representation learning phase aims to learn latent service representations over the service sequence corpus. Two models, skip-gram and CBOW in Word2Vec, are widely applied neural network models in NLP to learn word representations given a corpus of sentences [9]. In our study, we decide to employ the skip-gram model mainly because it works well with small amount of training data and represents well even for rare tokens [12]. In our scenarios, existing scientific workflow datasets are not so large as that in the field of NLP. Furthermore, the significant amount of long-tail services (e.g., newly published services) may be able to receive reasonable exposure using the skip-gram model, although they rarely appear in the service sequence corpus.

Algorithm 1 illustrates the step-wise procedure of offline representation learning. Note that during the training process, we update vectorized representations with stochastic gradient descent, as shown in lines 6 and 7.

D. Online Service Recommendation

Following the skip-gram model, given a specific sequence of service tokens $[t_{l-u}, ..., t_{l-1}, t_l]$, the offline learnt service representations enable us to identify the top-K most *relevant* service tokens to be the most potential service tokens following t_l. According to the skip-gram model, though, the predicted contextual service tokens may not only be upstream of t_l in some workflows, but also be downstream service tokens as well. However, in the scenario of workflow composition, what we need to recommend is the service tokens that are potentially appear after t_l. Therefore, we define a scoring function which is used to rank next potential service tokens in descending order as follows:

$$score(t_l, t) = p_{suc}(t_l, t) \times sim(t_l, t) \tag{3}$$

$$p_{suc}(t_l, t) = \frac{\exp(N_{suc}(t_l, t))}{\exp(N_{pre}(t_l, t)) + \exp(N_{suc}(t_l, t))} \tag{4}$$

where $p_{suc}(t_l, t)$ empirically models the probability that t appears after t_l, $N_{suc}(t_l, t)$, $N_{pre}(t_l, t)$ are the numbers of occurrences that any service $c \in t$ appeared in repository as a successor or a precursor service of t_l, respectively. $sim(t_l, t)$ is the similarity between t_l and t that can be calculated from the service representations learnt offline.

Algorithm 1 Offline Learning Service Representation

```
Input: workflows $\mathcal{W}$, windows size $w$, dimension size $d$ and
    sequence generation type T
Output: vectorised service representations $\Omega \in \mathbb{R}^{|\mathcal{C}| \times d}$
 1: Initialize $\Omega \in \mathbb{R}^{|\mathcal{C}| \times d}$
 2: $\mathcal{D} \leftarrow GenerateSequences(\mathcal{W}, \mathrm{T})$
    // generate service token sequences according to specific gen-
    eration type T, see in next section.
 3: for each sequence $S \in \mathcal{D}$ do
 4:     for each token $t_i \in S$ do
 5:         for each $t_j \in S[i - w : i + w]$ do
 6:             $J(\Omega) = -\log p(t_j | \Omega(t_i))$
 7:             $\Omega = \Omega - \eta \times \frac{\partial J}{\partial \Omega}$ // $\eta$ is the learning rate.
 8:         end for
 9:     end for
10: end for
```

III. Service Sequences Generation

In this section, we explore three different ways, each of which might be adopted for a specific scenario, to generate sequential service tokens from the constructed knowledge graph WSKG introduced in the previous section. The three generation methods are: DFS-based generation, BFS-based generation, and PW-based generation.

A. Depth First Search (DFS) based Generation

In this method, we consider a workflow composition process as a path extension process from start to end. Particularly, we consider individual workflows. In order to model the sequential behaviour of composing a specific workflow $w \in \mathcal{W}$, depth first search method is applied in WSKG, along the path with the label w to generate sequences of service tokens. Specifically, for any service entity $c_i \in \mathcal{C}$, let $S^w_{i,m} = t^i_1 \xrightarrow{w} t^i_2 \xrightarrow{w} ... \xrightarrow{w} t^i_{|S^i_w|}$ denote a generated sequence starting from c_i and going along the path with label w, representing a workflow, to next unvisited neighbor services until meeting terminal services which have no successor services along label w. Here $t^i_1 = c_i$, $t^i_j \in \mathcal{T}$ and $\forall t^i_j$ has $|t^i_j| = 1$, which means every token is a single service. $m \in [1, M]$ and M is the total number of sequences starting from c_i.

Note that for a workflow with label w, the sequences are generated from not only its starting services but also all intermediate services. In this way, the generated sequences can cover as many as possible sequential dependencies among services in the workflow w.

B. Breadth First Search (BFS) based Generation

The main idea of the BFS based service tokens generation lies in multiple items recommendation. In order to better achieve business goal, a real-world recommender system typically not only recommend single items but also a bundle of items. Actually, recommending next bundle of services can be regarded as recommending next basket of items that a user might want to buy in a single visit in e-commerce scenarios [7], [8]. Our earlier research also studied how to recommend unit of work (UoW) with a collection of services usually used together, based on network analysis [4].

For the similar reason, we consider service bundles as services tokens when generating sequential services tokens from WSKG. Applying BFS based generation strategy, for any service $c_i \in \mathcal{C}$ in a workflow w, the sequence of service tokens starting from c_i can be generated as: $S^w_i = \left[t^i_1, t^i_2, ..., t^i_{|S^w_i|}\right]$, where $t^i_k \subseteq \mathcal{C}$ is a set of multiple services that are at the k^{th} level of the breadth first searching path. Specially, t^i_1 is a set containing only c_i, and it can be either a starting service or an intermediate service.

C. Probabilistic Walk (PW) based Generation

Inspired by DeepWalk [11], which uses random walk to generate sequences of nodes to learn latent representations for vertices in a graph, we leverage a variant of random walk, i.e., probabilistic walk, to generate sequences of services based on the constructed knowledge graph WSKG.

Recall that the DFS and BFS based generation algorithms discussed earlier both generate a service sequence starting from service token t_j along the path with a specific label w. In contrast, the probabilistic walk based sequence generation algorithm considers the generation of next service token rooted at t_j as a stochastic process, with random variables $T^1_j, T^2_j, ..., T^l_j$ where T^{l+1}_j is a service generated with probabilities from the neighbors of service token t_l. Note that, such a service token T^{l+1}_j existing in the sequence of $\left[T^1_j, T^2_j, ..., T^l_j\right]$ is not allowed, meaning that the generated sequence is acyclic. On the one hand, restarting from a vertex did not show any improvement in our experimental results. On the other hand, it is generally meaningless to invoke a previously invoked service in the same workflow. Specifically, for a service $u \in \mathcal{C}$, let $N_u \subseteq \mathcal{C}$ denote the set of directed neighbor services in the whole WSKG, we model the probability $p_{u,v}$ that service token v can be generated after u:

$$p_{u,v} = p(v|u) = \sigma(\frac{o_{u,v}}{o_u}) = \frac{\exp(\frac{o_{u,v}}{o_u})}{\sum_{n \in N_u} \exp(\frac{o_{u,n}}{o_u})} \quad (5)$$

where $v \in N(u)$, σ is the commonly used softmax function, $o_{u,v}$ is the number of occurrence of relationships between u and v in the whole WSKG and $o_u = \sum_{n \in N(u)} o_{u,n}$.

Two factors may impact the effectiveness of the PW based generation strategy. One factor is the length of a visiting path, i.e., l, which determines when to stop while walking along a path. The other factor is the number of walks starting at each vertex, i.e., θ. In practice, without specifying θ, some linkages may not be traversed. As a result, the generated sequences may not cover all tangible dependency relationships, tampering the ability of trained representations to predict next suitable services. As [11] suggested, although it is not strictly required to do this, it is a common practice to tune the two factors to speed up the convergence of stochastic gradient descent in Algorithm 1. In the next section of experiments, We will discuss in detail the effects of l and θ.

IV. Experiments

A. Dataset

Our testbed is myExperiment.org [3]. We examined all Taverna-generated workflows published on myExperiment.org up to October 2021, with a total of 2,910 workflows and 9,120 services. Table I lists the summarized information of the dataset.

TABLE I
Statistical Information about myExperiment Dataset

Total # of workflows		2,910
Total # of services		8,837
Total # of sequences	BFS	10,788
	DFS	41,163
	PW	50,314
Average length of sequences	BFS	5.0
	DFS	9.0
	PW	7.0

B. Experimental Setup

We randomly used 80% of the workflows to generate service sequences as training data to learn service representations. The remaining 20% of workflows were utilized as the test data. In the offline learning phase, we set the number of window size as 3, and the dimension size d as 50. The generated service sequences with length smaller than 2 were removed from the training set. As for the online recommendation phase, similar to the *leave-one-out* task that has been widely used in sequential recommendation, we adopted the *leave-last-service-out* cross-validation to evaluate our approach.

For each workflow in the test set, we removed the terminate services of the workflow and used the services just before the terminate services as the inputs of the trained representation model to rank top K service tokens as candidates. Hence, the experiments were turned to evaluate whether the recommended service tokens hit the ground truth. Additionally, in the test set, we removed the linkages which never occurred in the sequences generated from the training data, considering that the skip-gram model is not able to predict tokens that are not in the vocabulary.

C. Evaluation Metrics

All of our evaluation metrics aim to measure whether the recommended top K entries hit the ground truth or not. For DFS-based and PW-based generation strategy, each entry is a single service token. An entry $e_i \in R$ hits the ground truth G if $e_i \in G$, where R is the recommended result, and $|R| = K$. For BFS-based generation strategy, an entry $e_i \subseteq \mathcal{C}$ in the recommended list might be a service token comprised of multiple services. In this case, the entry e_i hits the ground truth G if $\exists c_j \in e_i$ such that $c_j \in G$.

We employed *PRE@K*, *REC@K* and *F1@K*, which are short for *precision*, *recall* and *F1 Score*, respectively, as the evaluation metrics of our recommendation approach. For every workflow in the testing set, they can be calculated as follow:

$$precision = \frac{|R \cap G|}{|R|} \quad (6)$$

$$recall = \frac{|R \cap G|}{|G|} \quad (7)$$

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \quad (8)$$

where $|R \cap G|$ means the number of hit entries in the top K recommended results. Additionally, note that the index of the hit result (i.e., *idx*) in the recommended list is a significant factor to users' willingness to reuse recommended services. We thus introduced a variant of MRR, aka *VMRR*, as an evaluation metric as well. It helps us to capture the overall performance of the rank of the hit entries in the recommended results. The calculation of *VMRR* is:

$$VMRR = \begin{cases} 0 & R \cap G = \varnothing \\ \frac{|R| - idx}{|R|} = 1 - \frac{idx}{K} & else \end{cases} \quad (9)$$

For these four metrics, we reported their mean values over all workflows in the test dataset. The higher the value, the better the performance.

D. Parameter Sensitivity

We designed experiments to evaluate how changes to l and θ will effect the performance of the probabilistic walk based generation strategy. In such experiments, duplicate generated sequences were removed. Figs. 2, 3 and 4 show the effects of l and θ to the PW-based generation strategy. In terms of precision, recall and F1, $l = 5$ is better than $l = 3$, $l = 10$, $l = 15$ and $l = 20$. Note that the median and mean lengths of workflows are 8 and 9.49, respectively, both of which are closed to 9. Approximately, for a workflow with length of 9, the average length of generated sequences with BFS-based

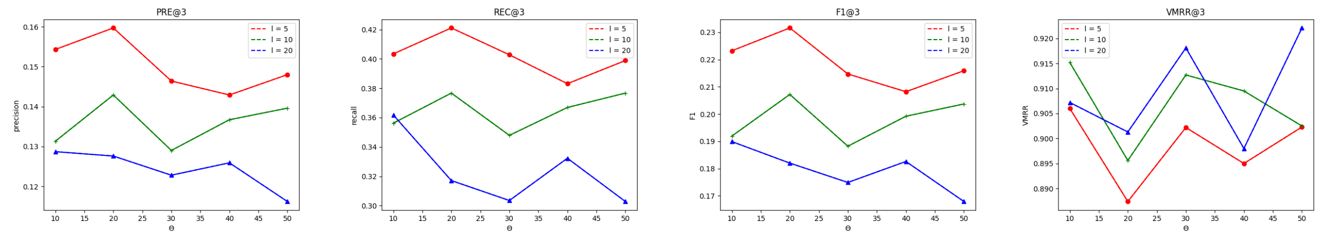

Fig. 2. The effect of l and θ with top-3 recommendations

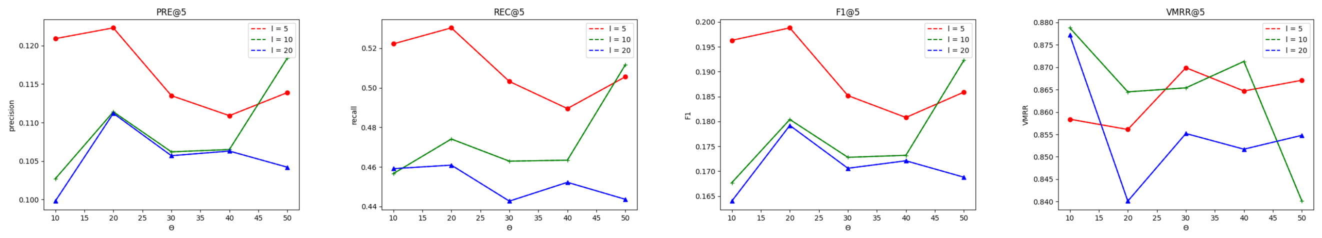

Fig. 3. The effect of l and θ with top-5 recommendations

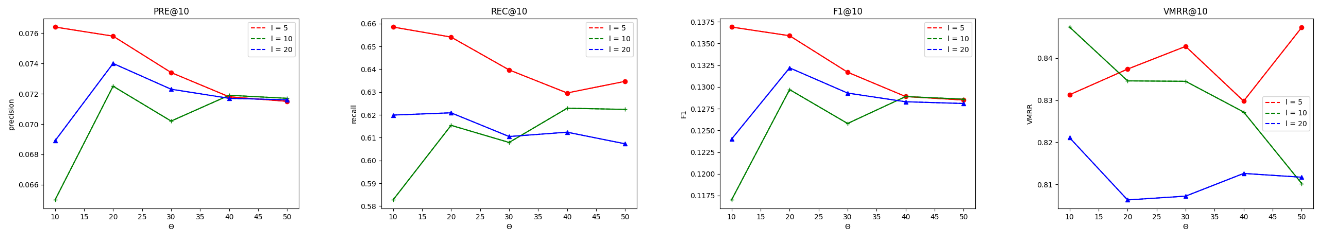

Fig. 4. The effect of l and θ with top-10 recommendations

TABLE II
Overall Recommendation Results

Metrics	PRE@3	REC@3	F1@3	VMRR@3	PRE@5	REC@5	F1@5	VMRR@5	PRE@10	REC@10	F1@10	VMRR@10
BFS	0.3059	0.6258	0.4109	0.7651	0.2191	0.6897	0.3323	0.8730	0.1031	0.7397	0.1810	0.9412
DFS	0.2384	0.6142	0.3435	0.7516	0.1591	0.6721	0.2573	0.8612	0.0893	0.7455	0.1595	0.9362
PW	0.2564	0.6457	0.3609	0.7735	0.1668	0.6967	0.2692	0.8681	0.0912	0.7617	0.1629	0.9392
BFS-DR	**0.3144**	0.6465	**0.4231**	0.7789	**0.2249**	0.6979	**0.3402**	0.8779	**0.1099**	0.7579	**0.1920**	0.9445
DFS-DR	0.2449	0.6243	0.3518	0.7607	0.1630	0.6835	0.2632	0.8668	0.0922	0.7624	0.1645	0.9404
PW-DR	0.2668	**0.6862**	0.3842	**0.8186**	0.1808	**0.7609**	0.2922	**0.8994**	0.0993	**0.8226**	0.1772	**0.9552**

generation strategy is around 5. A lower l results in lower coverage over all linkages between services, which reduces the capability of discovering suitable services following a specific service. A higher l makes it more likely to generate service sequences that are not existed in workflows.

The parameter θ in this work is similar to the parameter γ in [11]. For θ, 10 is almost the best option. The reason might be that, for all nodes in the constructed WSKG, their mean out-degrees and mean numbers of downstream nodes are 2.38 and 1.69, respectively, both of which are close to 2.0. Approximately, for a binary tree with height of 5, to reach a specific leaf from the root with probabilities at forks, it needs about 10 attempts. On the one hand, a lower θ might result in higher probability to miss tangible dependencies that are useful for next service recommendation. On the other hand, a higher θ tends to generate duplicate sequences, which might be misleading for representation learning.

In summary, our experiments demonstrated that l and θ matter for probabilistic walk based sequence generation strategy. Our suggestion is that when applying the PW-based strategy, it will be good to get a glimpse at the length of workflows and the out-degree of service nodes in the constructed graph.

E. Performance Comparison

We designed experiments to answer two research questions. (1) Which sequence generation strategy acts the best? (2)

Should duplicates from the generated sequences be removed? Table II presents the overall recommendation performance metrics over six generation strategies. BFS-DR, DFS-DR and PW-DR are results of BFS, DFS and PW with duplicate sequences removed, respectively. For each metric, the best result is highlighted in bold and the second best is underlined. According to the lessons we learnt as discussed earlier, we set l and θ to 5 and 10, respectively, for PW and PW-DR.

RQ1: Which sequence generation strategy is the best? According to the experimental results, in terms of precision and F1, BFS-based strategies outperform others; in terms of recall and VMRR, PW-DR performs the best. Specifically, we hold that recall and VMRR are better metrics than precision and F1 to gauge the performance between different strategies. Recall that our goal is to find the most suitable services following a specific service. It means that we care more about how many adjacent services we can fetch, instead of how many candidates are adjacent services. The higher recall means higher possibility to help a user increase the efficiency of service composition. Besides, for the hit index, we hope it could be as low as possible. The higher VMRR could encourage users to reuse the recommended services more. Therefore, our experiments have demonstrated that PW-based strategies outperform others.

Table II shows that the recall remains around 70% for all strategies at $K = 5$, meaning that in the ground truth, over half of adjacent services are found if we recommend 5 services. Especially, the VMRR@5 is around 0.90. It means that generally the first or the second entry in the recommended list hits the ground truth. We thus have demonstrated that our approach is quite useful to help users save time to composite a workflow by reusing recommended services.

RQ2: Should duplicates from the generated sequences be removed? For all three aforementioned types of strategies, there might appear duplicate service sequences. The reason is that some services and their dependency relationships in a workflow w_i might exist in another workflow w_j, in a form of service chain. As for the PW-based strategy, three factors may result in duplicates: the length of a path, the frequencies of the dependency relationships in the path and the predefined parameter θ. A shorter path with more frequent dependency relationships tends to be generated as a sequence with higher probability in case of a higher θ. Similar to insurmountable duplicates in a natural language corpus [10], duplicates in the corpus of service token sequences generated from the knowledge graph might impact the effectiveness of our approach. It is a *feature-but-not-bug* problem.

Table II also shows that, for all generation strategies, removing duplicates achieves better performance than remaining them in the training set. As for PW-based strategies, the higher probability of a linkage is, the more potential it would be traversed while walking, which makes it more likely to generate duplicate sequences. PW-DR is better than PW in terms of all metrics. Therefore, we suggest to investigate the impact of duplicate sequences in advance, when applying language modeling techniques to learn service representations.

V. Conclusions & Future Work

In this paper, we have applied deep learning methods in NLP to study workflow recommendation problem. Workflow composition process is viewed as a sequential service generation process, and the problem of service recommendation is formalized as a problem of next word prediction in NLP. A knowledge graph is constructed on top of workflow provenance, and we develop three strategies to create a corpus of service sequences from the knowledge graph. Deep learning method in NLP is then applied to learn service representations. The resulted service embeddings are used to support incremental workflow composition at run-time, associated with structural information embedded in the knowledge graph.

In the future, we plan to extend our research by taking into account users' profile information to enable personalized workflow recommendation.

Acknowledgment

Our work is partially supported by National Aeronautics and Space Administration under grants 80NSSC21K0576 and 80NSSC21K0253.

References

[1] A. L. Lemos, F. Daniel, & B. Benatallah, "Web Service Composition: A Survey of Techniques and Tools", *ACM Computing Surveys (CSUR)*, 48(3), 2015, pp. 1-41.

[2] W. Tan, J. Zhang, & I. Foster, "Network Analysis of Scientific Workflows: A Gateway to Reuse", *Computer*, 43(9), 2010, pp. 54-61.

[3] C. A. Goble, J. Bhagat, S. Aleksejevs, D. Cruickshank, D. Michaelides, D. Newman, M. Borkum, S. Bechhofer, M. Roos, P. Li, & D. D. Roure, "myExperiment: A Repository and Social Network for the Sharing of Bioinformatics Workflows", *Nucleic Acids Research*, 38, 2010, pp. W677-W682.

[4] J. Zhang, M. Pourreza, S. Lee, R. Nemani, & T. J. Lee, "Unit of Work Supporting Generative Scientific Workflow Recommendation", in Proceedings of International Conference on Service-Oriented Computing, Nov. 2018, pp. 446-462.

[5] J. Zhang, W. Wang, X. Wei, C. Lee, S. Lee, L. Pan, & T. J. Lee, "Climate Analytics Workflow Recommendation as A Service-provenance-driven Automatic Workflow Mashup", in Proceedings of IEEE International Conference on Web Services, Jun. 2015, pp. 89-97.

[6] J. Zhang, W. Tan, J. Alexander, I. Foster, & R. Madduri, "Recommend-as-you-go: A Novel Approach Supporting Services-oriented Scientific Workflow Reuse", in Proceedings of IEEE International Conference on Services Computing, Jul. 2011, pp. 48-55.

[7] F. Yu, Q. Liu, S. Wu, L. Wang, & T. Tan, "A Dynamic Recurrent Model for Next Basket Recommendation", in Proceedings of The 39th ACM SIGIR International Conference on Research and Development in Information Retrieval, Jul. 2016, pp. 729-732.

[8] S. Rendle, C. Freudenthaler, & L. Schmidt-Thieme, "Factorizing Personalized Markov Chains for Next-basket Recommendation", in Proceedings of The 19th International Conference on World Wide Web, Apr. 2010, pp. 811-820.

[9] T. Mikolov, K. Chen, G. Corrado, & J. Dean, "Efficient Estimation of Word Representations in Vector Space", arXiv preprint, arXiv:1301.3781, 2013.

[10] A. Schofield, L. Thompson, & D. Mimno, "Quantifying the Effects of Text Duplication on Semantic Models", in Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, 2017, pp. 2737-2747.

[11] B. Perozzi, R. Al-Rfou, & S. Skiena, "Deepwalk: Online Learning of Social Representations", in Proceedings of The 20th ACM International Conference on Knowledge Discovery and Data Mining, Aug. 2014, pp. 701-710.

[12] K. Stoitsas, "The Use of Word Embeddings for Cyberbullying Detection in Social Media", Doctoral dissertation, Tilburg University, Jul. 2018.