Facilitating Data-Centric Recommendation in Knowledge Graph

Jia Zhang[1], Maryam Pourreza[1], Rahul Ramachandran[2], Tsengdar J. Lee[3], Patrick Gatlin[2], Manil Maskey[2],
Amanda Marie Weigel[4]
[1]*Department of Electrical and Computer Engineering, Carnegie Mellon University, USA*
[2]*NASA/MSFC, USA*
[3]*Science Mission Directorate, NASA Headquarters, USA*
[4]*University of Alabama Huntsville, USA*
{jia.zhang, maryam.pourreza}@sv.cmu.edu;{rahul.ramachandran, tsengdar.j.lee, patrick.gatlin,manil.maskey}@nasa.gov;
amw0039@uah.edu

***Abstract*—In order to help Earth scientists share knowledge and expedite scientific exploration, NASA is developing NASA Earth Science Enterprise (ESE) whose underlying basis is NASA Science Knowledge Graph (SKG). This paper focuses on the information model design of the SKG, where entity- and relationship-encapsulated features are extracted as first-class citizens in SKG. The rationale is that typological structural analysis can thus be exploited for runtime knowledge extraction, discovery, and prediction, in addition to semantic analysis. Based on the information model, a knowledge discovery technique is equipped to answer queries and provide personalized recommendation based upon the SKG. Deep learning techniques, i.e., Stacked Denoising Auto-Encoders (SDAE) and Translating Embeddings (TransE), are applied to detect structural similarity and explore paths at runtime, respectively, supporting higher scalability and performance. Experiments on a science-oriented testbed serves as a proof of concept and demonstrates the feasibility and effectiveness of the techniques.**

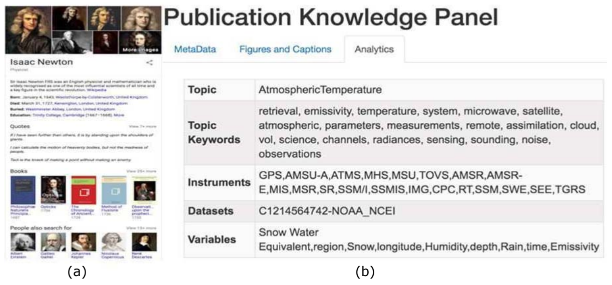
***Index Terms*—Science knowledge graph, knowledge discovery, deep learning**

I. Introduction

Data-centric recommendation has become a critical issue in the modern scientific research. Take Hurricane science as an example. It encompasses a wide array of research working to improve Hurricane observation methods, modeling and prediction, data assimilation, understanding of hurricane impacts, and dynamic and physical processes. To conduct this research, a scientist's goal is to use the best data, tools, methodologies, and models available to answer a hypothesis. Thus, the scientist needs to accurately locate the sources where data can be found, available user guides, information on how the data was acquired, and available tools and models to use with this data. Such a process has been proven to be highly tedious, time consuming and inefficient [7] without computer support.

To address the problem, researchers have been studying data recommendation from various aspects. Our previous research has proposed a concept of data service container that treats data as first-class citizen and organizes around data the software and procedures that can process the data [18]. In this paper, we move one step forward to study the scalability of realizing data service containers and explore how to enable the data service containers to provide direct answers to user queries in an effective manner.

Google Knowledge Graph[1] (name changed to Knowledge Vault) project aggregates structured and detailed information about defined topic, so that users could resolve their query without having to navigate and assemble information manually. Fig. 1(a) shows the knowledge panel when searching for "Isaac Newton." The knowledge panel summarizes information that people commonly wish to know about Newton, by aggregating his profile including his short bio and major achievements. However, a researcher may be more interested in knowing his science-related information like his theories and major experiments conducted. Such information may be scattered among Newton's publications and will not be provided by Google Knowledge Vault.



Topic	AtmosphericTemperature
Topic Keywords	retrieval, emissivity, temperature, system, microwave, satellite, atmospheric, parameters, measurements, remote, assimilation, cloud, vol, science, channels, radiances, sensing, sounding, noise, observations
Instruments	GPS,AMSU-A,ATMS,MHS,MSU,TOVS,AMSR,AMSR-E,MIS,MSR,SR,SSM/I,SSMIS,IMG,CPC,RT,SSM,SWE,SEE,TGRS
Datasets	C1214564742-NOAA_NCEI
Variables	Snow Water Equivalent,region,Snow,longitude,Humidity,depth,Rain,time,Emissivity

Fig. 1. Knowledge graph and science knowledge graph

In addition to human-related search, domain scientists may be more interested in science keyword-related search. For example, an Earth scientist may query over keyword "Hurricane." What she would really like to know, instead of Wikipedia explanation of the term and a huge number of websites containing the Hurricane keyword in text, is the information regarding a collection of questions such as: Which datasets should I study on this topic? How to process this dataset? Any results came from this dataset? What did other researchers do

[1]https://www.google.com/intl/en_us/insidesearch/features/search/knowledge.html

on this dataset? Can I repeat their process? and Can I revise their process and rerun?

Limited on People, Places, and (general) Things, answers of such research-oriented queries will not be answered by Google Knowledge Vault. Thinking about where to find possible answers to these questions, though, keys are usually scattered and hidden in publication literature. For an example as shown in Fig. 1(b), analytics over an Earth science journal paper typically will reveal the datasets used in the study, including the variables examined and algorithms used to process the datasets.

Therefore, this on-going project aims to design and develop a Science Knowledge Graph (SKG) oriented to scientific researchers. The ultimate goal is to provide a one-stop gateway that is able to mine, ***all together***, data-centric knowledge including datasets, tools and models, algorithms, statistical analyses, projects, topics, hypotheses and conclusions, as well as hidden information. In this paper, we focus on the data model and fundamental network structure of the SKG. Based on them, we propose a tailored technique of how to record, retrieve, infuse, and refer data-centric knowledge.

Our contributions in this paper are summarized in three-fold. First, we have developed an information model to carry key information entities and relationships around scientific data. Traditional entity- and relationship-encapsulated features are extracted as first-class citizens in our SKG, so that implicit feature relationships can be turned into explicit structural relationships in the SKG. As a result, we could leverage structural relationships embedded in SKG to explore hidden information at runtime, in contrast to semantic analysis where application-specific rules have to be manually coded. Second, based upon the information model and knowledge network constructed, we have developed a tailored technique to answer community-oriented queries and provide personalized recommendation. Third, we have applied deep learning algorithms (Stacked Denoising Auto-Encoders (SDAE) [14] and Translating Embeddings (TransE) [2] to detect structural similarity and explore paths at runtime, respectively, for addressing scalability and performance issues.

We have also conducted a collection of experiments over a science-oriented testbed (myExperiment.org). Our experiments have demonstrated the viability and effectiveness of our techniques. Note that although focusing on Earth science, our techniques established can be applied to other domains.

The remainder of the paper is organized as follows. In Section 2, we discuss related work. In Section III, we introduce two motivating examples and our strategy. In Section IV and V, we present the information model and tailored technique for knowledge inference. In Section VI and VII, we address the potential scalability issue by applying deep learning-powered techniques to perform topological structure analysis and path finding. In Section VIII, we discuss experimental studies. Finally in Section IX, we draw conclusions.

II. Related Work

Recent years have witnessed a collection of knowledge graph projects. Google Knowledge Graph project[1] integrates both structured data (e.g., Freebase[2], Wikipedia[3]) and unstructured data (e.g., the web) to answer user queries. DeepDive [17] from Stanford provides an engineering procedure to process dark data into databases. GeoDeepDive [16] is an extension of DeepDive, focusing on processing dark data from geological articles. Microsoft Academic Graph (MAG)[4] [12] builds a heterogeneous graph aiming to support generic scientific research on scholarly big data. IBM Watson [6] aims to create a question answering (QA) computing system by learning domain knowledge from various sources. On building these knowledge bases, knowledge extraction relies on extensive human involvement by defining hand-crafted extraction rules or hand-labeled training data. According to Nickel et al. [10], existing knowledge base construction efforts can be divided into two categories, based on whether a fixed or open lexicon of entities are employed. In a schema-based approach, on the one hand, tuples (entities and relationships) are represented by globally unique identifiers, and all possible relationships are predefined in a fixed vocabulary. In a schema-free method, on the other hand, an Open Information Extraction technique is adopted so that tuples are represented via normalized but not disambiguated strings.

We have studied the existing technology landscape from three angles: strategies for knowledge graph construction, systems for knowledge graph construction, and tools and algorithms on knowledge graphs. While our other paper reports our efforts from the first two angles [4], this paper focuses on the fundamental information model and knowledge inference techniques on the knowledge network constructed.

DeepDive [17] applies statistical learning and inference to construct knowledge base. Aiming to offer a toolkit, DeepDive provides natural language processing tools, machine learning algorithms, and statistical inference and learning methods to help users to build and customize a knowledge base from multiple input data sources, existing knowledge bases and taxonomies. In contrast to their work focusing on building a knowledge base, our work proposes an information model dedicated to a science knowledge graph equipped with knowledge discovery and inference technique.

Microsoft Academic Graph (MAG) [12] is a heterogeneous graph containing scientific publication records, citation relationships between the publications, authors, organizations, journals, conferences, and fields of study. In contrast to MAG where nodes are high-level concepts encapsulating rich semantic features, our work extracts those features as individual nodes in the knowledge network. Thus, typological structural analysis can be applied for knowledge discovery instead of semantic analysis, for higher scalability and extensibility. To

[2]https://developers.google.com/freebase/

[3]https://en.wikipedia.org/wiki/Wikipedia:Database_download

[4]https://www.microsoft.com/en-us/research/project/microsoft-academic-graph/

eliminate the scalability issue caused by our method, deep learning techniques are applied.

In our previous work, we have developed a software social network to support data processing-centric recommendation leveraging software usage history [19]. In our other efforts, we predict possible software service collaboration in a workflow context [1], and develop a technique to calculate service collaboration possibility based on intent and context [19]. Our earlier works are integrated into our SKG analysis.

III. Motivating Examples and Strategy

We discuss two motivating examples in this section, illustrating how topology-based network analysis and network path exploration techniques can be exploited to facilitate knowledge discovery and recommendation.

A. Motivating Example #1

According to [13], a scientific paper typically encapsulates a collection of common knowledge (semantic) items such as its datasets studied, the focused physical variables, and the algorithms and tools used to study the datasets. A typical graph-oriented data model, however, adopts a low-order modal, meaning that the number of node types is small. For example, a social network is usually a one-modal network, meaning that all nodes bear the same node type, i.e., person. If adopting such a data model, nodes in our knowledge graph will be individual papers and users. As shown in Fig. 2 on the left-hand side, a paper node embeds its corresponding knowledge items as attributes/features. In order to compare the similarity between two papers at runtime, the two nodes are compared according to their carried attributes and values. Such a calculation usually adopts semantic web technique that may not scale at runtime.

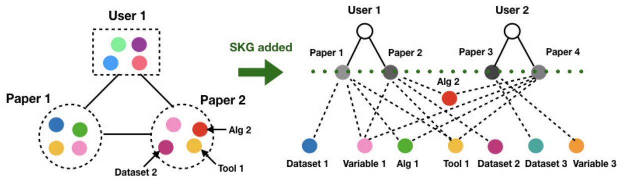


Fig. 2. Motivating Example #1

We intend to tackle the runtime scalability problem, since our project aims to support a large number of queries from the community over the SKG simultaneously. Our basic strategy is to construct a high-order-modal knowledge network, where attributes now become first-class entities in an SKG. As shown in Fig. 2 on the right-hand side, semantic entities are extracted and become individual nodes in SKG, and the aforementioned node-attribute relationship (e.g., between a paper and semantic items) is turned into edge relationship among nodes. In other words, some implicit encapsulation relationships are turned into explicit structural relationships in the knowledge graph. Such extracted semantic entities add abundant structural relationships, which can be leveraged to enable topological mining and investigation. Taking the highly simplified example in Fig. 2, the network structure can be constructed offline. At runtime, assume a query intends to recommend dataset for user 1. By studying the network topology, we may conclude that user 1 and user 2 have similar subtree structure. Although they never co-authored an article, their researches seem to overlap on studying the same variables on the same dataset using the same algorithms and tools. Therefore, we can recommend dataset 3 to user 1 from user 2's experience.

Revisiting the Fig. 2 example, it can be seen that a traditional network model will only include the nodes above the separation line. In contrast, we propose to extract the hidden semantic entities (underneath the separation line) and make them first-class citizens in the network. This example illustrates that our finer granularity in SKG enables topology-based network analysis. Compared with semantics-based similarity computation at run-time, our strategy also enables optimization and parallelism on real-time computation because calculations over different nodes can be conducted simultaneously.

B. Motivating Example #2

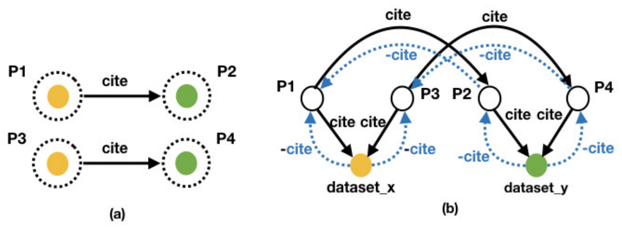


Fig. 3. Motivating Example #2

Fig. 3 shows another motivating example. Fig. 3(a) describes a scenario comprising two paper citations. paper_1 cites paper_2, and paper_3 cites paper_4. Note that both paper_1 and paper_3 cite dataset_x, and both paper_2 and paper_4 cite dataset_y. If using the traditional way to represent their relationships as in Fig. 2(a), paper_4 could only be linked to and recommended to paper_1 after calculating the similarity between paper_1 and paper_3, and paper_2 and paper_4. It is obviously neither explicit nor intuitive.

In contrast, linking paper_4 to paper_1 can be easily inferred through network path routing methods. As shown in Fig. 3(b), two paths lead paper_1 to paper_4: (1) paper_1 $\xrightarrow{\text{cite}}$ dataset_x $\xrightarrow{\text{-cite}}$ paper_3 $\xrightarrow{\text{cite}}$ paper_4; (2) paper_1 $\xrightarrow{\text{cite}}$ paper_2 $\xrightarrow{\text{cite}}$ dataset_y $\xrightarrow{\text{-cite}}$ paper_4. Meanwhile, such multiple paths strengthen the connection possibility among the two entities in the network, if facing a recommendation. Furthermore, extracting the original internal attributes as first-class citizens in the network makes recommendation scalable. Using semantics-based analysis requires to build up rules for each type of attribute. For example, one has to establish rules to calculate the similarity between two papers based on their common citation on the same datasets, i.e., citing the same dataset_x by both paper_1 and paper_3. If another attribute (e.g., algorithm) is taken into consideration, new rules have to be added into the similarity calculation algorithm. On the contrary, if attributes are extracted as nodes, the same path finding algorithm can be used to navigate across the network thus to ensure scalability.

IV. Information Model

In this section, we discuss our information model establishment. As discussed in the last section, one core idea of our SKG is to create a multimodal network with fine-grained semantic concepts as first-class citizens, i.e., nodes. Recall that our goal is to extract and link valuable information from the vast data sources of existing knowledge. It is obviously critical to decide the granularity of the nodes, meaning an information model that defines key information and relationships in the Earth science domain.

Sponsored by Google, Microsoft etc, schema.org provides community-oriented shared vocabularies for structured data on the Internet. Its *actions* tag, however, provides a powerful facility for extensibility. However, schema.org is limited in terms of the objects it defines for scientific fields. Meanwhile, the data model of the Global Change Information System (GCIS, https://data.globalchange.gov/) describes the structure of information, including the overlaps between relational and semantic representations of the data. Being rich, however, it focuses on national climate change assessment reports only. Another related work is the book "Eloquent Science," where Schultz summarizes the structure of a science paper [11].

Learned from schema.org, GCIS data model, and Schultzs book, our strategy is to scrutinize how an Earth scientist reads a paper and identify reusable information components to be extracted from the literature. Working with Earth scientists closely, Fig. 4 summarizes how we extend the GCIS data model to establish the basic information model in SKG. For each paper, in addition to metadata (title, author, affiliations, contact information), we focus on what an Earth scientist wishes to learn: data, tools, methodology, hypotheses, and conclusions. A typical Earth science paper starts with some hypotheses. The paper then explains how the authors demonstrate their hypotheses over some specific datasets. Techniques, tools, algorithms, and workflows are explained in detail to explain how they conduct experiments and analytics. Findings and conclusions summarize the contributions of the paper.

Our team is formed with computer scientists, domain scientists, and graduate students in the domain. From many brainstorming and workshops, we found that the aforementioned semantic entities are what an Earth scientist usually extracts from reading a paper. Learning from this cognitive process, we use this information model as the underlying data model, where domain knowledge serves as schema to guide automatic information and extraction toward building the SKG.

V. Knowledge Discovery

Now that the underlying data model is established, an SKG can be constructed. Detailed methods of how we extract the semantic components from publications and incrementally build an SKG testbed are reported in our another paper [4]. In this article, we focus on explaining how we discover knowledge in our established SKG. We will first define semantic entity type and relationship type based on the information model defined in Section IV.

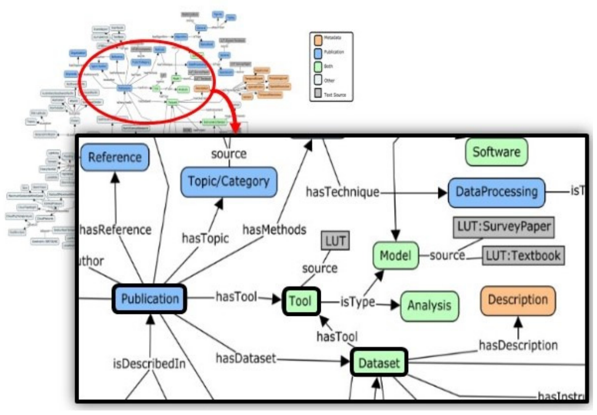


Fig. 4. Information model in SKG

A. Problem Formalization

Definition 1 (Science Knowledge Graph Building Blocks)**.** An entity type refers to one of the semantic entities $\{ET\}$ identified in the information model (in Section IV). A relation type refers to one of the relationships $\{R\}$ identified in the information model (in Section IV), or its inverse relationship $\{R^{-1}\}$. A relation R_x defines a transition starting from an entity type $l(R_x)$ and pointing to another entity type $r(R_x)$: $\forall R_x \in \{R\} \cup \{R^{-1}\}, \exists ET_l, ET_r \in ET \Rightarrow l(R_x) = ET_l \xrightarrow{R_x} ET_r = r(R_x), 1 \le l, r \le |ET|$.

In other words, we consider R_x and $ET_l \xrightarrow{R_x} ET_r$ equivalent, meaning that each relation implies a 3-tuple (original entity type, relation type, target entity type). With the building blocks defined, we can define a science knowledge graph.

Definition 2 (Science Knowledge Graph)**.** A science knowledge graph is a directed graph $SKG = G(V, E)$:

1) V denotes a collection of nodes where each node is an instance of an entity type: $\forall v \in V, \exists i \Rightarrow v = a(ET_i)$;
2) E denotes a collection of edges where each edge is an instance of a relation type: $\forall e \in E, \exists j \Rightarrow e = a(R_j)$;
3) $\forall e \in E, e^{-1} \in E$ denotes the inverse of the relation

According to the definition, each edge is an instance of a relation type: $\forall e = [v_l \xrightarrow{e} v_r] \in E, \exists R_x = [ET_l \xrightarrow{R_x} ET_r] \in R \cup R^{-1} \Rightarrow e = a(R_x), v_l = a(ET_l), v_r = a(ET_r)$.

Definition 3 (Path Pattern)**.** A path pattern P in an SKG is defined as a sequence of relation types concatenated. $P = R_1 R_2 ... R_L$ defines an L-length relation types chained together: $\forall 1 \le i < L, r(R_i) = l(R_{i+1})$. We also define the starting node type $l(P)$ and the ending node type $r(P)$ of a path pattern $P : l(P) = l(R_1); r(P) = r(R_L)$. A path traversal across an SKG thus is an instance of a path pattern over a starting node, in other words, implying a concatenation of relations instances applied to an entity instance. Each path p has a starting node and an ending node, represented as $l(p)$ and $r(p)$, respectively.

Given a starting node $v_1 \in V$, a path pattern P of length L over v_1 can be denoted as: $v_1 \xrightarrow{e_1} v_2 \xrightarrow{e_2} ... \xrightarrow{e_L} v_{L+1}$. It implies a sequence of relations consecutively operated over v_1: $v_1.a(P) = v_1.a(R_1).a(R_2). ... .a(R_L)$. Note that such an operation may result in a collection of different paths: $p = [v_1 \xrightarrow{R_1} v_2 \xrightarrow{R_2} ... \xrightarrow{R_L} v_{L+1}] \in v1.a(P)$. Also $l(p) = a(l(P))$ and $r(p) = a(r(P))$.

Definition 4 (Query in SKG)**.** A query in a science knowledge graph is a 4-tuple: $q =< I, S, T, R >$, where I denotes the intent of the query (typically in a collection of words), S denotes a collection of required entity types to start from, T denotes a collection of expected entity types to target on, and R denotes a collection of relation types that have to be used (i.e., included).

In other words, a query aims to explore some paths guided by intent and some relations, starting from some entity types and ending with some entity types. Consider the query of the motivating example in Fig. 3:

(Q1) Assume that a researcher desires to study on how to predict California wild fire, recommend papers for her to read.

The query suggests to explore the paths in the following pattern:

$P : \{Dataset = California\_wild\_fire\} \xrightarrow{\text{paper\_cite\_data}^{-1}} Paper \rightarrow ... \rightarrow Paper$

The starting entity type is $dataset$, and the starting nodes are instances of dataset related to California wild fire. The expected entity type is $Paper$. One relation type that has to be included is $paper\_cite\_data^{-1}$. The intent of the query is to find papers on fire prediction topic.

B. Path Finding Heuristics

In order to find paths for such a query $q =< I, S, T, R >$, our focus will be to find path patterns, meaning which types of path are good for solving the query. Similar to the Path Ranking Algorithm [8], we define a reward function for computing scores of the nodes covered along the paths under exploration.

Definition 5 (Node Intent and Path Context)**.** All latent topics in an SKG form an Intent space of T. The intent of a node v, denoted as $\phi_v = \{\phi_{1,v}, \phi_{2,v}, ..., \phi_{|T|,v}\}$, is a distribution of topics in the node's description over the Intent space of the network SKG, where the $|T|$-dimensional vector of probabilities sum to 1: $\sum_{i=1}^{|T|} \phi_{i,v} = 1$. The context of a path p is denoted as $\psi_p = \{\psi_{1,p}, \psi_{2,p}, ..., \psi_{|T|,p}\}$, as a union of the intent of all the nodes encompassed in the path. It can be calculated using a SoftMax function σ such that:

$$\psi_{j,p} = \sigma(\sum_{n \in p} \phi_{j,n}) = \frac{e^{\sum_{n\in p} \phi_{j,s}}}{\sum_{j=1}^{|T|} e^{\sum_{n \in p} \phi_{j,n}}} \tag{1}$$

where $\phi_{j,n}$ denotes the j^{th} intent value of node $n \in p$ and $\psi_{j,p}$ denotes the j^{th} intent value of path p.

Based on the definitions above, we can calculate the reward function of a query q, with a partial path p as an instance of path pattern P, with $p.V$ representing its encompassed nodes, to a candidate node c through a relation R toward an ultimate intent $q.I$:

$$\Omega(R(r(p), c)|q) = \alpha \cdot sim(\phi_c, \phi_{q.I}) + \beta \cdot sim(\psi_p, \phi_{q.I}) + \frac{\gamma}{|p.V - 1|} \tag{2}$$

Here, α, β, and γ are weights showing the importance of intent similarity of the candidate node and the query intent, context similarity of the path and the query, and the path length, respectively. These weights have values between 0 and 1 and also $\alpha + \beta + \gamma = 1$.

Based on the reward function of each edge defined above, we can compute the reward score of each node through a path exploration. Given a path pattern $P = R_1R_2...R_L$ and a set of starting nodes S_q ($\forall n \in S_q \Rightarrow n = a(S) = a(l(R_1))$), a reward function Ω for a node v is defined as $\Omega_{P,S_q}(v)$ recursively:

$$\Omega_{P,S_q}(v) = \sum_{v' \in r(P')} \Omega_{P',S_q}(v') \cdot \frac{\Omega(R_L(v', v)|q)}{|R_L(v')|} \tag{3}$$

The initialization of the reward function is when path pattern P has length zero:

$$\Omega_{P,S_q}(v) = \begin{cases} \frac{1}{|S_q|}, & \text{if } v \in S_q \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

C. Path Finding Implementation

In this section, we describe how we implement the path finding heuristics and recommend a list of nodes $\{v\}$ of type T from the SKG that can satisfy user's query. Algorithm 1 gets the whole SKG graph containing nodes V and edges E along with the user's query as its input. This algorithm returns sorted candidate nodes which can satisfy user's query, according to the probability of their occurrence in the future.

In Algorithm 1, first all candidate pairs of nodes are extracted from the graph in lines 5-20. For finding the candidate pairs, we find all the nodes from V that have a type of $q.S$ or $q.T$ as candidate source or target nodes, respectively. Then for every pair of source node and target node, we check if there exists a path from the source to target with length less that λ. All these pairs are considered candidate node pairs.

Then in lines 21-29, we assign a score to each candidate pair and prune those ones which do not have the edge types specified by $q.R$ along their paths. Therefore, for every pair, say $< v_s, v_t >$ in the candidate pairs list, we first find all paths from v_s to v_t which contain $q.R$ among the edges. If there exists such paths, then we find the shortest one from this set and assign a score to it based on Equation 5.

$$\Omega(p,c)|q) = \alpha \cdot \sum_{n_i \to n_j \in p} w_{ij} * sim(\phi_{n_j}, \phi_{q.I}) + \beta \cdot sim(\psi_p, \phi_{q.I}) + \frac{\gamma}{|p.V - 1|} \tag{5}$$

Note that Equation 5 is an extension of Equation 2 defined as the reward function of a path p, as an instance of path pattern P, reaching to a candidate node c for query q. Specially, the weight of an edge (i.e., every segment of a path) is taken into consideration. In this function, w_{ij} denotes the weight of every edge along the path p and ϕ_{n_j} is the intent of every target node in this path.

Finally, in line 30, the candidate pairs are sorted in descending order based on their computed scores. The target nodes $\{v_t\}$ from this sorted list are then returned as the result of the algorithm.

Algorithm 1 Recommend Nodes

```
1:  procedure RecommendNodes(SKG, $q$)
2:      $candidateSourceNodes \leftarrow \emptyset$
3:      $candidateTargetNodes \leftarrow \emptyset$
4:      $candidatePairs \leftarrow \emptyset$
5:      for $v \in V$ do
6:          if $v.ET \in q.S$ then
7:              $candidateSourceNodes \leftarrow candidateSourceNodes + v$
8:          else
9:              if $v.ET \in q.T$ then
10:                 $candidateTargetNodes \leftarrow candidateTargetNodes + v$
11:             end if
12:         end if
13:     end for
14:     for $v_s \in candidateSourceNodes$ do
15:         for $v_t \in candidateTargetNodes$ do
16:             if there is path from $v_s$ to $v_t$ with length less than $\lambda$ then
17:                 $candidatePairs \leftarrow candidatePairs + < v_s, v_t >$
18:             end if
19:         end for
20:     end for
21:     $nodePairsWithScores \leftarrow \emptyset$
22:     for $< v_s, v_t > \in candidatePairs$ do
23:         $pathSet \leftarrow$ Paths from $v_s$ to $v_t$ containing $q.R$
24:         if $pathSet \neq \emptyset$ then
25:             $path \leftarrow$ shortest path from $pathSet$
26:             $score \leftarrow calculateScore(path)$
27:             $nodePairsWithScores \leftarrow nodePairsWithScores + << v_s, v_t >, score >$
28:         end if
29:     end for
30:     $\{v_t\} \leftarrow$sort $nodePairsWithScores$ based on the scores in descending order
31:     return $\{v_t\}$                              ▷ Sorted List of Recommended Nodes
32: end procedure
```

VI. Topological Structural Analysis

In Equation 3, the reward score of a node can be viewed as an accumulation of the scores of all edge segments of the paths pointing to the node. Equation 2 shows that the reward score of an edge segment is computed based on the intent of the target node, against the intent of the query and that of the starting node. In other words, the reward function is calculated based on the paths navigated. Recall in motivating example 1 in Fig. 2, paper 2 and paper 4 share many similar neighbors. Such topological similarity could suggest the similarity between the two nodes. Since such a sharing implies many possible paths that yield higher rewards, we believe more weights should be explicitly added into the calculation.

The structure of a node refers to the topological structure of a sub-graph rooted by the node. Thus, similarity comparison among nodes can be turned into similarity comparison among the sub-graphs in the SKG. A number of algorithms are available to measure the similarity between graphs. For example, the edit distance-based [3] approach assigns a cost value to every operation of transforming one graph to the other, and attempts to identify such a sequence of operations with a minimum cost. For another example, Fernandez and Valiente [5] measure the similarity between two graphs based on their maximum common sub-graph and minimum common supergraph.

Recall that our method of extracting encapsulated features from nodes and edges significantly increases the complexity of the knowledge network. Calculating the structural similarity among many sub-graphs at runtime raises significant scalability concern. Therefore, we apply deep learning technique to compute the topological similarity among nodes offline.

A. *SDAE-based Entity Structure Learning*

Among others, Stacked Denoising Auto-Encoders (SDAE) is a supervised deep learning model for translating a stacked denoising autoencoder into a composition of denoising autoencoders in the ground space [14]. SDAE is typically used to learn hidden feature vectors of images. In this project, we adopt SDAE to learn latent feature vectors of nodes in an SKG, based on their topological structure.

Fig. 5 illustrates our basic idea, where a 4-layer SDAE is used. For each node in SKG, we quantify its topological structure into an embedding vector (e.g., a feature matrix) based on its connections to other nodes in the graph. Aggregating the embedding vectors of all comprising nodes, an SKG receives an $X = N \times N$ embedding matrix where N denotes the number of nodes in the SKG. Applying Gaussian noise to a node i, the noised embedding vector ($\widetilde{X}_i^{(0)}$) acts as the input layer to the network. Vector $\widetilde{X}_i^{(4)}$ of the output layer is the reconstruction of the input layer. The first half of the layers ($\widetilde{X}_i^{(1)}$ and $\widetilde{X}_i^{(2)}$) serve as encoders, and the second half of the layers ($\widetilde{X}_i^{(3)}$ and $\widetilde{X}_i^{(4)}$) as decoders. The generated hidden layer ($\widetilde{X}_i^{(2)}$) represents a lower-dimension feature vector for the node. The same encoding-decoding process is applied to all nodes as the training process. The resulting bottleneck hidden layer ($\widetilde{X}^{(2)}$) is considered the latent feature matrix of the SKG.

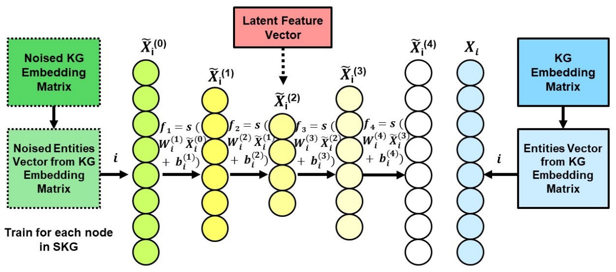

Latent Feature Vector
Noised KG Embedding Matrix
Noised Entities Vector from KG Embedding Matrix
Train for each node in SKG
$\widetilde{X}_i^{(0)}$
$\widetilde{X}_i^{(1)}$
$\widetilde{X}_i^{(2)}$
$\widetilde{X}_i^{(3)}$
$\widetilde{X}_i^{(4)}$
X_i
$f_1 = s(W_i^{(1)} \widetilde{X}_i^{(0)} + b_i^{(1)})$
$f_2 = s(W_i^{(2)} \widetilde{X}_i^{(1)} + b_i^{(2)})$
$f_3 = s(W_i^{(3)} \widetilde{X}_i^{(2)} + b_i^{(3)})$
$f_4 = s(W_i^{(4)} \widetilde{X}_i^{(3)} + b_i^{(4)})$
i
i
KG Embedding Matrix
Entities Vector from KG Embedding Matrix

Fig. 5. One iteration of model training

B. *Pre-training*

An SDAE model training may result in local optimization instead of global one. Therefore, an unsupervised pre-training process is applied. Let the number of layers be L, the number of latent features for layer $L/2$ as n. Let X_i be the SDAE input (i.e., clean) vector from the Embedding Matrix for each node. Let noised input be $\widetilde{X}_i$. Let W_i and b_i be the weight parameter and bias parameter for the SDAE network, respectively.

As shown in Fig. 5, for layer j we have either an encode function or a decode function:

$$f_j = s(W_i^{(j)}\widetilde{X}_i^{(j-1)} + b_i^{(j)})$$

The learning objective function is defined as:

$$\min_{(W_i,b_i)} ||X_i - \widetilde{X}_i||_F^2 + \lambda \sum_{j\in L} ||W_i^{(j)}||_F^2$$

where λ is a regularization parameter and $||.||_F$ denotes the Frobenius norm. To minimize the objective function, stochastic gradient descent is applied, similar to [15].

In our model, the above SDAE process is trained for each node in SKG, individually. Fig. 5 illustrates a simplified example with an SDAE with four layers. The topological structure of a node is transformed into a feature vector (e.g., with eight features). By adding Gaussian noise, the noised entity vector acts as input to the SDAE on one end, while the clean vector acts as input to the SDAE on the other end. Running a round of pre-training, a hidden layer results in a six-feature vector, with weight $W^{(1)}$ and bias $b^{(1)}$.

VII. Scalability Control

Recall that we advocate to extract many encapsulated node features and edge features to become first-class citizen nodes in our knowledge graph. Apparently it is necessary to study the scalability issue due to the increase of the number of nodes, the order of the graph, the types of edges, and the number of edges. Our strategy is to seamlessly integrate the state-of-the-art algorithm on matrix embeddings scaling (i.e., TransE) into our path exploration algorithm described in Section V.

A. *Introduction to TransE*

TransE [2] is a machine learning-based method dealing with comprehensive knowledge graph, where edges are modeled as translations (in the form of vector embedding) operating on vector embeddings of the nodes. Its goal is to learn low-dimension vectors for the vector embedding of every node and every type of edge. As shown in Fig. 6(a), an edge e between two nodes v_s and v_f is represented as an edge vector $\overrightarrow{e}$ linking two nodes' vectors $\overrightarrow{v_s}$ and $\overrightarrow{v_f}$: $\overrightarrow{v_s} + \overrightarrow{e} = \overrightarrow{v_f}$. Through a learning process, all vectors are turned into low-dimension k vector embeddings: $e, v_s, v_f \in R^k$: $v_s + e \approx v_f$.

The learning process can be summarized as below. Let node set be denoted as V, edge set as E, and node-edge-node triplet set as T. For each pair of triplet $(v_s, e, v_f) \in T$, construct its corrupted triplets: (v'_s, e, v_f) and (v_s, e, v'_f) by replacing its left-hand side or right-hand side node with a random node, respectively. All corrupted triplets form a set T':

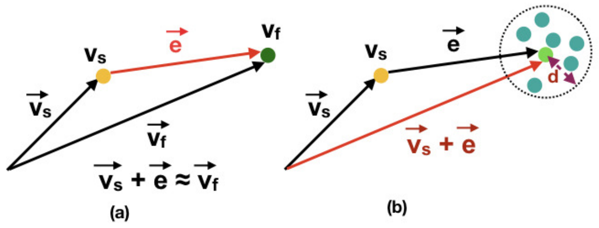


Fig. 6. transE-based KG navigation

$$T'_{(v_s,e,v_f)} = \{(v'_s, e, v_f)|v'_s \in N\} \cup \{(v_s, e, v'_f|v'_f \in N)\}$$

Then the learning process can be represented by minimizing the following margin-based objective function over the training set:

$$min \sum_{(v_s,e,v_f)\in T} \sum_{(v'_s,e,v'_f)\in T'} [\gamma + d(v_s+e, v_f)) - d(v'_s+e, v'_f)]_+$$

where $[x]_+$ indicates the positive part of x, $\gamma > 0$ is a margin hyperparameter, and d is a dissimilarity function.

B. *TransE-powered Path Exploration*

Based on TransE, we revise our path exploration algorithm to address scalability. The TransE algorithm is first applied, offline, to obtain low-dimensional vector embeddings for each node and type of edge in the SKG. Fig. 6(b) illustrates an example. Given a node v_s and a relation type e, we can obtain a targeting node $\overrightarrow{v_t} = \overrightarrow{v_s} + \overrightarrow{e}$. As shown in Fig. 6(b), the targeting node points to a virtual node not existing in the network; however, it is surrounded by some nodes. Centered on the virtual node, a circle with a small diameter d will encompass a collection of nodes quite close to it. In other words, we aim to find nodes with minimal distances from the virtual node:

$$\min_{v_j} ||\overrightarrow{v_j}, \overrightarrow{v_s} + \overrightarrow{e}||^2 \le d^2$$

In addition to distance, we use the query-specific intent and context information to guide the selection. Be more specific, we tend to select the candidate node favoring the query intent and contextual requirements the most. The selection criteria thus is refined as below:

$$\max_{||\overrightarrow{v_j}, \overrightarrow{v_s}+\overrightarrow{e}||^2 \le d^2} similarity_{(intent,context)}(v_j, v_s)$$

So far we have discussed how to move one step over an edge in the SKG. Finding a path in the SKG can be represented by a sequence of steps.

VIII. Experiments

In this section, we introduce our experiments and discuss the findings.

A. *Experiment #1*

The first testbed is the collection of all workflows accumulated by myExperiment.org, the largest public repository of scientific workflows. Until May 2018, the testbed contains more than 4,200 workflows. We focus on Taverna workflows comprising web services, since we can write code to analyze their structure. Crawling for the publicly available Taverna workflows provided us with 3,277 workflows from 2,030 unique ones. After finding those workflows, we extracted all WSDL, SoapLab, BioMoby, and SADI services along with the REST calls and found 513 unique web services. Following the data links between services in every workflow helped us create a network with these 513 nodes. Table 1 shows a summary of our testbed.

TABLE I
SUMMARY OF EXPERIMENTS

A	# Unique workflows	2,030	
	# Workflow versions	3,277	
	# Unique web services	513	
	# Unique service operations	1,248	
	# Unique directed edges in network	326	
	# Pairs of nodes with directed paths in network	664	
	# Pairs of nodes with indirected paths in network	2,484	
B	**With Local Services as First-class Citizen**		**Improvement**
	# Unique web services	573	$1.117\times$
	# Unique directed edges in network	1,489	$4.567\times$
	# Pairs of nodes with directed paths in network	47,230	$71.130\times$
	# Pairs of nodes with indirected paths in network	95,870	$38.595\times$

The baseline network is a graph $< V, E >$, where V denotes all web services contained in all workflows. There is only one edge type in this network - $followedBy$. If two services are connected in a workflow, there is an edge linking the two service nodes in the network. Each edge carries two features: one is the workflow where this connection happens; the other one is the possible local service in between the connection. *Local services* are known utility services that are carried by Taverna as reusable components. As summarized in Table 1 part A, this baseline network contains 513 nodes and 326 edges. Directed paths among services are 664, meaning that such paths can be used directly. Undirected paths for recommendation among services are 2,484.

We then extracted one edge feature out as a first-class citizen, that is local service. All local services were extracted from the edges and became independent nodes in the network. For example, consider an initial edge ($s_1 \xrightarrow{\text{followedBy}} s_2$) linking two services nodes s_1 and s_2 through an intermediate local service ls_3. By extracting the local service as a node, we now have three nodes with two more $followedBy$ edges: $s_1 \xrightarrow{\text{followedBy}} ls_3$ and $ls_3 \xrightarrow{\text{followedBy}} s_2$. As summarized in Table 1 part B, for the entire network, the number of nodes increases by 11.7% to 573, the number of directed paths increases 71 times to 47,230, and the number of indirected paths increases 38 times to 95,870. This means that many more possible software (i.e., service) collaborations may be recommended through the extracted 60 local services involved.

We then evaluated the recall of our method. Started from the network just built with all web services and local services as nodes, all edges were removed in the network and marked at time t_0. All workflows were sorted based on their publication timestamps, representing the total $M = 716$ time points to study the network. Following the order from the earliest timestamp, $followedBy$ edges (with timestamp as a feature) were added into the network among the corresponding web services and local services nodes. Before an edge was added between a pair of services nodes, we checked whether there existed a path between them, with each segment being marked by some earlier timestamp. If such a path exists, it means such a connection could be predicted by our approach. For example, assume at time t_x, a $followedBy$ edge is to be added from s_i to s_j labeled by t_x. If there exists a path $s_i \xrightarrow{t_{y_1}} k_1 \xrightarrow{t_{y_2}} ...k_{l-1} \xrightarrow{t_{y_l}} s_j$, where $y_o < x, 1 \leq o \leq l$, and $k_m \in N, 1 \leq m < l-1$, it means such a connection could be predicted by our approach. The recall of our method is reported in table II. For example, in 2008, a total of four new edges were added in the network. Our method could have predicted two edges, leaving a recall of 50%. As shown in Table II, in most of the years, our method could have predicted quite a few service connections, while the total average recall being 42%.

TABLE II
AVERAGE RECALL OF THE ALGORITHM FOR DIFFERENT YEARS

Year	Total # new edges	Total # predictable edges	Average recall
2008	4	2	0.5
2009	4	2	0.5
2010	1	0	0
2011	17	15	0.88
2012	10	4	0.4
2013	8	4	0.5
2014	2	0	0
2015	2	2	1
2016	1	0	0

B. Experiment #2

We also studied the feasibility of leveraging topological structure to facilitate software recommendation. Performance is the major concern for comprehensive structural analysis, thus we applied the SDAE algorithm as discussed in the earlier Section VI.

Using the method described in Section VI, we gave our SKG adjacency matrix as the input for the SDAE algorithm, considering every node's adjacency vector as a mini batch for this algorithm. Then after the pre-training phase, we used latent vectors created by SDAE for each node in SKG to compare the structural similarity of the nodes. For running the SDAE algorithm, we used the Sigmoid function and two hidden layers of size 20. Also, the pre-training epochs, pre-training learning rate, and corruption level were considered 1000, 0.2, and 0.3, respectively.

Then, we evaluated the correlation of the structural difference of SKG nodes and the difference of latent vectors created by SDAE. This difference was calculated based on the Euclidean distance of the adjacency vectors for each pair of nodes and also the Euclidean distance of latent vectors for every pair of nodes. Fig 7 shows the results of our evaluation. The relationship between the two metrics appears nonlinear. There is little correlation when the structural difference between SKG nodes is less than 4. It almost looks like the latent vector difference decreases for higher values of SKG node difference. Meanwhile, the relationship does not seem to scale when the difference between SKG nodes is greater than 4. The average correlation of these two metrics in our dataset was 0.49. In other words, the latent vectors created by the SDAE method can well reflect the structural similarity of pairs of nodes in SKG. This experiment proved that applying offline SDAE method could help to address the scalability issue of comparing structural similarity among SKG nodes at runtime.

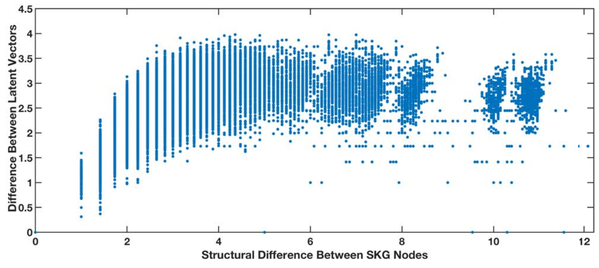


Fig. 7. Correlation of structural difference of SKG nodes and the difference between latent vectors

C. Experiment #3

In the third experiment, we evaluated the TransE-powered path exploration method. In order to evaluate this, we used the transE code provided by Lin et al. [9] and gave the baseline network and a size as the inputs for their algorithm. This method provides us with vectors for every node in the network and another vector for the only edge type we have. The size of vectors were specified as the input and for our evaluation we used $size = \{50, 75, 100, 150, 200, 250, 300, 350, 400, 450, 500\}$.

For every edge $e : v \rightarrow v'$ in the baseline network, we calculated the Euclidean distance of the two vectors $\vec{v} + \vec{e}$ and $\vec{v''}$ where v'' is every node in the network that is not equal to v. Then we sorted the distances calculated for every node and we obtained the rank of v' in the list.

We then evaluated the effect of pruning nodes based on their intent similarity on the rankings of the target nodes of every edge. In order to do this, for every source node v and for every edge e, we decided the intent space of the candidate target nodes by computing the difference between the intent of the target workflow and that of the source node. Then from the candidate target nodes resulted from the TransE calculation, we pruned the ones whose distances are higher than 0.41 or whose similarity with the intent space less than 0.01. Then again we sorted the distances and obtained the ranking of the target nodes for every edge.

Fig. 8 shows the average ranking of the target nodes for every edge in the network for various sizes of transE vectors. This figure shows the results for Test 1 in which we did not consider the intent similarity and Test 2 in which we considered intent similarities. As can be seen in this figure, the best vector size for our testbed is 50 for which we have the average ranking of 2.54 for Test 1 and 2.08 for Test 2. This figure also shows that using intent similarity can increase our recommendation accuracy for 67%.

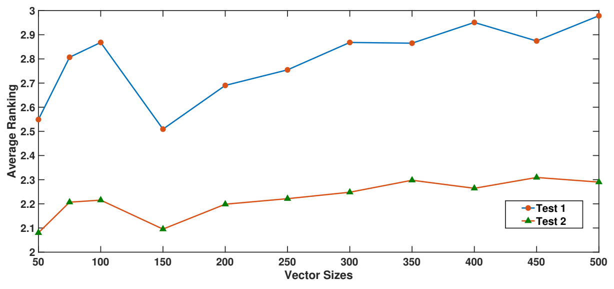


Fig. 8. Average ranking of target nodes for every edge by using transE-powered path exploration method

IX. Conclusions

Knowledge graph has become an important way to support knowledge sharing and discovery. Aiming to build a science knowledge graph, this paper proposes an information model rooted in cognitive analysis of how researchers write and read papers. Based on the information model, a novel way of building science knowledge graph is presented. Semantic entities are extracted as first-class citizens instead of acting as features encapsulated in nodes and edges. Complementary to adopting existing semantic web technique to answer community-oriented user queries, we have developed a technique that counts on topological structural analysis to gain higher scalability at runtime. Semantic information can be leveraged offline to enrich the knowledge network. Deep learning techniques are applied to explore hidden and potential connections.

In our future work, we plan to further study how to address the performance and scalability issues in an SKG. For example, how to fine tune the SDAE algorithm as a lack of labels. In addition, we plan to build a rich SKG from extracting the semantic entities from literature based on our information model. We plan to study how it could effectively respond to community-oriented queries. Furthermore, we plan to expand our SKG technique to other Earth science areas beyond Hurricane.

Acknowledgment

This work is partially supported by National Aeronautics and Space Administration under grant NNX16AE15G, and National Science Foundation under grant ACI-1443069.

References

[1] Q. Bao, J. Zhang, X. Duan, R. Ramachandran, T.J. Lee, Y. Zhang, Y. Xu, S. Lee, L. Pan, P. Gatlin, and M. Maskey, ”A Fine-Grained API Link Prediction Approach Supporting Mashup Recommendation,” in Proceedings of The 24th IEEE International Conference on Web Services (ICWS), 2017, pp. 220-228.

[2] A. Bordes, N. Usunier, A. Garcia-Durn, J. Weston, and O. Yakhnenko, ”Translating Embeddings for Modeling Multi-Relational Data,” in Proceedings of The 26th International Conference on Neural Information Processing Systems (NIPS), Vol. 2. 2013, pp. 2787-2795.

[3] H. Bunke, ”Error Correcting Graph Matching: on the Influence of the Underlying Cost Function,” IEEE Transactions on Pattern Analysis and Machine Intelligence, 21(9), Sep. 1999, pp. 917922.

[4] X. Duan, J. Zhang, R. Ramachandran, P. Gatlin, M. Maskey, J.J. Miller, K. Bugbee and T.J. Lee, ”A Neural Network-Powered Cognitive Method of Identifying Semantic Entities in Earth Science Papers,” in Proceedings of IEEE International Conference on Cognitive Computing (ICCC), pp. 9-16.

[5] M.-L. Fernndez and G. Valiente, ”A Graph Distance Metric Combining Maximum Common Subgraph and Minimum Common Supergraph,” Pattern Recognittion Letters, 22(67), 2001, pp. 753758.

[6] D.A. Ferrucci, ”Introduction to ’This is Watson’,” IBM Journal of Research and Development, 56(3.4), 2012, pp. 1-15.

[7] A. Labrinidis and H.V. Jagadish, ”Challenges and Opportunities with Big Data,” in Proceedings of VLDB Endowment, 5(12), Aug. 2012, pp. 2032-2033.

[8] N. Lao and W.W. Cohen, ”Fast Query Execution for Retrieval Models based on Path-Constrained Random Walks,” in Proceedings of The 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2010, pp. 881-888.

[9] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, ”Learning Entity and Relation Embeddings for Knowledge Graph Completion,” in Proceedings of The 29th AAAI Conference on Artificial Intelligence, 15, 2015, pp. 2181-2187.

[10] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich, ”A Review of Relational Machine Learning for Knowledge Graphs,” Proceedings of the IEEE, 2016, 104(1), pp. 11-33.

[11] D. Schultz, Eloquent Science: A Practical Guide to Becoming a Better Writer, Speaker, and Atmospheric Scientist, American Meteorological Society, 2009.

[12] A. Sinha, Z. Shen, Y. Song, H. Ma, D. Eide, B.-J. (Paul) Hsu, and K. Wang, ”An Overview of Microsoft Academic Service (MAS) and Applications,” in Proceedings of The 24th International Conference on World Wide Web (WWW), 2015, pp. 243-246.

[13] F. Suppe, ”The Structure of a Scientific Paper,” Philosophy of Science, 65(3), 1998, pp. 381-405.

[14] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, ”Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion,” The Journal of Machine Learning Research, 11, 2010, pp. 33713408.

[15] H. Wang, N. Wang, and D.-Y. Yeung, ”Collaborative Deep Learning for Recommender Systems,” in Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2015, pp. 1235-1244.

[16] C. Zhang, V. Govindaraju, J. Borchardt, T. Foltz, C. R, and S. Peters, ”GeoDeepDive: Statistical Inference using Familiar Data-Processing Languages,” in Proceedings of The ACM SIGMOD International Conference on Management of Data (SIGMOD), 2013, pp. 993-996.

[17] C. Zhang, C. R, M. Cafarella, C. De Sa, A. Ratner, J. Shin, F. Wang, and S. Wu, ”Deepdive: Declarative Knowledge Base Construction,” Communications of the ACM, 60(5), May 2017, pp. 93-102.

[18] J. Zhang, W. Wang, X. Wei, C. Lee, S. Lee, L. Pan, and T.J. Lee, ”Climate Analytics Workflow Recommendation as a Service - Provenance-Driven Automatic Workflow Mashup, in Proceedings of The 22nd IEEE International Conference on Web Services (ICWS), 2015, pp. 89-97.

[19] J. Zhang, M. Pourreza, S. Lee, R. Nemani, and T.J. Lee, ”Unit of Work Supporting Generative Scientific Workflow Recommendation,” accepted by International Conference on Service Oriented Computing (ICSOC), 2018.