OpenKN: An Open Knowledge Computational Engine for Network Big Data

Yantao Jia, Yuanzhuo Wang, Xueqi Cheng, Xiaolong Jin and Jiafeng Guo
CAS Key Laboratory of Network Data Science and Technology
Institute of Computing Technology
Chinese Academy of Sciences
Beijing, P. R. China
jiayantao, wangyuanzhuo, cxq, jinxiaolong, guojiafeng@ict.ac.cn

***Abstract*—With the coming of the era of big data, it is most urgent to establish the knowledge computational engine for the purpose of discovering implicit and valuable knowledge from the huge, rapidly dynamic, and complex network data. In this paper, we first survey the mainstream knowledge computational engines from four aspects and point out their deficiency. To cover these shortages, we propose the open knowledge network (OpenKN), which is a self-adaptive and evolutionable knowledge computational engine for network big data. To the best of our knowledge, this is the first work of designing the end-to-end and holistic knowledge processing pipeline in regard with the network big data. Moreover, to capture the evolutionable computing capability of OpenKN, we present the evolutionable knowledge network for knowledge representation. A case study demonstrates the effectiveness of the evolutionable computing of OpenKN.**

***Keywords*-open knowledge network; knowledge computational engine; evolutionable knowledge network; evolutionable computing**

I. Introduction

Network big data refers to the massive data generated in the cyberspace and available on the Internet. It has a few typical features, such as multi-sourced, heterogeneous, interactive, bursty, and noisy. More formally, the features of big data can be defined in four dimensions, or 4Vs: volume (amount of data), variety (range of data types and sources), velocity (speed of data generation and manipulation in and out), veracity (integrity of data for users to trust it). In addition to the 4Vs, researchers also added a fifth V, namely Value, which refers to the usefulness or importance of data.

Network big data implicitly contains tremendous highly-interconnected knowledge. Building up knowledge computational engine is an effective means for mining rich knowledge. Generally, a knowledge computational engine is an end-to-end and holistic knowledge processing pipeline containing the initial data crawling, knowledge extraction and maintenance, knowledge understanding and computation, and the final knowledge application and service. As a universal infrastructure of knowledge computational engine, the web oriented knowledge bases mainly consisting of concepts, instances and relations have been extensively studied and built in recent years. Typical examples include KnowItAll [1], NELL [2], Freebase [3], YAGO [4], [5], ConceptNet[1], DBpedia [6], Probase [7], Google's knowledge graph [8] and so on. With the successful applications in semantic search, automatic question and answering, digital reading, etc, more and more knowledge computation methodologies based on the knowledge bases, such as the intent-aware query similarity calculation [9] and the link inference techniques [10] are established so as to make the knowledge computation engine a full-fledged life cycle. Well-known examples of the knowledge computational engines include Wolframalpha[2], NELL, Google's knowledge graph, Probase, KB in WalmartLabs [11], etc.

However, due to the 5Vs of the network big data, there is still room for improvement of current knowledge computational engines to systematically understand the intrinsic semantics in the network big data. Firstly, most knowledge computational engines are set up in a coarse and error-prone manner by disregarding the diversity and ambiguity of network big data. For example, when extracting the is-a relationship between concepts and instances, state-of-the-art Hearst pattern based syntactic or semantic iteration methods always sacrifice recall for precision owing to the stiff extraction rules. This leads to the missing of a considerable fine-grained concepts or relations, and furthermore affects the subsequent knowledge computation task. Secondly, it still lacks an efficient way to capture and discover newly emerging instances. For example, many knowledge bases do not consider appropriate incremental update strategies for timely update in accordance with the velocity of the network big data. By carefully analyzing the core techniques and methods of current knowledge bases from four aspects, we discover the deficiency in regard with the 5Vs of network big data. To cover these shortages, we introduce the knowledge computational engine for network big data, the Open Knowledge Network (i.e. OpenKN). Furthermore, we demonstrate its merits for knowledge discovery. Specifically, the contributions of our work are two-fold:

1. We propose a self-adaptive knowledge computational engine for network big data, i.e., the OpenKN. The self-adaptiveness of OpenKN enables the engine to incrementally

[1]http://conceptnet5.media.mit.edu/
[2]http://www.wolframalpha.com

update itself by capturing newly emerging instances and concepts over time, and have the capability of evolutionable computing.
2. To characterize the evolutionable computing of OpenKN, we present the evolutionable knowledge network, which is a heterogeneous network with nodes and edges anchored with the temporal and spatial information as well as their correlation. A case is studied to show the effectiveness of the evolutionable computing of OpenKN.

The remainder of this paper is organized as follows. In Section II, we briefly survey the construction techniques of the mainstream knowledge computational engines. In Section III, we introduce OpenKN and its architecture, demonstrate its two salient features. A case study is described to show the effectiveness of OpenKN in the scenario of relation inference in Section IV. Finally, the paper is concluded in Section V.

II. The Mainstream Knowledge Computational Engines

In this section, we survey the existing techniques related to the knowledge computational engine from the aspect of its important component, i.e. the knowledge base. Specifically, we unfold the story from four aspects: the construction of knowledge bases from various data sources, the integration of external knowledge to enlarge the knowledge bases, the update of knowledge bases, and various applications of knowledge bases. Each aspect is concluded by one or two limitations with regard to the 5Vs of network big data.

A. The construction of knowledge bases

To build a web knowledge base, one needs to extract the instances or entities, concepts or classes, and the relations from the web. Generally, this can be obtained by information extraction techniques from the web. Early techniques, due to the particular domain and limited volume of data, depend mostly on human annotation and expertise. This needs costly efforts but leads to a relatively high precise of the extraction results. In this sense, automatic methods are proposed which enable machines to understand the web texts. We mainly discuss the automatic information extraction techniques in this paper.

The automatic extraction methods of building a knowledge base generally consist of two types: the syntactic iteration method and the semantic iteration method [7]. Former examples include KnowItAll [1], TextRunner [12], [13] and NELL [2], which adopt the so called bootstrapping strategy by starting with a predefined syntactic pattern or seed, and then generating new patterns to guide further extraction. The syntactic patterns are usually of high quality, such as the Hearst patterns, which seem to be very rare in large scale web texts. Meanwhile, due to the ambiguity of language, the syntactic method is limited to understand the deep semantic differences in texts, especially in short texts of microblogs. Therefore, the semantic-level iteration techniques emerged. Typical example is Probase [7], a probabilistic knowledge base owing to Microsoft. The iteration also uses a fixed set of syntactic patterns, but relies on existing background knowledge in a probabilistic manner to understand the text semantically. For example, for the sentence "animals other than dogs such as cats", the syntactic iteration will extract (cat isA dog), which the semantic iteration obtain the correct result (cat isA animal), because it detected probabilistically that the likelihood of animals given the cats is much higher than the likelihood of dogs given the cats. This intuition makes sense since it is much less likely for sentences like "dogs such as cats" than those like "animals such as cats".

The automatic iteration methods, i.e. both the syntactic and the semantic ones, still have limitations toward the velocity and veracity of network big data. On one hand, the veracity of the data makes the template-based methods invalid in the newly-established context in which no syntactic or semantic patterns have been predefined, or the predefined patterns cause semantic drift [14]. This is especially true for short texts, such as tweets, Facebook posts, etc. For more detail, see [15]. On the other hand, exactly identifying entities becomes harder due to the issue of ambiguity throughout the texts. Traditional techniques like the named entity disambiguation methods (e.g. methods based on the context similarity), are not effectively applicable because of the scarcity of critical context and evidence around the web.

B. Web-scale integration of various knowledge bases

The web-scale integration of different knowledge bases can be formalized into many issues, including ontology matching, ontology mapping, ontology integration, etc. These issues mainly have the roots of two identifiers, i.e., identifying duplicate entities or instances, and identifying similar classes.

The instance identifier, also known as entity co-reference resolution, identity uncertainty and record linkage, has been extensively studied in duplication elimination, object consolidation and other areas. The naive method is based on string similarity, such as the edit distance between strings. Afterwards, the machine learning techniques are integrated by adding domain knowledge into the process of similarity computation. These two methods have the limitation on scalability over large data and across different domains. Recent progress relies on the context of the entities, such as entity attributes, relations, or the natural language texts around them. For example, given two entities, we can find Wikipedia articles of themselves or their synonyms, and then compute the similarity of the vector created by the bag-of-words collection from the Wikipedia articles [16]. However, these context-based methods are again unrealistic in real data because of the little contextual information.

The class identifier aims to match and merge two classes in the knowledge bases. Most related work tries to quantify the similarity between the set of instances of the two classes, while others incorporate more criterions other than the instance-based methods. For example, in Probase, two classes are matched by measuring the similarity between the sets of their attributes. Although much work has been done on the integration of knowledge bases, there still exists much room for improvement. The heterogeneity of different ontologies and knowledge sources (the veracity of big data) makes the integration a tough task, in spite of the utility of some universal languages, e.g., RDF, OWL during the knowledge representation procedure. Moreover, most approaches addressing the integration are only examined and applied on small-scale knowledge bases [17]. An efficient and scalable integration framework harnessing trillions of entities and concepts is still an active research area in the big data era.

C. Web-scale update of knowledge bases

The update of knowledge bases is very important due to the dynamics of knowledge in daily life. The update of knowledge base includes updating the entities, concepts, relations, attributes, and other auxiliary information. However, there is little literature on this topic. Most knowledge bases adopt a reconstruction scheme on a relatively small data set. Even they considered the issue of incremental update, the scope is intensive and conservative. For example, recently Deshpande et al. [11] proposed an update strategy by reconstructing almost the entire knowledge base and considering certain simple incremental rules for the rest knowledge base. They used parallel processing to minimize the time in the entire reconstruction process and employed a data analyst to curate the knowledge base during the process. Similar examples include NELL [2], which fulfilled the update by the "never-ending" and "self-correct" techniques. Specifically, it retrained over time in a self-supervised manner to continue knowledge acquisition, and when errors occurred, experts are introduced for the maintenance and curation every few weeks.

As is stated above, the update of knowledge base is also confronted with challenges. For instance, the velocity of data forces the knowledge base to detect the changes, assess and validate the updates, and manage them. Moreover, an incremental update strategy with minimal cost is highly preferred.

D. Various applications of knowledge bases

Among the various applications of knowledge bases, we elaborate the relation inference task between entities in knowledge bases. For other applications, readers can refer to [15]. Generally, there is plenty of literature on relation inference in knowledge bases. Early research mostly depended on the classical logical rules, such as the first-order Horn clauses, the FOIL algorithm and its variant nFOIL. These rules are hand-crafted at the beginning, and then unsupervised learning method is designed, such as the Sherlock system [18]. To overcome the low efficiency and limited inference space, the probabilistic ingredient is integrated. The Holmes system [19] uses the inference rules to construct a forest of proof trees, and then employs the approximate probabilistic inference of Markov logic networks. Aside from these syntactic inference rules, weighted rules, and syntactic-semantic rules have been proposed and proven to be more flexible. For example, Lao et al. [20] established the extended path ranking algorithm combining both semantic and syntactic information encoded as relations in the knowledge base.

The relation inference in knowledge base still meets many opportunities. For instance, the massive volume of the knowledge base makes training computationally expensive. Although some speed-up techniques, such as Map-Reduce distribution framework, stratified sampling or piecewise indexed skills, are introduced, it is still not easy to say how much it really works quantitatively, e.g., to what extent the techniques can improve inference.

III. OpenKN

In this section, we shall introduce the knowledge computational engine OpenKN and its two salient features. Firstly, the architecture of OpenKN is depicted in Fig. 1. From Fig. 1, we can see that the four components of

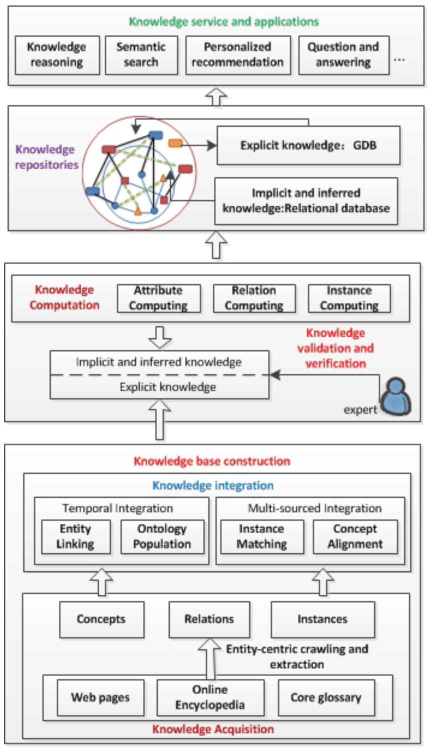

Figure 1. The architecture of OpenKN

OpenKN, i.e., the knowledge base construction, the knowledge validation and verification as well as the knowledge computation, the knowledge repositories, and the knowledge service and applications are listed in a bottom-up manner. These components provide an end-to-end knowledge processing workflow from the initial knowledge acquisition, knowledge integration, knowledge computation, knowledge validation and verification, knowledge repositories, and the final knowledge service and applications. We shall elaborate the first three components in detail respectively.

Firstly, the knowledge base construction of OpenKN is made up of two phrases in logical viewpoint, that is, the knowledge acquisition phrase and the knowledge integration phrase. The former phrase aims to extract concepts, entities or instances, relations from web pages, the online encyclopedia, and core glossary, etc. The latter phrase means temporal integration and multi-sourced integration. In particular, to demonstrate the idea of knowledge base construction in physical viewpoint, we redraw and organize the knowledge base construction in Fig. 2. It follows from Fig. 2 that the knowledge base of OpenKN includes two sub-bases from bottom up: (1) The general foundation base for the commonsense knowledge, i.e., the entities, relations, concepts that everyone knows. This can be established by extracting knowledge from the online encyclopedia, such as the Wikipedia, etc. (2) N domain-specific bases from domain 1 to domain n listed from left to right. According to the source of data of the domain knowledge, each domain-specific base can be further divided into three parts, the induced general foundation base (induced GFB), the domain foundation base, and the domain web base. For the induced GFB, it contains the commonsense knowledge inherited from the general foundation base to describe domain common knowledge. For the domain foundation base, it is built by collecting domain-specific dictionaries, such as the core glossary, etc. Finally, to capture the latest and real-time domain knowledge, we have to extract it from the huge amount of web pages, such as the latest news reported by the mainstream social medias to build the domain web base. In Fig. 2, black and orange circles represent the knowledge extracted from the web pages, and the edges between them denote their relations. With the newly emerging web pages, the domain-specific bases grow adaptively. Last but not least, to reuse the well-known external knowledge bases, such as Freebase and YAGO, we also carry out the knowledge integration work. To sum up, after the knowledge base construction phrase, we obtain the explicit knowledge, namely, the raw material directly obtained from data.

Secondly, as another infrastructure of OpenKN, we design the Graph-based DataBase (GDB for short) as a huge knowledge repository supporting more than 10 billion explicit knowledge storage. Specifically, GDB is a new property graph data model with extensions on the definition of nodes and edges in contrast with the traditional graph data model,

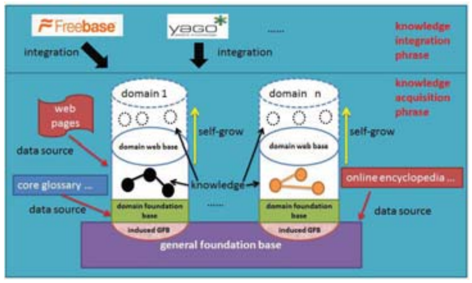

Figure 2. The knowledge base construction of OpenKN

such as the Neo4j[3], Titan[4]. More precisely, GDB consists Nodes and Edges, where each Node has a unique ID and supports a bunch of multi-valued attributes (e.g. lifetime, source of data, etc.), and each Edge has a unique Id and support a bunch of multi-valued relations (e.g. lifetime, weight, source of information, etc.). Overall, GDB describes a heterogeneous network different from the existing model called the evolutionable knowledge network, and we shall discuss this network later. To ensure the scalability, GDB is established based on HBase with a series of Tables for Nodes and Edges, respectively.

Thirdly, the knowledge validation and verification process aims to check the redundancy, conflict or contradiction, and incompleteness of the knowledge obtained by the knowledge base construction by means of expertise. Meanwhile, the knowledge computation process includes the attribute computing, the relation computing, the instance computing, etc. For the detail of the knowledge computation process, we shall discuss later with the second feature of OpenKN.

Next, we focus on the two features of OpenKN: the self-adaptability and evolutionable computing. These two features together explain the exact meaning of the terminology "open" of OpenKN.

A. The self-adaptability of OpenKN

The self-adaptability of OpenKN is reflected in two schemes, the adaptive knowledge evolution process (AKEP) and the adaptive knowledge acquisition strategy (AKAS) shown in Fig. 3.

More precisely, AKEP is used to describe the law of knowledge explosion, which can be divided into two phrases: the self-update of the knowledge base of OpenKN and the syntactic-semantic integration with other similar knowledge bases. In the self-update process, the growth can be accomplished by two effective operations on the knowledge bases: addition ($\oplus$) and multiplication ($\otimes$), and a set of elementary rules. The elementary rules are used for the evolution of primitive knowledge bases, where the primitive knowledge bases are defined to be the knowledge bases which cannot be represented by others under the two

[3]http://www.neo4j.org
[4]http://thinkaurelius.github.io/titan/

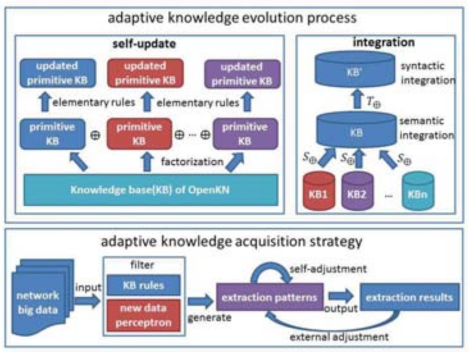

Figure 3. The self-adaptability of OpenKN

effective operations. The primitive knowledge bases can be interpreted as the basis of the vector space of the knowledge bases, and other knowledge bases can be represented as the linear combination of the primitive bases under the operations $\oplus$ and $\otimes$. For the terminology of vector space, see [21]. On the other hand, the integration of two knowledge bases can be divided into two operations: the semantic-level integration $S_{\oplus}$ and the syntactic-level integration $T_{\oplus}$.

AKAS aims to acquire dynamic knowledge over time and space. Given the network big data, AKAS uses the component "filter" to generate the syntactic-semantic extraction patterns, e.g. such-as, is-a. The filter is composed of knowledge base rules (KB) and new data perceptron. KB rules guarantee the relative consistence of extraction patterns with different types of data, while the new data perceptron is used to examine whether new data appears so as to adjust extraction strategies. The extraction patterns can be adjusted both by themselves and by the extraction results in turn. For the self-adjustment phrase, for example, when the pattern "such as" meets some exceptions, e.g. "animals other than dogs such as cats", it can not only detect this in the probabilistic way that Probase does, but also recognize them in virtue of fuzzy ontology techniques, see [22] for preliminary terminology. The external adjustment means the extraction rules can be adjusted by the final extraction results.

The self-adaptability of OpenKN can effectively meet the challenge of the velocity of network big data. On one hand, it makes the knowledge base enable to capture the newly emerging data. On the other hand, different rules like the elementary rules and the KB rules, keep the update with the pace of time.

B. The evolutionable computing of OpenKN

To demonstrate the evolutionable computing characteristic of OpenKN, we first define the evolutionable knowledge network for knowledge representation. This representation is also the prototype or graph data model which our knowledge repository GDB describes.

1) The evolutionable knowledge network: An evolutionable knowledge network is a heterogeneous network in which vertices and edges are anchored in both time and space, and a series of functions. More formally, given a time set T and a set S of spatial information, an evolutional knowledge network $G_{T,S}$ is a network defined as a eight-tuple

$$G_{T,S} = (V, E, \phi, \psi, \theta, \tau, \lambda, \eta),$$

where V is the set of vertices, E is the set of labeled edges, i.e., the set of triples (u, v, r) for $u, v \in V$ and $r \in \mathcal{R}$. $\phi : V \to \mathcal{A}$ is the type mapping function on the vertex set such that each vertex $v \in V$ is assigned a single type $\phi(v) \in \mathcal{A}$. $\psi : E \to \mathcal{R}$ is the relation mapping function such that each pair of vertices is assigned at most $|\mathcal{R}|$ relations. $\theta : V \to 2^T$ is the time mapping function on the vertex set V returning the set of timestamps describing the lifetime of a given vertex, where 2^T is the power set of T. $\tau : E \to 2^T$ is the time mapping function on the edge set E returning the set of timestamps of the presence of a given edge. Note that we record the timestamps of both the vertices and the edges. $\lambda : V \to 2^S$ is the space mapping function on the vertex set V returning the set of spatial information describing the geographical activities of a given vertex, where 2^S is the power set of S. $\eta : E \to 2^S$ is the space mapping function on the edge set E returning the set of spatial information describing the happening places of a given edge. Note that we record the time and space of both vertices and edges in the evolutionable knowledge network. It should be mentioned that since the knowledge bases of OpenKN can be domain-specific, the evolutionable knowledge network can be domain-specific in the same manner, in which the vertex type mapping function ϕ and relation type mapping function ψ are domain-dependent.

Let us take the domain of academia as an example. In an academic evolutionable knowledge network, vertices can be mapped to five types: authors (A), papers (P), conferences (C), organizations (O), and domain key words (K). Edges may represent the coauthor relationship between authors, the citation relationship between paper, etc. Moreover, each node is anchored in various temporal and spatial information, e.g. birthday, birthplace, affiliation, date of graduation, etc. Similarly, edges, are also anchored in time and space, e.g., those between two authors keep tracking of their collaboration year and city. We illustrate one snippet of such evolutionable knowledge network in Fig. 4. where the evolutionable knowledge network has the set of vertices $V = \{a, b, c, d, e, f, g, h, i, j, k, l, m, n\}$, the time set $T = \{t_1, t_2, t_3, t_4, t_5, t_6\}$ and the space set $S = \{s_1, s_2, s_3, s_4, s_5\}$. On the right-hand side of the figure, we list some functions and their corresponding values. For example, the equation $\psi(a, b, \mathrm{coauthor}) = \mathrm{coauthor}$, states that the two vertices a and b have the coauthor relationship. The equation $\phi(a) = A$ indicates that vertex

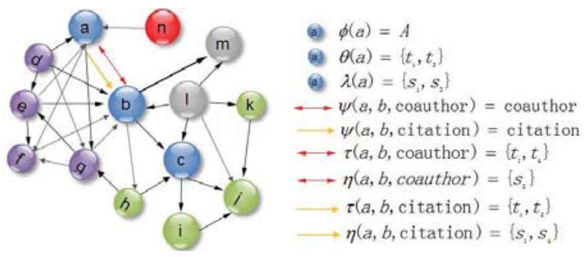
a
n
m
d
e
b
l
k
f
g
c
h
j
i
$\phi(a) = A$
$\theta(a) = \{t_1, t_2\}$
$\lambda(a) = \{s_1, s_3\}$
$\psi(a, b, \mathrm{coauthor}) = \mathrm{coauthor}$
$\psi(a, b, \mathrm{citation}) = \mathrm{citation}$
$\tau(a, b, \mathrm{coauthor}) = \{t_1, t_4\}$
$\eta(a, b, coauthor) = \{s_2\}$
$\tau(a, b, \mathrm{citation}) = \{t_1, t_2\}$
$\eta(a, b, \mathrm{citation}) = \{s_1, s_4\}$

Figure 4. An evolutional knowledge network $G_{T,S}$. The five different colors, i.e., blue, purple, red, gray and green of vertices can be mapped to the five types A, P, C, O, K, respectively.

a is of type A. The equations $\theta(a) = \{t_1, t_2\}$ and $\lambda(a) = \{s_1, s_3\}$ are the values of the time mapping function and space mapping function on vertex a, respectively. Similarly, $\tau(a, b, \mathrm{coauthor}) = \{\mathrm{t}_1, \mathrm{t}_4\}$ and $\eta(a, b, \mathrm{coauthor}) = \{\mathrm{s}_2\}$ are the values of the time mapping function and space mapping function on the edge $(a, b, \mathrm{coauthor})$, respectively.

The evolutional knowledge network of OpenKN is said to be evolutionable in two ways. On one hand, the network can capture the newly emerging data, or from the viewpoint of structure, the "newly emerging knowledge network" from the web, and acquire it promptly to update itself. On the other hand, the knowledge network can integrate the data of other knowledge bases, by converting them to the required form, or in other words, the "revised knowledge network", and then assimilating them to compose a new knowledge network. These processes are illustrated in Fig. 5 and 6. In Fig. 5, the leftmost network is an established evolutionable knowledge network. When new knowledge is acquired, it is integrated into the existing one by two steps. Firstly, the new knowledge is represented as a knowledge network, and is linked with the existing knowledge in the knowledge network. Secondly, the vertices and edges of the new knowledge network "interact" with those in the existing knowledge network, and finally they become a whole part. In Fig. 6, the existing knowledge base is first transformed into the revised knowledge network. Then the integration procedure is done by following the scenario in Fig. 5.

The evolutionable property of evolutionable knowledge network leads to a complete evolution life cycle [23] for the network evolution in literature, including detecting the need for evolution, suggesting changes, validating the changes, assessing the impact of evolution, and managing changes. Meanwhile, the evolution is soft so as to guarantee the freshness of the network. Note that we can also define the primitive evolutionable knowledge networks corresponding to the primitive knowledge bases mentioned in Fig. 3.

2) The evolutionable computing operators of OpenKN: The evolutionable computing of OpenKN can be formalized by two categories of operators, or say the evolutionable computing operators including the vertex operators and the edge operators. Specifically, the vertex operators of OpenKN

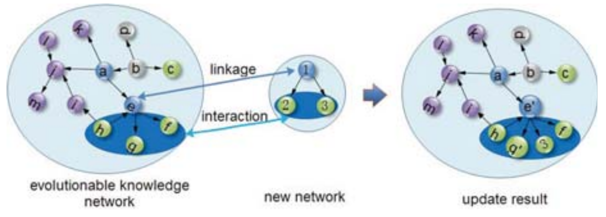

Figure 5. The procedures of evolutionable knowledge network to perceive new knowledge network and update itself.

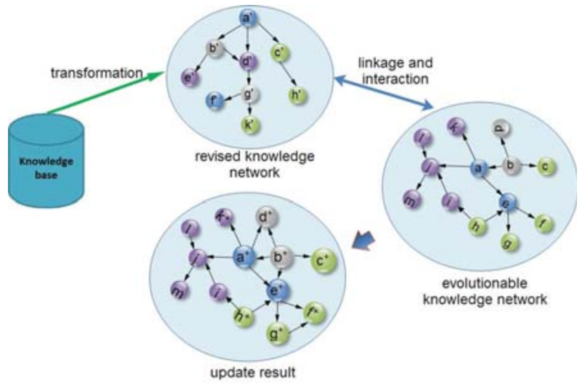

Figure 6. The procedures of evolutionable knowledge network to transform existing knowledge base to an evolutionable knowledge network for integration.

are divided into three sub-operators, the vertex extraction operators, the vertex-level integration operators and the vertex inference operators. Similarly, the edge operators are divided into three sub-operators, the relation extraction operators, the edge-level integration operators and the relation inference operators. In particular, all these sub-operators involve the operations on the temporal and spatial information assigned to each vertex and edge. These operators are also embodied in the adaptive knowledge evolution process and the adaptive knowledge acquisition strategy mentioned before. Namely, the vertex and edge extraction operators contribute the adaptive knowledge acquisition strategy, and the other operators account for the adaptive knowledge evolution process. The proposal of the evolutionable computing operators of OpenKN is the first work to summarize all algorithms concerning the holistic knowledge processing pipeline. It can help to deeply discover the intrinsic value of the network big data, and provides a comprehensive overview of different operations and their interactions during the knowledge computation process.

IV. CASE STUDY

In this part, we show one type of evolutionable computing operators of OpenKN, denoted by KP-LIM to make the relation inference in the domain of academia. We crawl the data from the scholar bibliographic website SoScholar[5],

[5]http://soscholar.com

and build the academic evolutionable knowledge network which contains five types of vertices: Author (A), Paper (P), Organization (O), Conferences (C), and Key words (K). Note that the SoScholar website is established "openly" from many sources, such as the DBLP bibliography network, the ACM digital libarary, the IEEE digital libarary. We select a subset of authors in the website who have published more than 3 papers in top conferences in five areas, i.e. Information Retrieval, Data Mining, Artificial Intelligence, Machine Learning, and Computer Science, between 1928 and 2012 as the seed users. Then we continue to find their coauthors, affiliated organizations, and key words mentioned in their paper. In total, the number of authors, paper, organizations, conferences, and key words is about 7 million, 10 million, 500 thousand, 14 thousand, and 40 thousand, respectively. Summing up these numbers leads to 17,680,000 vertices of the academic evolutionable knowledge network. For the edges of academic evolutionable knowledge network, we list the types of their ends and relations in TABLE I. In total, the number of pairs of vertices with these relations is 79,611,027 in the data set. Besides, we also collect the creating time and the spatial information of all the relations. For example, the relation "write" is anchored in the timestamp when the paper was published, and in the venue (i.e., conference) where the paper was published. For the spatiotemporal information of vertices, for the sake of simplicity, we only record the authors lifetime when he was affiliated in an organization, and the author's locations as his affiliated organizations.

Table I
THE TYPES OF ENDS AND RELATIONS OF EDGES.

Types of ends	Types of relations
A,P	write, written by
C,P	publish, published by
P,K	mention, mentioned by
A,O	employee, employer
P,P	cite, cited by

After establishing the academic evolutionable knowledge network, we continue to determine the primitive knowledge networks. In fact, it can be verified that this can be done by extracting some specific triangles in the evolutionable knowledge network. Therefore, we find all the triangles in the academic evolutionable knowledge network and randomly pick up 1500 triangles such that the vertices cover all the five vertex types. Then we arbitrarily remove one edge from each triangle such that the relations of the removed edges are as heterogeneous as possible, and keep track of these triangles. For example, in TABLE II, we list a snippet of such removed triangles. The third column presents the types of the three vertices and the two edges of the triangle such that the two vertices typed in the first column can be inferred as the relation in the second column. For example, two authors with type A's are coauthor relationship if they wrote the same paper typed in P. In particular, the number in the brackets (i.e. (2)) followed by some relations states that the relations are written twice and they are undirected.

Table II
A SNIPPET OF THE REMOVED TRIANGLES.

Types of ends	Types of relations	Removed triangles
A, A	coauthor (2)	$A \leftarrow P \rightarrow A$
P, P	same mention (2)	$P \leftarrow K \rightarrow P$
O, O	collaboration (2)	$O \leftarrow P \rightarrow O$
A, P	cite,cited by	$A \leftarrow P \rightarrow P$
A, A	coauthor (2)	$A \leftarrow C \rightarrow A$

Based on the Soscholar data set, we examine the performance of the operator KP-LIM by comparing it with the Logistic regression method that is frequently used for link inference task as the supervised baseline, and the best unsupervised predictor M-CN+ANC+OAWpres (CN for short) proposed in [24]. To evaluate the performance of different approaches, we use the commonly used measure Precision. The comparison is illustrated in Fig 7, where the number of fractions of missing links, denoted by α is the percentage of the sampled missing links over the total links between vertices in the evolutionable knowledge network. It can be seen that KP-LIM performs best for each value of the fraction. Moreover, as α increases, KP-LIM outperforms better than any of the other methods. Specifically, KP-LIM obtains about 1% $\sim$ 22% increase on precision compared with the Logistic method, and about 20% $\sim$ 36% increase on precision compared with the CN method. On average, these two increases are about 8% and 24%. This demonstrates the effectiveness of the KP-LIM.

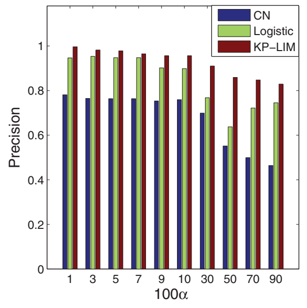

Figure 7. The precision obtained by using the three relation inference methods over the increasing fractions of missing links.

V. CONCLUSIONS

In this work, we first survey the current knowledge computational engines. Then to meet the demand of real-time and accurate knowledge acquisition and discovery from network big data, we introduce the notion of open knowledge network, a self-adaptive knowledge computational engine for network big data. And then elaborate its major characteristics to address the knowledge processing toward the 5Vs of network big data. Finally, we describe a case study for the evolutionable computing operator of OpenKN.

Acknowledgment

This work is supported by National Grand Fundamental Research 973 Program of China (No. 2014CB340401, 2013CB329602), National Natural Science Foundation of China (No. 61173008, 61232010, 61303244, 61402442), Beijing nova program (No. Z121101002512063) and National Science Supported Planning (No. 2012BAH39B02).

References

[1] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates, "Web-scale information extraction in knowitall:(preliminary results)," in *Proceedings of the 13th international conference on World Wide Web*. ACM, 2004, pp. 100–110.

[2] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka Jr, and T. M. Mitchell, "Toward an architecture for never-ending language learning." in *AAAI*, 2010.

[3] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: A collaboratively created graph database for structuring human knowledge," in *Proc. SIGMOD*, 2008, pp. 1247–1250.

[4] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: a core of semantic knowledge," in *Proceedings of the 16th international conference on World Wide Web*. ACM, 2007, pp. 697–706.

[5] J. Hoffart, F. M. Suchanek, K. Berberich, and G. Weikuma, "Yago2: a spatially and temporally enhanced knowledge base from wikipedia," *Artificial Intelligence*, vol. 194, pp. 28–61, 2013.

[6] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "Dbpedia: A nucleus for a web of open data," in *The semantic web*. Springer, 2007, pp. 722–735.

[7] W. Wu, H. Li, H. Wang, and K. Q. Zhu, "Probase: A probabilistic taxonomy for text understanding," in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. ACM, 2012, pp. 481–492.

[8] A. Singhal, "Introducing the knowledge graph: things, not strings," May 2012, http://googleblog.blogspot.co.uk/2012/05/introducing-knowledge-graph-things-not.html.

[9] J. Guo, X. Cheng, G. Xu, and X. Zhu, "Intent-aware query similarity," in *Proceedings of the 20th ACM international conference on Information and knowledge management*. ACM, 2011, pp. 259–268.

[10] Z. Zhao, Y. Jia, Y. Wang, and X. Cheng, "Content-structural relation inference in knowledge base." in *AAAI*, 2014.

[11] O. Deshpande, D. S. Lamba, M. Tourn, S. Das, S. Subramaniam, A. Rajaraman, V. Harinarayan, and A. Doan, "Building, maintaining, and using knowledge bases: a report from the trenches," in *Proceedings of the 2013 international conference on Management of data*. ACM, 2013, pp. 1209–1220.

[12] A. Yates, M. Cafarella, M. Banko, O. Etzioni, M. Broadhead, and S. Soderland, "Textrunner: open information extraction on the web," in *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*. Association for Computational Linguistics, 2007, pp. 25–26.

[13] O. Etzioni, M. Banko, S. Soderland, and D. S. Weld, "Open information extraction from the web," *Communications of the ACM*, vol. 51, no. 12, pp. 68–74, 2008.

[14] A. Carlson, J. Betteridge, R. C. Wang, E. R. Hruschka Jr, and T. M. Mitchell, "Coupled semi-supervised learning for information extraction," in *Proceedings of the third ACM international conference on Web search and data mining*. ACM, 2010, pp. 101–110.

[15] F. Suchanek and G. Weikum, "Knowledge harvesting in the big-data era," in *Proceedings of the 2013 international conference on Management of data*. ACM, 2013, pp. 933–938.

[16] S. Cucerzan, "Large-scale named entity disambiguation based on wikipedia data." in *EMNLP-CoNLL*, vol. 7, 2007, pp. 708–716.

[17] T. Lee, Z. Wang, H. Wang, and S.-w. Hwang, "Web scale taxonomy cleansing," *Proceedings of the VLDB Endowment*, vol. 4, no. 12, 2011.

[18] S. Schoenmackers, O. Etzioni, D. S. Weld, and J. Davis, "Learning first-order horn clauses from web text," in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2010, pp. 1088–1098.

[19] S. Schoenmackers, O. Etzioni, and D. S. Weld, "Scaling textual inference to the web," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2008, pp. 79–88.

[20] N. Lao, A. Subramanya, F. Pereira, and W. W. Cohen, "Reading the web with learned syntactic-semantic inference rules," in *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, 2012, pp. 1017–1026.

[21] D. C. Lay, "Linear algebra and its applications," 1997.

[22] Y. Cai, C.-m. A. Yeung, and H.-f. Leung, "Fuzzy computational ontologies in contexts," 2012.

[23] F. Zablith, G. Antoniou, M. d'Aquin, G. Flouris, H. Kondylakis, E. Motta, D. Plexousakis, and M. Sabou, "Ontology evolution: a process-centric survey," *The Knowledge Engineering Review*, pp. 1–31, 2013.

[24] G. Rossetti, M. Berlingerio, and F. Giannotti, "Scalable link prediction on multidimensional networks," in *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on*. IEEE, 2011, pp. 979–986.