International Journal on Semantic Web and Information Systems
Volume 15 • Issue 2 • April-June 2019

Language-Independent Type Inference of the Instances from Multilingual Wikipedia

Tianxing Wu, School of Computer Science and Engineering, Southeast University, Nanjing, China

Guilin Qi, School of Computer Science and Engineering, Southeast University, Nanjing, China

Bin Luo, School of Computer Science and Engineering, Southeast University, Nanjing, China

Lei Zhang, Institute AIFB, Karlsruhe Institute of Technology, Karlsruhe, Germany

Haofen Wang, Gowild Inc., Shenzhen, China

ABSTRACT

Extracting knowledge from Wikipedia has attracted much attention in recent ten years. One of the most valuable kinds of knowledge is type information, which refers to the axioms stating that an instance is of a certain type. Current approaches for inferring the types of instances from Wikipedia mainly rely on some language-specific rules. Since these rules cannot catch the semantic associations between instances and classes (i.e. candidate types), it may lead to mistakes and omissions in the process of type inference. The authors propose a new approach leveraging attributes to perform language-independent type inference of the instances from Wikipedia. The proposed approach is applied to the whole English and Chinese Wikipedia, which results in the first version of MulType (Multilingual Type Information), a knowledge base describing the types of instances from multilingual Wikipedia. Experimental results show that not only the proposed approach outperforms the state-of-the-art comparison methods, but also MulType contains lots of new and high-quality type information.

KEYWORDS

Attribute Extraction, Knowledge Base, Random Graph Walk, Semantic Web, Type Inference

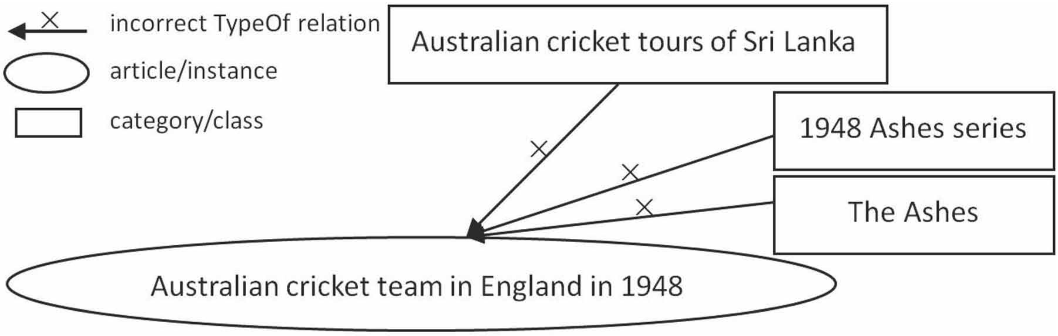
1. INTRODUCTION

Extracting knowledge from Wikipedia has attracted much attention in recent ten years. So far, different kinds of knowledge have been extracted and published as linked open dataset. One of the most valuable kinds of knowledge is type information, which refers to the axioms stating that an instance is of a certain type. These axioms can be denoted as triples, each of which is composed of a *TypeOf* relation linking from a class to an instance, e.g. "*President of the United States*" *TypeOf* "*Barack Obama*" and "*Country in Europe*" *TypeOf* "*Italy*". Type information can benefit many applications in different research fields, such as entity search (Ding et al., 2015; Tonon et al., 2013), question answering (Kalyanpur et al., 2011) and product recommendation (Hepp, 2008).

In Wikipedia, we treat a category and an article respectively as a class[1] and an instance. Thus, it seems that we can obtain the type information directly by transforming the user-generated subsumption relation from a category to an article to the *TypeOf* relation from a class to an instance. However, this user-generated subsumption relation essentially represents a *TopicOf* relation, which only means that the category is seen as a topic of the article, e.g. category "*Obama Family*" *TopicOf* article

DOI: 10.4018/IJSWIS.2019040102

Figure 1. Some categories (i.e. classes) with plural headword nouns of the article (i.e. instance) "Australian cricket team in England in 1948"


incorrect TypeOf relation
article/instance
category/class
Australian cricket tours of Sri Lanka
1948 Ashes series
The Ashes
Australian cricket team in England in 1948

"Barack Obama". Such *TopicOf* relations are obviously different from the *TypeOf* relations defined in knowledge bases (KBs).

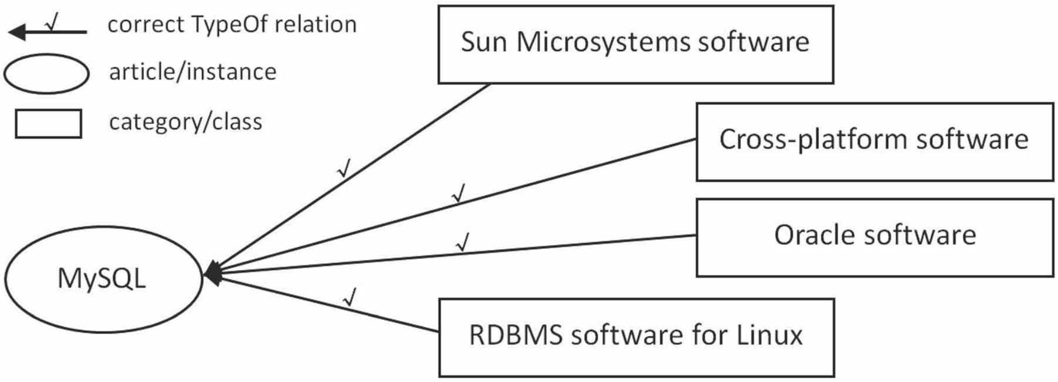
In order to overcome this problem, some works (De Melo & Weikum, 2010; Suchanek, Kasneci, & Weikum, 2008) try to discover the correct *TypeOf* relations among the user-generated *TopicOf* relations. The heuristic rules applied in the above works are similar. The central idea is that "the *TopicOf* relation from a category to an article can be transformed to the *TypeOf* relation from a class to an instance with high accuracy, when the headword noun of the given category is plural or countable" Though these heuristic rules have been proved effective, they still suffer from the following problems:

- These rules are language dependent, thus cannot be used in some languages (e.g. Chinese and Japanese), in which nouns have no explicit singular or plural forms.
- These rules cannot catch the semantic associations between instances and classes (i.e. candidate types), which may lead to mistakes and omissions in the process of type inference. For example, some classes with plural headword nouns may be incorrect types of instances (see Figure 1). Besides, all the classes with singular headword nouns are ignored, but many of them are correct types (see Figure 2).

To solve the above two problems, we introduce a language-independent feature, i.e. attributes, to help build the semantic associations between instances and classes, and infer the types of the instances from multilingual Wikipedia. Intuitively, given attributes "*actors, release date, director*" of a certain instance, people may infer that the type of this instance is "*Movie.*" However, given attributes "*name, foreign name*", people cannot infer the type of the instance because too many classes have the attribute "*name*" or "*foreign name.*" Hence, we propose an attribute-driven type inference assumption: "in Wikipedia, if an instance contains representative attributes of one of its classes (e.g. "*release date*" of "*Movie*"), there may exist a *TypeOf* relation from the class to the instance with high probability" In this paper, we study how to utilize attributes to perform language-independent type inference. It poses three main challenges as follows:

- **Data**. At first, we need to get the attributes of instances and those of classes. Since the attributes of instances are explicitly defined in infoboxes (see Figure 3(a)) in Wikipedia, we can directly extract them from infoboxes in different languages. However, the question is: where can we extract the attributes of classes?
- **Linguistics**. To use attributes to help infer the types of the instances from multilingual Wikipedia, we need to get the attributes of classes in different languages. Can we extract the attributes of classes without language-dependent features?

Figure 2. Some categories (i.e. classes) with singular headword nouns of the article (i.e. instance) "MySQL"



- **Model**. After obtaining the attributes of instances and those of classes, we can easily associate instances with their corresponding classes using shared attributes. However, the question is: how to define a model based on the above attribute-driven type inference assumption, to infer whether there exists a *TypeOf* relation from one class to the given instance?

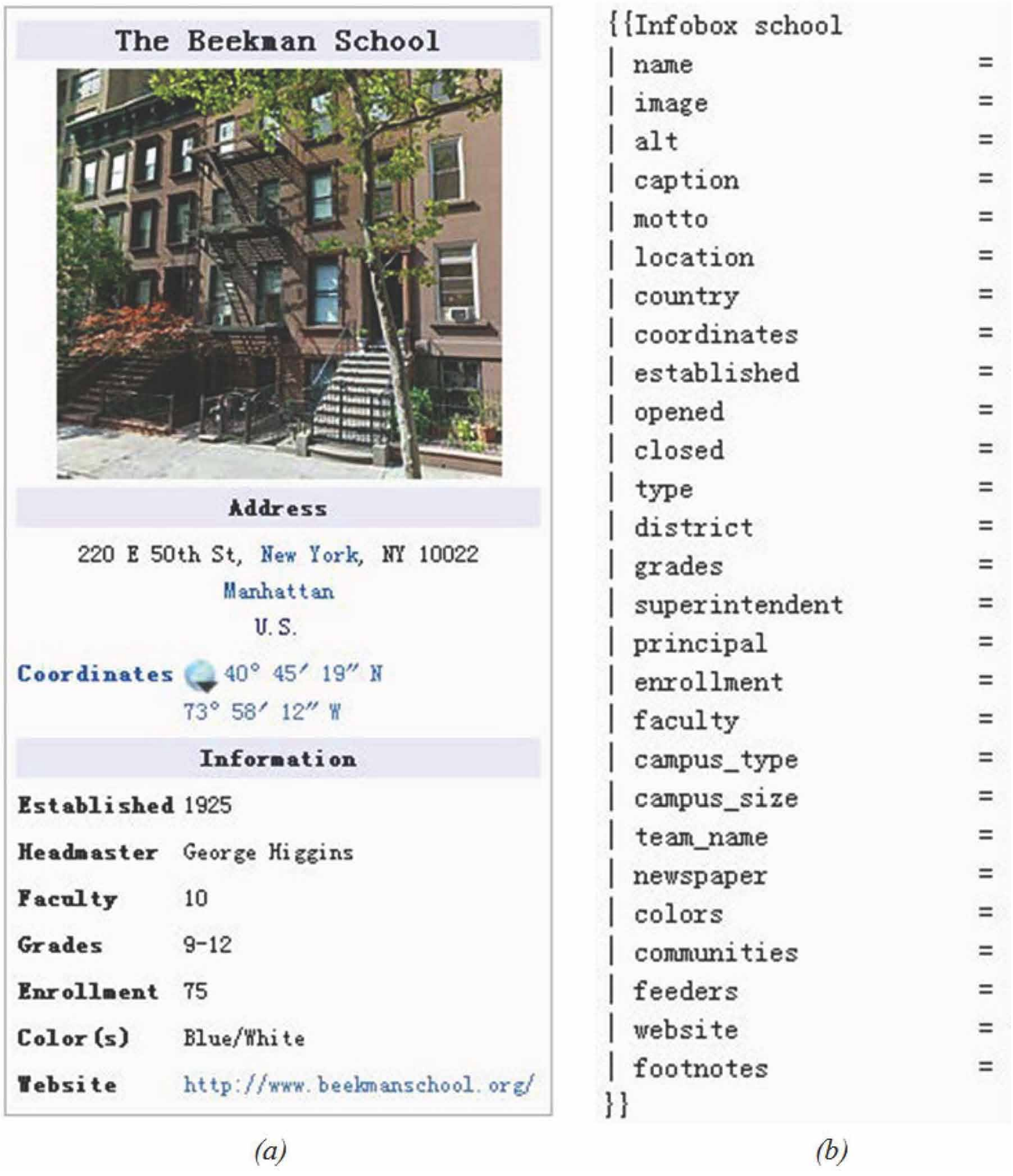
To handle these challenges, after extracting the attributes of instances from infoboxes, we present a general algorithm incorporating several language-independent rules to get the attributes of classes. Then, we propose a random graph walk model to compute the probability of a *TypeOf* relation existing between a class and the given instance. Finally, considering that the attributes of many instances are still incomplete or missing, for each instance, we apply a method to get its most similar instances, which already have attributes and are used to further improve the proposed random graph walk model.

Since all parts of the proposed approach do not rely on any linguistic feature, it can be applied in different languages, and here we evaluate our approach in two important languages: English and Chinese, which are top two most spoken languages[2] (including native and non-native speakers) all over the world. English is also the most widely used language in international activities, so it is valuable to mine English type information. Since Chinese is a representative language in which nouns have no explicit singular or plural forms, testing our approach in Chinese can verify whether it is able to solve the language-dependent problem of the rule-based approach. After applying our approach on the whole English and Chinese Wikipedia, we get the first version of MulType (Multilingual Type Information), which is a knowledge base (KB) describing the types of instances from multilingual Wikipedia. Compared with DBpedia (Bizer et al., 2009; Lehmann et al., 2015), Yago (Suchanek, Kasneci, & Weikum, 2008; Mahdisoltani, Biega, & Suchanek, 2015) and LHD (Kliegr, 2015), MulType not only first contributes a large scale of Chinese type information, but also contains lots of new English type information.

In summary, the main contributions of this work are listed as follows:

- We propose a language-independent approach to type inference of the instances in different languages from Wikipedia, which facilitates multilingual type information mining.
- We publish the obtained type information (i.e. MulType) as the open data on the Web and provide three different ways to access the data, including the downloading of the whole data, lookup service and SPARQL endpoint.
- We carry out a comprehensive set of experiments to evaluate our approach. Experimental results show that the proposed approach not only harvests the large-scale, high-quality English and Chinese type information, but also significantly outperforms the designed comparison methods in terms of precision, recall and F1-score.

Figure 3. An example of: (a) infobox; (b) infobox template



The Beekman School

Address

220 E 50th St, New York, NY 10022
Manhattan
U.S.

Coordinates	40° 45′ 19″ N 73° 58′ 12″ W

Information

Established	1925
Headmaster	George Higgins
Faculty	10
Grades	9-12
Enrollment	75
Color(s)	Blue/White
Website	http://www.beekmanschool.org/

(a)

```
{{Infobox school
| name                  =
| image                 =
| alt                   =
| caption               =
| motto                 =
| location              =
| country               =
| coordinates           =
| established           =
| opened                =
| closed                =
| type                  =
| district              =
| grades                =
| superintendent        =
| principal             =
| enrollment            =
| faculty               =
| campus_type           =
| campus_size           =
| team_name             =
| newspaper             =
| colors                =
| communities           =
| feeders               =
| website               =
| footnotes             =
}}
```

(b)

The rest of this paper is organized as follows. Section 2 outlines the related work. Section 3 introduces our approach, including the details of attribute extraction and type information generation. Section 4 presents the experimental results. Section 5 shows the Web access to MulType. The last section concludes this paper and describes our future work.

2. RELATED WORK

In this section, we review the related work on type inference in knowledge base (KB) construction, type inference in KB completion, type inference of named entities and class attribute extraction.

2.1. Type Inference in KB Construction

Type inference in KB construction aims to infer the types of instances from scratch. Auer et al. (Auer & Lehmann, 2007) proposed an approach to infer instance types using the local names of infobox templates in Wikipedia. Gangemi et al. (Gangemi et al., 2012) proposed an approach to extract type information from the abstracts of articles in Wikipedia leveraging a set of lexico-syntactic patterns, disambiguate the identified types with WordNet, and align these types to different ontologies. Similarly, Kliegr (Kliegr, 2015) also utilized Hearst patterns (Hearst, 1992) to extract types from the first sentences of articles in Wikipedia, and the types were disambiguated to DBpedia classes. Our paper is different from them, because we try to infer the types of instances from the existing classes (i.e. categories) in Wikipedia, not from the local names of infobox templates or the abstracts of articles.

The most relevant works (De Melo & Weikum, 2010; Suchanek, Kasneci, & Weikum, 2008) used heuristic rules to perform language-dependent type inference from the classes in Wikipedia. Here, we focus on language-independent type inference of the instances from multilingual Wikipedia. We make a detailed comparison between our approach and the rule-based approach in experiments.

2.2. Type Inference in KB Completion

There also exist some works on complementing the missing type information in KBs using the existing types of instances. Nuzzolese et al. (Nuzzolese et al., 2012) used two techniques respectively based on induction and abduction, exploiting the link structure in Wikipedia to complete type information of DBpedia with the existing instance types. Paulheim and Bizer (Paulheim & Bizer, 2013; Paulheim & Bizer, 2014) proposed a heuristic link-based type inference mechanism SDType, which can be applied to any RDF KB. Kliegr and Zamazal (Kliegr & Zamazal, 2016) introduced a text-mining based approach for inferring instance types in KBs. This approach first maps classes between different KBs using the Statistic Type Inference (STI) algorithm (Kliegr & Zamazal, 2014), which is a generic co-occurrence-based algorithm calculating the common instances of classes to support such mappings. Then, it utilizes a hierarchy of Support Vector Machines classifiers to infer new types of the instances in given KBs. Melo et al. (Melo, Völker, & Paulheim, 2017) utilized hierarchical multi-label classification to perform type completion in RDF KBs. Their works are different from ours in that we infer the types of instances without leveraging any existing types of instances in current KBs.

2.3. Type Inference of Named Entities

Type inference of named entities is classifying identified entities in the text into a set of predefined types. Many approaches have been proposed to solve this problem. Both the FIGER system (Lin, Mausam, & Etzioni, 2012) and PEARL system (Nakashole, Tylenda, & Weikum, 2013) rely on relational patterns to mine type information with the types in the given KB. Another kind of approach (Fleischman & Hovy, 2002; Rahman & Ng, 2010; Ling & Weld, 2012; Yosef et al., 2012; Yosef et al., 2013) is to design different features and different classifiers to assign the predefined types to named entities. FINET (Del Corro et al., 2015) generates candidate types using a sequence of multiple extractors and selects the most appropriate types using the algorithm of word-sense disambiguation. Compared with previous approaches, FINET is less reliant on a specific KB or training data. Recently, embedding-based approaches (Ren et al., 2016; Ma, Cambria, & Gao, 2016; Abhishek, Anand, & Awekar, 2017) are popular and they aim to embed entities and types into the same vector space and design different score functions to predict whether there is a *TypeOf* relation between the given type and entity.

The input of the systems given in these works is a piece of text, which may contain named entities. These systems will identify such entities in the text and infer their types. Different from them, our work does not infer types of the entities in a given text. The input of our work is an instance and the classes in its corresponding Wikipedia page (Figure 1 and Figure 2 give two examples), and the output is the correct types (from such classes) of the given instance.

2.4. Class Attribute Extraction

Acquiring attributes of classes without language-dependent features is one of the main challenges for type inference in our paper. Pasca et al. (Pasca, 2007; Pasca & Van Durme, 2007; Pasca & Van Durme, 2008) proposed different methods to design and mine extraction patterns to get the attributes of classes from Web scale documents or query logs. Lee et al. (Lee et al., 2013) first presented different ways to extract the attributes of instances and attributes of classes from documents, query logs and KBs, and then further mapped the attributes of instances into a broad class space. Since Web scale documents or query logs cannot be easily obtained and almost all the seed patterns of extraction are language-specific, we cannot utilize these methods to extract attributes of classes from multilingual Wikipedia.

3. APPROACH

In this section, we introduce our approach for language-independent type inference in Wikipedia. The whole workflow is given in Figure 4. The input is all instances (i.e. articles) with their corresponding classes (i.e. categories) in Wikipedia. Then, the part of attribute extraction aims to extract the attributes of instances from infoboxes and those of classes with a general algorithm incorporating several language-independent rules. Subsequently, the part of type information generation first acquires each given instance's most similar instances that have attributes using an integrated instance similar score, and then performs type inference with a random graph walk model. Finally, the output of our approach is a large-scale KB of type information called MulType.

3.1. Attribute Extraction

Since the attributes of instances can be easily extracted from the infoboxes in their corresponding Wikipedia pages, we discuss how to extract the attributes of classes from multilingual Wikipedia without using language-specific features in this section.

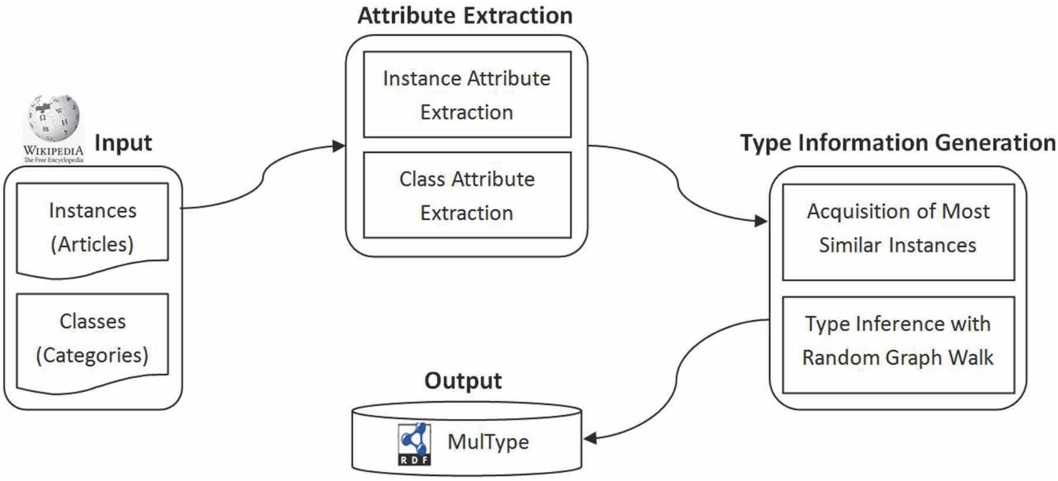
In Wikipedia, infobox templates (Figure 3(b)) gives an example of infobox template containing the attributes of class "*school*") are used to describe the characteristics of classes, so we treat infobox templates as one of the sources for extracting attributes of classes. Thus, we define an Infobox Template based Extraction Rule (IT-ER) as follows:

Definition 1: *Infobox Template based Extraction Rule (IT-ER). Given an infobox template **it** and a class c, the local names of **it** and c are respectively denoted as n(it) and n(c). All the attributes of **it** can be propagated to c if*

- *n(**it**) and n(c) are the same (ignoring case and the difference in singular and plural forms for some languages),*
- *or "Category:n(**it**)"can redirect to c,*
- *or n(**it**) can redirect to n(c) or n(c) can redirect to n(**it**).*

For example, infobox template "*Template:Infobox islands*" and class "*Category:Islands*" have the same local name "*islands*". In addition, it is obvious to find that the singular forms of "*n(Category:Countries)*" (i.e. "*Country*") and "*n(Template:Infobox country)*" (i.e. "*country*") are the same when ignoring case. Besides, if we submit the query "*Category:n(Template:Infobox university)*" to Wikipedia, it can be redirected to "*Category:Universities and colleges*". Moreover,

Figure 4. The workflow of our proposed approach


Attribute Extraction
Instance Attribute Extraction
Class Attribute Extraction
Input
Instances (Articles)
Classes (Categories)
Type Information Generation
Acquisition of Most Similar Instances
Type Inference with Random Graph Walk
Output
MulType

"*n(Category:States of the United States)*" can redirect to "*n(Template:Infobox U.S. state)*". Since the infobox templates are summarized by the wisdom of crowds, we argue that the attributes in any infobox template are correct and complete. Hence, we will not add new attributes to the class with attributes extracted using IT-ER.

Inspired by the inheritance in object-oriented programming (Meyer, 1988) that one class can inherit all the attributes of its super-class (e.g. class "*Manager*" can inherit the attribute "*gender*" from its super-class "*Person*"), we try to apply this idea to extract the attributes of classes using the class hierarchy (i.e. category system) of Wikipedia. The inheritance is reasonable when there exists a strict *isA* relation between classes. Thus, we first collect sets of *isA* relations between classes from open knowledge bases, including DBpedia, Yago, BabelNet (Navigli & Ponzetto, 2012), WikiTaxonomy (Ponzetto & Strube, 2007; Ponzetto & Strube, 2008) and Zhishi.schema (Wang et al., 2014; Wu, Wang, & Qi, 2014). Each set corresponds to one language. Then, we utilize these sets of *isA* relations to refine the class hierarchies of different languages, i.e. removing the edges between the classes in the hierarchies when we cannot find an ***isA*** relation between them. As a result, each language corresponds to a refined class hierarchy. Based on such refined class hierarchies, we define a Top-Down Hierarchy-based Extraction Rule (TDH-ER) as follows:

Definition 2: *Top-Down Hierarchy-based Extraction Rule (TDH-ER). If a class c has no attribute extracted from infobox templates, and it has super-classes with attributes, then all the attributes of its super-classes should be inherited by c.*

Since the authors of the work (Dowty, Wall, & Peters, 1981) propose that a class is traditionally a placeholder for a set of instances sharing similar attributes (properties), some subsequent works (Pasca & Van Durme, 2007; Lee et al., 2013; Stojanovic, 2004) leverage this idea to extract the attributes of classes from their corresponding instances or sub-classes. Here, we extend this idea to extract the attributes of classes in the refined class hierarchies of Wikipedia. A Bottom-Up Hierarchy-based Extraction Rule (BUH-ER) based on the idea of majority voting is defined as follows:

Definition 3: *Bottom-Up Hierarchy-based Extraction Rule (BUH-ER). If a class c has no attribute extracted from infobox templates, and it has hyponyms (include instances and sub-classes) with attributes, then the attributes can be propagated to c when they are shared by more than half of these hyponyms.*

To utilize the proposed language-independent rules together, we present a general iterative algorithm (i.e. Algorithm 1 given in Figure 5) for extracting the attributes of classes in multilingual Wikipedia. In Algorithm 1, given one language L, we first input its corresponding refined class hierarchy $CH^L = (C^L, R^L)$ and a 2-tuple $TUP^L = (C^L, A^L)$, where C^L is the set of all classes in the refined class hierarchy, R^L is the set of *isA* relations (i.e. directed edges) between the classes in C^L, A^L represents sets of attributes corresponding to the classes in C^L, and each set in A^L is initialized as $\varnothing$. Subsequently, we apply IT-ER to each class in C^L (*line 1*). To use TDH-ER, we need to handle the classes in CH^L from the root node to leaf nodes. Conversely, to use BUH-ER, we should deal with the classes in CH^L from leaf nodes to the root node. Therefore, we initialize three queues (a queue is a First-In-First-Out data structure): $Queue_{td}$, $Queue_{bu}$ and $Queue$ as $\varnothing$ (*line 2*), then sort the classes in C^L by their maximum depths in CH^L (in ascending or descending order) (*line 3*), and put each sorted class into $Queue_{td}$ and $Queue_{bu}$ in ascending order and descending order, respectively (*line 4-5*). In top-down class attribute extraction with TDH-ER, we only need to set the variable $Queue$ equal to $Queue_{td}$, and handle the classes in $Queue$ in order (*line 7-10*). Similarly, in bottom-up class attribute extraction with BUH-ER, we set the variable $Queue$ equal to the $Queue_{bu}$, and handle the classes in $Queue$ in order (*line 13-16*). We iteratively perform top-down class attribute extraction and bottom-up class attribute extraction over CH^L until convergence (i.e. TUP^L stops changing (*line 11-12*) and (*line 17-18*)). Note that although there exist some noise relations (i.e. *TopicOf* relations) from classes to instances, the quality of extraction can still be guaranteed because BUH-ER is applied after IT-ER and TDH-ER, and based on the idea of majority voting, a large number of correct *isA* relations can greatly relieve the negative effect of these noise relations in this algorithm.

3.2. Type Information Generation

After extracting attributes of instances and classes, each instance, its attributes and the classes in its corresponding Wikipedia page naturally form a graph, in which the given instance and its classes are linked by shared attributes. To compute the probability of the class being the type of the given instance leveraging the graph structure, we first propose a random graph walk model to infer the types of each given instance. Considering the attributes of many instances are still incomplete or missing, we then present a method to get each instance's most similar instances that have attributes and improve the random graph walk model with this kind of data.

3.2.1. Random Graph Walk Model

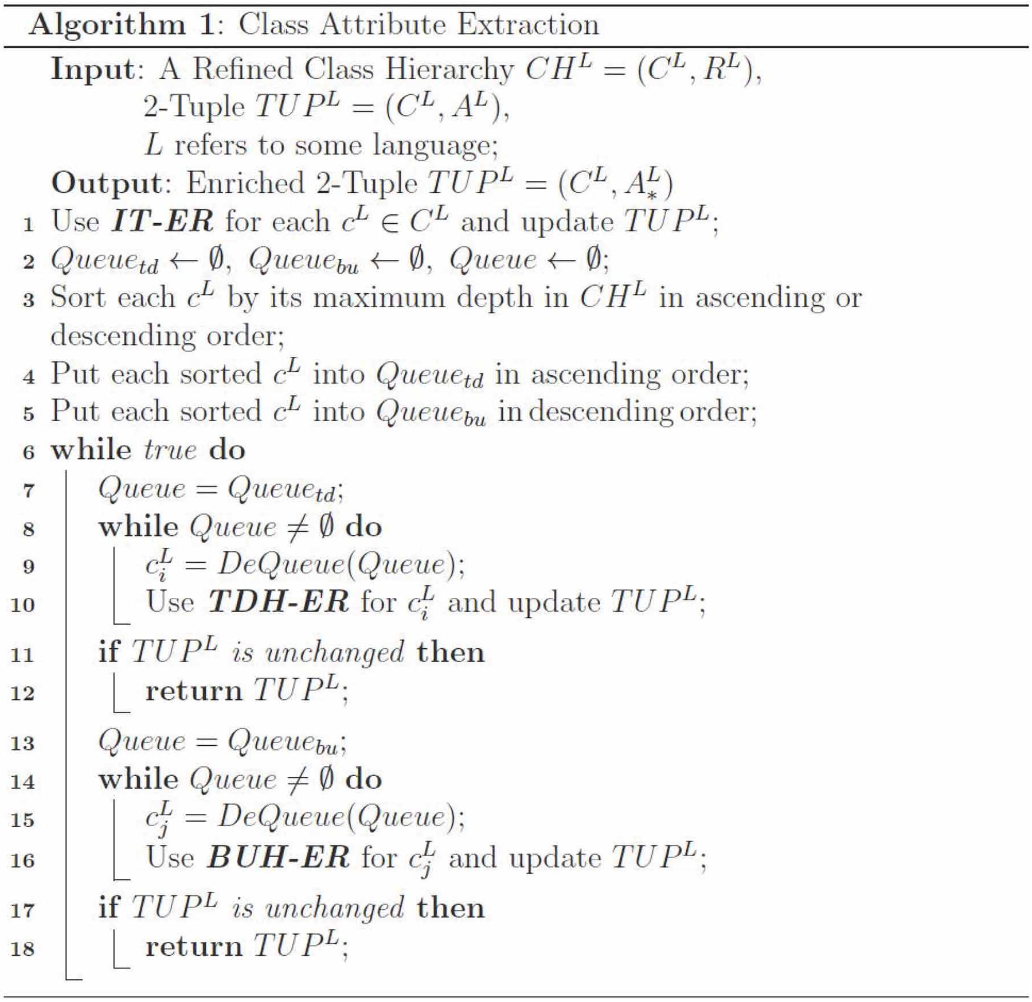
3.2.1.1. Graph Construction

Given an instance i, it may contain a set of attributes $\{a_j^i\}_{j=1}^n$ and have a set of classes $\{c_k^i\}_{k=1}^m$ in its corresponding Wikipedia page. Each class c_k^i may also correspond to a set of attributes $A^{c_k^i}$. Here, we create a directed edge from i to each of its attributes a_j^i. If an attribute a_j^i is also an element in the set $A^{c_k^i}$ of some class c_k^i, a directed edge will be built from a_j^i to c_k^i. An example of the constructed graph of i is shown in Figure 5. Since all the attributes are derived from Wikipedia, people are free to use different labels to represent an attribute with the same meaning (e.g. author and writer). This may result in the absence of many edges from attributes to classes. Hence, we group synonymous attributes using BabelNet (Navigli & Ponzetto, 2012), and if some class c_k^i does not have the attribute a_j^i, but has a synonymous attribute of a_j^i, then the directed edge connecting from a_j^i to c_k^i will also be created.

3.2.1.2. Random Graph Walk

According to the attribute-driven type inference assumption described in Section 1, if an instance contains more representative attributes of one of its corresponding classes, then the class may probably

Algorithm 1. Algorithm for class attribute extraction



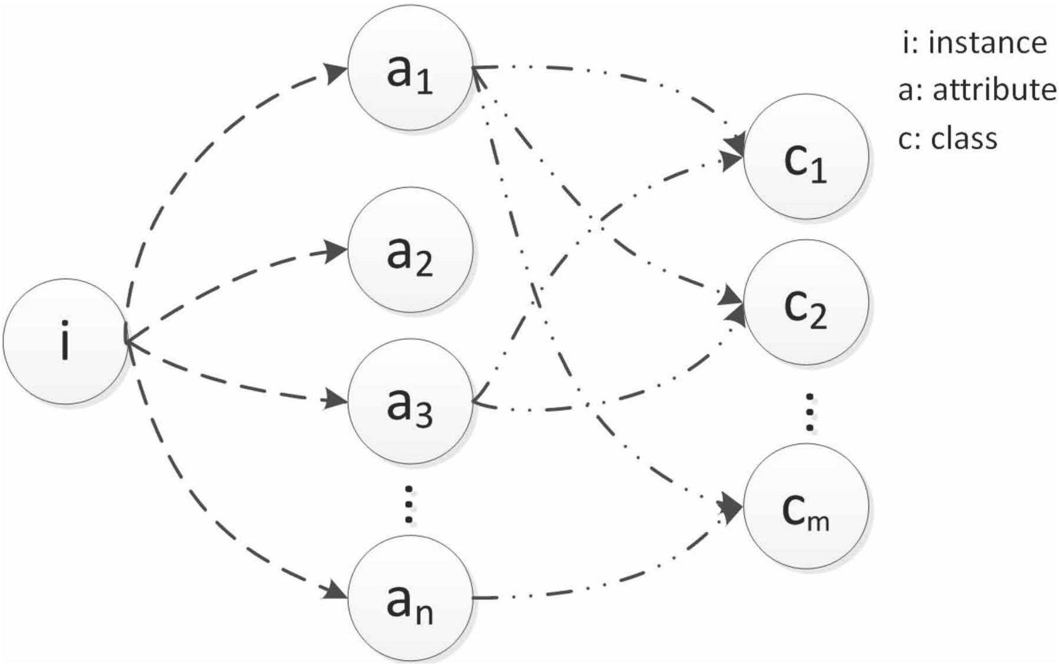
```
Algorithm 1: Class Attribute Extraction
Input: A Refined Class Hierarchy $CH^L = (C^L, R^L)$,
       2-Tuple $TUP^L = (C^L, A^L)$,
       $L$ refers to some language;
Output: Enriched 2-Tuple $TUP^L = (C^L, A^L_*)$
Use IT-ER for each $c^L \in C^L$ and update $TUP^L$;
$Queue_{td} \leftarrow \emptyset$, $Queue_{bu} \leftarrow \emptyset$, $Queue \leftarrow \emptyset$;
Sort each $c^L$ by its maximum depth in $CH^L$ in ascending or descending order;
Put each sorted $c^L$ into $Queue_{td}$ in ascending order;
Put each sorted $c^L$ into $Queue_{bu}$ in descending order;
while true do
    $Queue = Queue_{td}$;
    while $Queue \neq \emptyset$ do
        $c^L_i = DeQueue(Queue)$;
        Use TDH-ER for $c^L_i$ and update $TUP^L$;
    if $TUP^L$ is unchanged then
        return $TUP^L$;
    $Queue = Queue_{bu}$;
    while $Queue \neq \emptyset$ do
        $c^L_j = DeQueue(Queue)$;
        Use BUH-ER for $c^L_j$ and update $TUP^L$;
    if $TUP^L$ is unchanged then
        return $TUP^L$;
```

be the type of this instance. Driven by this idea, on the constructed graph of each instance, if some class can be reached by more of its representative attributes, then the probability of the class being the type of the given instance is higher. Based on this analysis, we design a random graph walk model for type inference of each instance. In this model, we first compute two kinds of transition probabilities between different nodes in the constructed graph of the given instance, and then get the probability of a *TypeOf* relation existing between a class and the given instance through random walks over the whole graph.

Transition Probability Starting from the Given Instance. Given an instance i, the transition probability P_{IA} from i to one of its attributes a^i_j in the constructed graph is defined as

$$P_{IA}\left(i,a^i_j\right)=\frac{Weight(a^i_j)}{\sum_{j=1}^{n}Weight(a^i_j)} \tag{1}$$

Figure 5. An example of the constructed graph of instance i



where *Weight*(a_j^i) is the reciprocal value of the frequency that a_j^i occurs in the attribute sets of all the classes in Wikipedia. We argue that the fewer classes an attribute is shared by, the more representative this attribute is, so if the frequency of a_j^i is low, its weight will be high, which shows that a_j^i is a representative attribute because it is shared by few classes. When walking a step from *i* to a_j^i, the walk tends to choose the most representative attribute (i.e. the attribute belonging to the fewest classes) so that it has better chance to walk to the correct classes (i.e. types).

Transition Probability Starting from the Attributes. Given an instance *i*, to walk one step away from *i*'s attribute a_j^i to a class c_k^i, c_k^i is picked uniformly from the classes that have edges connecting a_j^i. This transition probability P_{AC} is defined as

$$P_{AC}\left(a_j^i, c_k^i\right) = \frac{1}{|N_C|} \tag{2}$$

where $|N_C|$ is the number of the classes that have edges connecting a_j^i in the constructed graph of *i*. This definition means that the classes having edges connecting a_j^i have an equal opportunity as the destination. For example, in Figure 6, there are two directed edges from attribute a_3 to classes c_1 and c_2, so both of the transition probabilities $P_{AC}(a_3, c_1)$ and $P_{AC}(a_3, c_2)$ are 0.5.

With these two kinds of transition probabilities, we define a 2-step random walk process over the graph of each instance as follows:

1. From an instance *i*, walk a random step to one of its attributes a_j^i with the transition probability $P_{IA}\left(i,a_j^i\right)$ (Equation 1).
2. From an attribute a_j^i, walk a random step to one of the classes c_k^i with the transition probability $P_{AC}\left(a_j^i,c_k^i\right)$ (Equation 2).

Based on this random walk process for instance *i*, the random graph walk model computes the probability P_{rgw} of reaching each class c_k^i from *i* using the following formula:

$$P_{rgw}\left(i,c_k^i\right)=\sum_{j=1}^{n}P_{IA}\left(i,a_j^i\right)\cdot P_{AC}\left(a_j^i,c_k^i\right) \tag{3}$$

We then normalize all of these probabilities (i.e. $P_{rgw}\left(i,c_k^i\right)$) to sum up to 1. Each normalized probability is taken as the probability of a *TypeOf* relation existing between each class and *i*. If the normalized probability of class c_k^i is greater than a threshold θ_1, c_k^i will be determined as a type of *i*.

3.2.2. Acquisition of Most Similar Instances

In order to better infer types of the instances which are short of attributes, we leverage each instance's most similar instances that have attributes to improve the proposed random graph walk model. Here, we present a method with a context similarity metric and an existing classes similarity metric to measure the similarity degree between instances.

Context Similarity Metric. To measure the context similarity degree between instances, we compute the similarities between the instances' vectors trained on the whole Wikipedia corpus. Each vector is treated as the context representation of each instance. If two instances appear in similar contexts, then they are close to each other in the same vector space. In our work, we use word2vec (Mikolov et al., 2013) to generate the vector of each instance. In the training process, each instance is taken as one word unit. Given two instances i_1 and i_2, the context similarity metric between their vectors $v(i_1)$ and $v(i_2)$ is defined as

$$CSM(i_1,i_2)=\frac{v(i_1)\times v(i_2)}{|v(i_1)|\times|v(i_2)|} \tag{4}$$

The context similarity metric is actually the cosine similarity between two vectors $v(i_1)$ and $v(i_2)$.

Existing Classes Similarity Metric. Similar instances tend to share similar topics (i.e. the classes in Wikipedia pages of instances). Thus, given two instances i_1 and i_2, we define the set of classes in the Wikipedia pages for each instance as its existing class set, and the existing classes similarity metric between two existing class sets $ECset(i_1)$ and $ECset(i_2)$ is defined as

$$ECSM(i_1,i_2)=\frac{|ECset(i_1)\cap ECset(i_2)|}{|ECset(i_1)\cup ECset(i_2)|} \tag{5}$$

The existing classes similarity metric is actually the Jaccard Similarity between two sets $ECset(i_1)$ and $ECset(i_2)$.

To balance the two metrics defined above, we get each instance's most similar instances that have attributes by maximizing the product of the two metrics. The Integrated Instance Similar Score between two instances i_1 and i_2 is defined as

$$IISS(i_1,i_2)=(CSM(i_1,i_2)+1)\times(ECSM(i_1,i_2)+1) \tag{6}$$

There may exist one metric value between i_1 and i_2 that equals 0. This is problematic because it will result in a 0 Integrated Instance Similar Score, which wipes out other metric values when they are directly multiplied. Thus, we add 1 to each value of both the metrics before the multiplication in Equation (6). If there does not exist an instance i_2 satisfying $IISS(i_1, i_2) > 1$, the instance i_1 will not get any similar instance.

3.2.3. Improved Random Graph Walk Model

For each instance, after obtaining its most similar instances that have attributes, we leverage this data to revise the constructed graph and improve the random graph walk model.

3.2.3.1. Graph Revision

Given an instance *i*, we revise its original constructed graph by adding *i*'s most similar instances $\{s_r^i\}_{r=1}^t$ and their attributes. We first enrich the attribute set $\{a_j^i\}_{j=1}^n$ of *i* with the attributes of the obtained most similar instances, which results in an updated attribute set $\{a_j^i\}_{j=1}^{n^+}$ ($n^+ \geq n$). We then create directed edges respectively from *i* to each element in $\{s_r^i\}_{r=1}^t$ and from each similar instance s_r^i to each of its attributes. If a newly added attribute is an element in the attribute set $A^{c_k^i}$ of some class c_k^i, a directed edge will also be built from this attribute to c_k^i. Figure 6 gives an example of the revised graph of *i*.

3.2.3.2. Improved Random Graph Walk

According to the revised graph of each instance, we need to redefine the transition probability starting from the given instance and revise the random walk process to improve the original model.

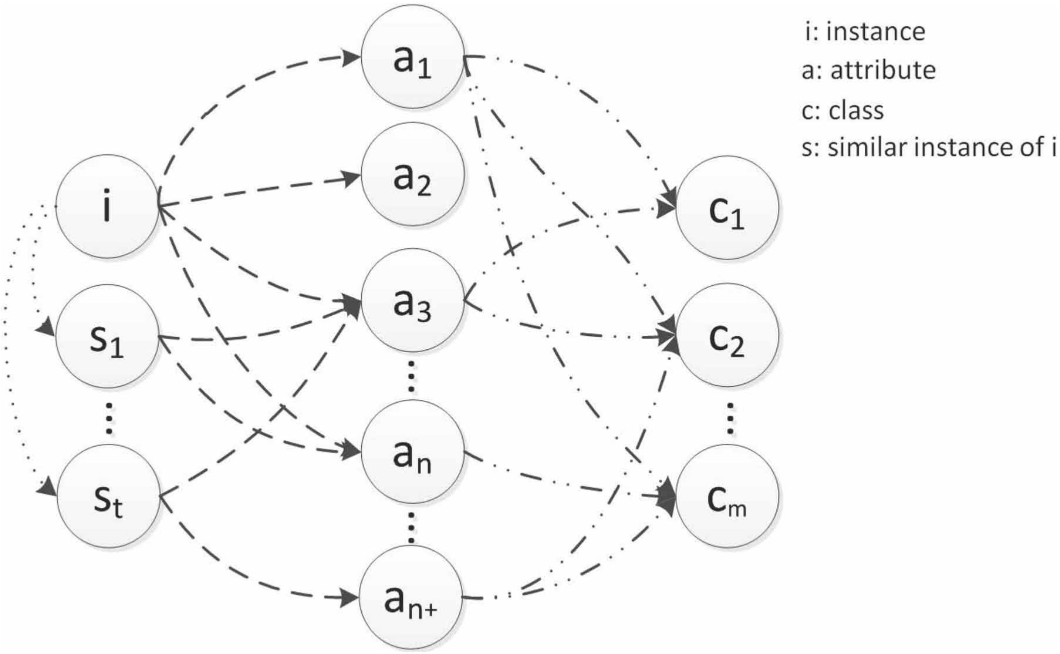
Revised Transition Probability Starting from the Given Instance. Given an instance *i*, when walking one step away from *i*, the walk may get to an attribute a_j^i or one of *i*'s most similar instances s_r^i. Therefore, we define a revised transition probability RP_{IA} from *i* to a_j^i and a new transition probability P_{IS} from *i* to s_r^i as

$$RP_{IA}\left(i,a_j^i\right)=\alpha\cdot P_{IA}\left(i,a_j^i\right) \tag{7}$$

$$P_{IS}\left(i,s_r^i\right)=(1\text{-}\alpha)\times\frac{IISS(i,s_r^i)}{IISS(i,*)} \tag{8}$$

Where $\alpha\in[0,1]$ is a constant, $P_{IA}\left(i,a_j^i\right)$ is the original transition probability (Equation (1)) from *i* to a_j^i, $IISS(i,s_r^i)$ is the Integrated Instance Similar Score between *i* and s_r^i (Equation (6)), and $IISS(i,*)$ is the sum of Integrated Instance Similar Score between *i* and each of its most similar instances. Here, there are two extreme cases for the value of α : 1) if *i* originally has its own attributes

Figure 6. An example of the revised graph of instance i



after the step of attribute extraction but does not have the most similar instances that have attributes, we set $\alpha=1$; 2) if *i* does not have its own attributes but does have the most similar instances that have attributes, we set $\alpha=0$.

The second revision for the original model is that we only modify the first step of the random walk process defined in Section 3.2.1.2. In the modified first step, starting from an instance *i*, the walk may get to an attribute a_j^i either by the directed edge from *i* to a_j^i with the transition probability $RP_{IA}\left(i,a_j^i\right)$ (Equation (7)), or by two steps through one of *i*'s most similar instances s_r^i with the transition probabilities $P_{IS}\left(i,s_r^i\right)$ (Equation (8)) and $P_{IA}\left(s_r^i,a_j^i\right)$ (Equation (1)). Therefore, we define a revised probability RP_{rgw} of reaching each *i*'s class c_k^i from *i* itself based on the revised random walk process as follows:

$$RP_{rgw}\left(i,c_k^i\right)=\sum_{j=1}^{n^+}RP_{IA}\left(i,a_j^i\right)\cdot P_{AC}\left(a_j^i,c_k^i\right)+\sum_{r=1}^{t}P_{IS}\left(i,s_r^i\right)\cdot\sum_{j=1}^{n^+}P_{IA}\left(s_r^i,a_j^i\right)\cdot P_{AC}\left(a_j^i,c_k^i\right) \quad (9)$$

After normalizing all of the above probabilities (i.e. $RP_{rgw}\left(i,c_k^i\right)$) to sum up to 1, we also need to learn a threshold θ_2 to decide whether there exists a *TypeOf* relation from the class c_k^i to the instance *i*. The details of parameters training are given in Section 4.1.3.

4. EXPERIMENTS

In this section, we evaluated the effectiveness of our approach for type inference and the quality of the obtained type information in MulType.

Table 1. The details of each dataset

Dataset	#Correct Triples (percentage)	#Incorrect Triples (percentage)
English	4,484 (*76.2%*)	1,402 (*23.8%*)
Chinese	4,972 (*85.8%*)	821 (*14.2%*)

4.1. Evaluation of the Proposed Approach

We evaluated our approach on two datasets in different languages. All the datasets and codes used in this work are publicly available[3].

4.1.1. Datasets and Evaluation Metrics

- **Data Sets:** We randomly selected 1,000 English articles and 1,000 Chinese articles from Wikipedia. To generate the evaluation benchmark for each dataset, we asked five graduate students to label whether there exists a *TypeOf* relation from each category (i.e. class) to the given article (i.e. instance). The labeled results are based on majority voting. The details of labeled correct (or incorrect) triples representing *TypeOf* relations are shown in Table 1.
- **Evaluation Metrics:** After training the parameters (will be explained in Section 4.1.3) in our approach and some of the designed comparison methods (will be introduced in Section 4.1.2) on the labeled datasets, we evaluated the results with three metrics, which are in terms of P (Precision), R (Recall) and F1 (F1-score).

4.1.2. Comparison Methods

In this paper, our approach is based on the attribute-driven random graph walk model (i.e. the improved random graph walk model proposed in Section 3.2.3.2). We compared our approach (denoted as ARGW) with the following methods.

- **Heuristic Rules (HR):** The heuristic rules are used in some works (De Melo & Weikum, 2010; Suchanek, Kasneci, & Weikum, 2008) for inferring the types of instances from Wikipedia by checking whether the headword noun of the given class is plural or countable. This is the state-of-the-art method to infer instance types from the existing classes in Wikipedia. Note that when applying this method to the Chinese dataset, we first transformed the Chinese instances and classes into English ones through the cross-language links in Wikipedia, and then inferred the types of instances.
- **Word Embedding (WE):** For any specific language, our approach generates a vector representation for each instance (introduced in Section 3.2.2) using word2vec (Mikolov et al., 2013) on the all Wikipedia articles of that language. Actually, after training, we got the vector representation for each word, so we denoted each class as the mean vector of the words in each class. Given an instance and one of its classes, we computed the cosine similarity between their vectors. If the similarity is greater than a fixed threshold ^{2} , then the class is inferred as the type of the given instance.
- **Attributes Similarity (AS):** Given an instance and one of its corresponding classes, this method directly calculates the Jaccard Similarity between their extracted attribute sets. If the similarity is greater than a fixed threshold ^{3} , then the class is inferred as the type of the given instance.
- **Simplified Version of Our Approach (S-ARGW):** In this method, after extracting the attributes of instances and those of classes, we replaced the improved random graph walk model with the original one (proposed in Section 3.2.1) without leveraging each instance's most similar instances that have attributes.

4.1.3. Parameter Tuning

In the experiment, we applied 5-fold cross validation to train parameters in our approach and the comparison methods. One kind of parameters is the thresholds for inferring whether there exists a *TypeOf* relation from a class to an instance in WE, AS, S-ARGW and ARGW. The other kind is the constant α (in Equation (7) and (8)) used in ARGW. Here, we recorded the F1-score when varying the threshold β of WE or γ of AS. The varying process for both β of WE and γ of AS is starting from 0 with a step size of 0.05. For the thresholds θ_1 of S-ARGW and θ_2 of ARGW, we observed that they are negatively correlated with the number *n* of classes for the given instance. In other words, the more classes occurring in the Wikipedia page of the given instance, the greater possibility of more types existing in these classes, the more types can be taken as destinations of the random walk in S-ARGW and ARGW, the lower normalized probability of reaching each class by the random walk. Hence, we recorded the F1-score when varying $\theta_1 \times n$ or $\theta_2 \times n$ with the constant α. The varying process for $\theta_1 \times n$ and $\theta_2 \times n$ is also starting from 0 with a step size of 0.05, and α is changed with a step size of 0.1 (starting from 0.1 to 0.9).

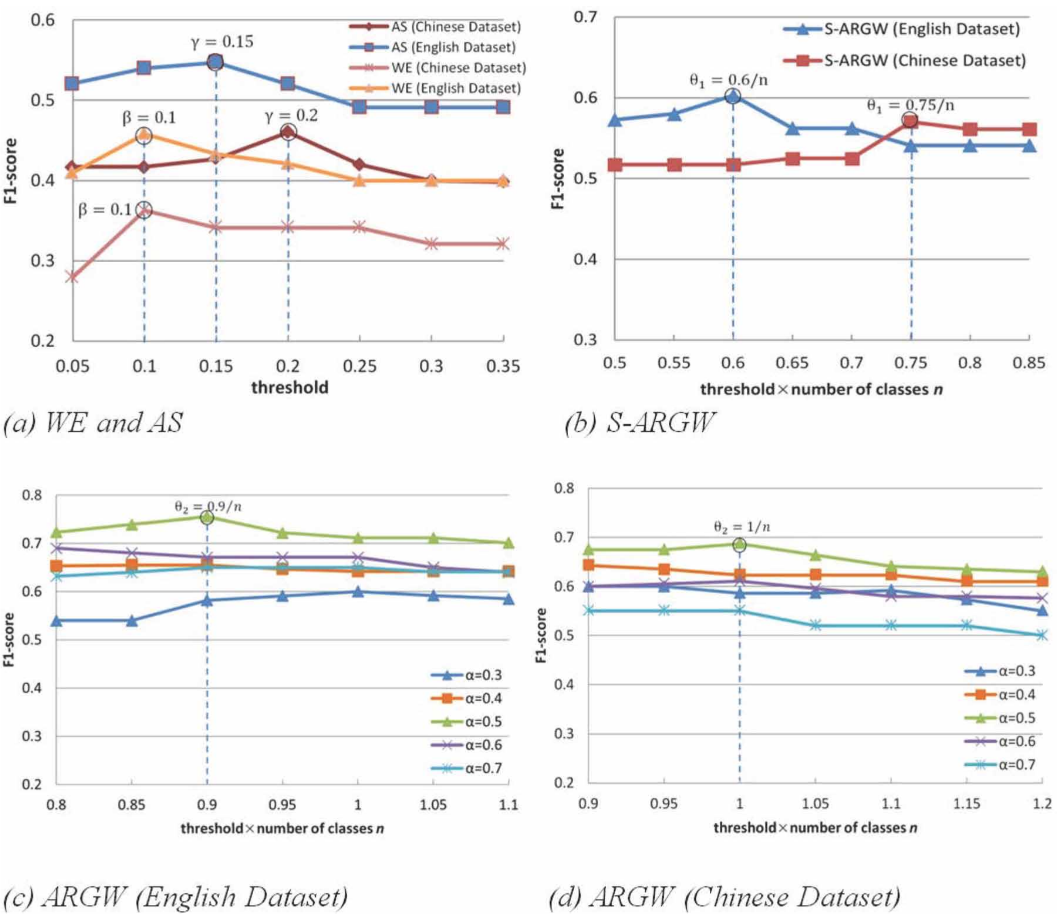
Figure 7 shows the final results of the parameters when F1-score reaches the peak (we omitted some unimportant parts of the tuning results due to the poor performance and limited space). For WE, the threshold β was set to 0.1 on both the English and Chinese dataset. For AS, the threshold γ was set to 0.15 and 0.2 on the English dataset and Chinese dataset, respectively. For S-ARGW, the threshold θ_1 was set to $0.6 / n$ and $0.75 / n$ respectively on the English dataset and Chinese dataset. For ARGW, we set the threshold $\theta_2 = 0.9 / n$ with the constant $\alpha = 0.5$ on the English dataset, and $\theta_2 = 1 / \mathrm{n}$ also with $\alpha = 0.5$ were applied to the Chinese dataset.

4.1.4. Overall Performance and Analysis

In our approach (i.e. ARGW) and the comparison methods WE, AS and S-ARGW, we need to extract the attributes of instances and those of classes. Here, we utilized the strategies of attribute extraction proposed in Section 3.1 to finish this task. For each instance when applying ARGW, to ensure the quality of type inference, we used at most its top-5 similar instances that have attributes. These similar instances are acquired by the method presented in Section 3.2.2. In another comparison method HR, we applied the head finding method proposed in (Ponzetto & Strube, 2007) to parse the headword noun of each class. Table 2 gives the overall results of our approach and the comparison methods on the labeled dataset, and we can see that:

- ARGW outperforms other approaches in all the evaluation metrics, which proves that our approach can better perform language-independent type inference of the instances from Wikipedia. All attribute-based approaches (i.e. AS, S-ARGW and ARGW) outperform the embedding-based approach WE in all evaluation metrics and the rule-based approach HR in precision, which demonstrates the positive impacts of our introduced language-independent feature: attributes. Since some instances and classes are short of attributes, some attribute-based approaches (i.e. AS and S-ARGW) are worse than the rule-based approach HR in recall and F1-score. However, our approach ARGW outperforms HR in these two evaluation metrics, which reflects the value of the most similar instances that have attributes for type inference.
- ARGW is better than S-ARGW in both precision and recall. This is because the attributes of the obtained most similar instances that can actually complement the attributes for two kinds of instances. One kind is the instances that already have attributes, but not complete. Thus, the attributes from the most similar instances can help this kind of instances infer types more accurately, which results in the improvement in precision. The other kind is the instances which have no attribute. ARGW can help this kind of instances perform type inference, which improves recall. In addition, ARGW and S-ARGW consistently outperform AS in precision, recall and

Figure 7. The results of parameter tuning in WE, AS, S-ARGW and ARGW



(a) WE and AS

(b) S-ARGW

(c) ARGW (English Dataset)

(d) ARGW (Chinese Dataset)

Table 2. Overall results of type inference (%)

Approach	English Dataset			Chinese Dataset		
	P	R	F1	P	R	F1
HR	83.9	53.1	65.0	89.4	48.1	62.5
WE	77.1	32.6	45.8	80.5	23.4	36.3
AS	85.2	40.3	54.7	91.8	30.8	46.1
S-ARGW	90.3	45.3	60.3	95.4	40.6	57.0
ARGW	**91.7**	**64.3**	**75.6**	**98.1**	**52.8**	**68.7**

F1-score. It means that applying our proposed attribute-driven type inference assumption to type inference is reasonable and more effective.

- Another interesting phenomenon is that our approach ARGW on the Chinese dataset achieves higher precision than that on the English dataset, but the contrary holds for recall. According to the statistics, each instance in datasets can get at least one most similar instance that have attributes, but only part of classes (86% classes in the English dataset and 72% classes in the Chinese dataset) can get attributes using the attribute extraction strategies in Section 3.1. Hence, lower proportion of the Chinese classes that have attributes is the main reason leading to lower recall. Besides, we found that each attribute is averagely shared by different number of classes in Wikipedia of different languages (718 classes on average per attribute in English Wikipedia and 77 classes on average per attribute in Chinese Wikipedia). Our attribute-driven type inference assumption relies on representative attributes of each class, and the fewer classes an attribute is shared by, the more representative this attribute is (reflected in Equation (1)). Thus, higher average number of classes per attribute in English Wikipedia leads to lower representativeness of attributes, which may affect the precision of ARGW on the English dataset. It also indicates that our approach can be further improved in computing the representativeness of attributes for each class.

4.2. Evaluation of MulType

With the empirical tuned parameters on the labeled datasets, we applied our proposed approach to the whole English and Chinese Wikipedia. Thus, we obtained 7,571,009 different English *TypeOf* relations and 400,349 different Chinese *TypeOf* relations. These relations were denoted as triples to constitute a KB called MulType. We analyzed the accuracy of the type information in different languages, gave the execution time for all parts of our approach to generate Multype, and compared MulType with other KBs.

4.2.1. Accuracy of the Type Information in MulType

Since there are no ground truths available, we cannot verify the type information in MulType automatically. At the same time, due to the large number of type information, it is also impossible to evaluate them manually. Therefore, we designed a sampling strategy and a labeling process to approximately evaluate the correctness of the type information in MulType.

Sampling. Given all the type information of some language *L*, sampling aims to extract a subset of type information (called samples) which can represent the distribution of the whole given data. We first randomly selected 100 instances from all instances of language *L*. Then, the samples of language *L* consist of the triples representing the *TypeOf* relations between the selected instances and their corresponding classes. This sampling strategy is similar to that of Yago.

Labeling. We used the similar labeling process as that used in Yago and Zhishi.schema. Five graduate students were invited to participate in the labeling process. Each annotator has three choices namely "*Correct*", "*Incorrect*" and "*Unknown*" to label each sample. After each student labeled all the samples, we computed 1) the Fleiss´ Kappa to evaluate the annotation consistence (Fleiss, 1971); 2) the average precision with the Wilson interval (Brown et al., 2001) at $\pm = 5\%$ to generalize our findings on the samples to the whole type information of language *L*.

We applied the sampling strategy and the labeling process to the English and Chinese type information in MulType. The following results show the high accuracy of the inferred type information in MulType:

- **Accuracy of English Type Information**. Given 100 randomly selected English instances, we got 582 samples in total. After labeling, the average number of "*Correct*" votes is 534, the precision achieves 91.42% ± 2.25% and the Fleiss´ Kappa is 0.796.
- **Accuracy of Chinese Type Information**. Given 100 randomly selected Chinese instances, we got 641 samples. The average number of "*Correct*" votes is 622, the precision is 96.74% ± 1.33% and the Fleiss´ Kappa is 0.741.

Table 3. Execution time for all parts in our approach to generate MulType

Each Part of Our Approach	Execution Time (hour) of Each Part	
	English Type Information	Chinese Type Information
Instance Attribute Extraction	1.93	0.24
Class Attribute Extraction	0.72	0.21
Most Similar Instances Acquisition	8.23	2.10
Type Inference	1.79	0.37
in Total	12.67	2.92

4.2.2. Execution Time for Generating MulType

We sequentially ran each part of our approach on English and Chinese Wikipedia, and used a Linux server with Intel Xeon E5-2630 v4 2.20 GHz CPU and 256GB memory. The execution time for generating MulType is given in Table 3. From this table, we can find:

- The most time-consuming part is most similar instances acquisition. This is because it uses the whole English (or Chinese) Wikipedia articles as the training corpus for the word2vec model to generate the vector representation of each instance.
- The execution time for generating English type information is much greater than that for generating Chinese type information. The main reason is that the instances, classes and attributes in English Wikipedia are much more than those in Chinese Wikipedia.
- Generating MulType (including English and Chinese versions) only needs 12.67+2.62=15.29 hours by sequentially executing each part of our approach on English and Chinese Wikipedia. The total execution time is acceptable, and it also has much room for improvement if we can parallelize some tasks in all parts of approach.

4.2.3. Comparison with Other Knowledge Bases

To our knowledge, MulType is the first work to publish large-scale Chinese type information as the open data. MulType contains 400,349 different Chinese *TypeOf* relations, 216,727 different Chinese typed instances and 25,406 different Chinese types. Since the existing open KBs do not contain Chinese type information, we only compared the English type information in MulType with that in the well-known KBs namely DBpedia, Yago and LHD. The English *TypeOf* relations in these three KBs are also mined from Wikipedia. We computed the *TypeOf* relation overlap, typed instance overlap and type overlap between MulType and other KBs by exact string matching of the labels for instances and types. The results are given in Table 4.

In MulType, there exist 7,571,009 different English *TypeOf* relations, 3,207,668 different English typed instances and 475,148 different English types. Compared with DBpedia, Yago and LHD, MulType contains the second largest number of English *TypeOf* relations. The overlap of English *TypeOf* relations between MulType and other KBs is not large, which indicates that the English *TypeOf* relations generated by our approach are good supplements to the existing KBs. For MulType, after removing the existing English *TypeOf* relations in DBpedia, Yago and LHD, we found MulType contributes 3,927,727 new English *TypeOf* relations in total.

The number of English typed instances in MulType (ranks the third) and those in other KBs are at the same order of magnitude, i.e more than 3 million. On the basis of the typed instance overlap, MulType can at least contribute more than 0.27 million new typed instance to each of other KBs. Besides, MulType has the largest number of English types and there are 148,395 new English types

Table 4. Comparison results between the English type information in MulType and that in other KBs

	MulType	DBpedia	Yago	LHD
TypeOf Relation Number	7,571,009	3,121,552	14,164,808	6,505,492
TypeOf Relation Overlap	/	80,693	3,521,747	123,531
Typed Instance Number	3,207,668	3,121,054	3,632,793	3,618,094
Typed Instance Overlap	/	2,445,095	2,928,265	2,749,709
Type Number	475,148	401	445,337	47,626
Type Overlap	/	29	326,753	3,648
Average Types per Instance	2.36	1.00	3.90	1.80

with singular headword nouns, which bring a lot of new English *TypeOf* relations compared with Yago. This also reflects the value of our proposed approach. Since the fact that many instances play different roles in real world, each instance may correspond to not only one type. Hence, for each given KB, we computed the average types of each English instance, and MulType also performs better than DBpedia and LHD in this point.

5. WEB ACCESS TO MULTYPE

To publish MulType as the open data for public access, we not only offer the downloading[4] of the whole data, but also provide lookup service and SPARQL endpoint. Lookup service allows querying with the Wikipedia URLs of instances. SPARQL endpoint provides a more professional querying way using SPARQL language.

5.1. Linked Data

According to the Linked Data principles[5], MulType creates URIs for all instances and types, and provides sufficient information when someone looks up a URI by the HTTP protocol. In order to represent instances and types of different languages, we design a URI pattern to indicate what the language of an instance or a type is. This pattern http://www.multype.org/[language]/[datatype]/[label] is composed of four parts. The first part http://www.multype.org/ is the default namespace. The second part specifies the language of an instance or a type. If the given instance (or type) is English (or Chinese), the second part is en (or zh). The third part shows the given URI represents an instance or a type. The last part is the label of an instance (or a type). For English labels, we replace the spaces between words with underlines. In addition, to generate legal URIs as "*href*" values for common Web browsing, we encode non-ASCII characters in the last part into UTF-8[6]. Two examples of the URIs in MulType are as follows:

- Given the English instance "Albert Einstein", its URI is denoted as http://www.multype.org/en/instance/Albert_Einstein.
- Given the Chinese type "动物 (*Animal*)", its corresponding URI is http://www.multype.org/zh/type/%E5%8A%A8%E7%89%A9, where %E5%8A%A8%E7%89%A9 is the UTF-8 format of the Chinese label "动物".

When publishing MulType, we follow the best practice recipes (Berrueta et al., 2008) and try to reuse the existing RDF vocabularies which have clear semantics and are widely used. rdf:label is used as the predicate to connect each URI to its corresponding label. Since there is no existing RDF vocabulary for the *TypeOf* relation, we store its inverse relation leveraging rdf:type. For example, the following triple <http://www.multype.org/en/instance/Albert_Einstein> rdf:type <http://www.multype.org/en/type/Jewish_physicists> represents that there exists a *TypeOf* relation from "*Jewish physicists*" to "*Albert Einstein*".

5.2. Lookup Service

To encourage non-Semantic Web community users to browse the type information in MulType, we provide a simple lookup service[7], where users can directly submit the Wikipedia URL of some instance to acquire type information. For example, if a user wants to know the types of the instance "*MySQL*", he or she can query with its corresponding Wikipedia URL https://en.wikipedia.org/wiki/MySQL. The returned types "*Cross-platform software*", "*Client-server database management systems*", "*RDBMS software for Linux*", "*Free database management systems*", "*Sun Microsystems software*" and "*Oracle software*" are shown in Figure 8.

5.3. SPARQL Endpoint

SPARQL Endpoint[8] is also provided for querying MulType. This querying way is appropriate for the users who know in advance exactly what information is needed. These users can submit customized queries to this endpoint over the SPARQL protocol[9]. We use AllegroGraph RDFStore[10] to store type information and provide querying capabilities.

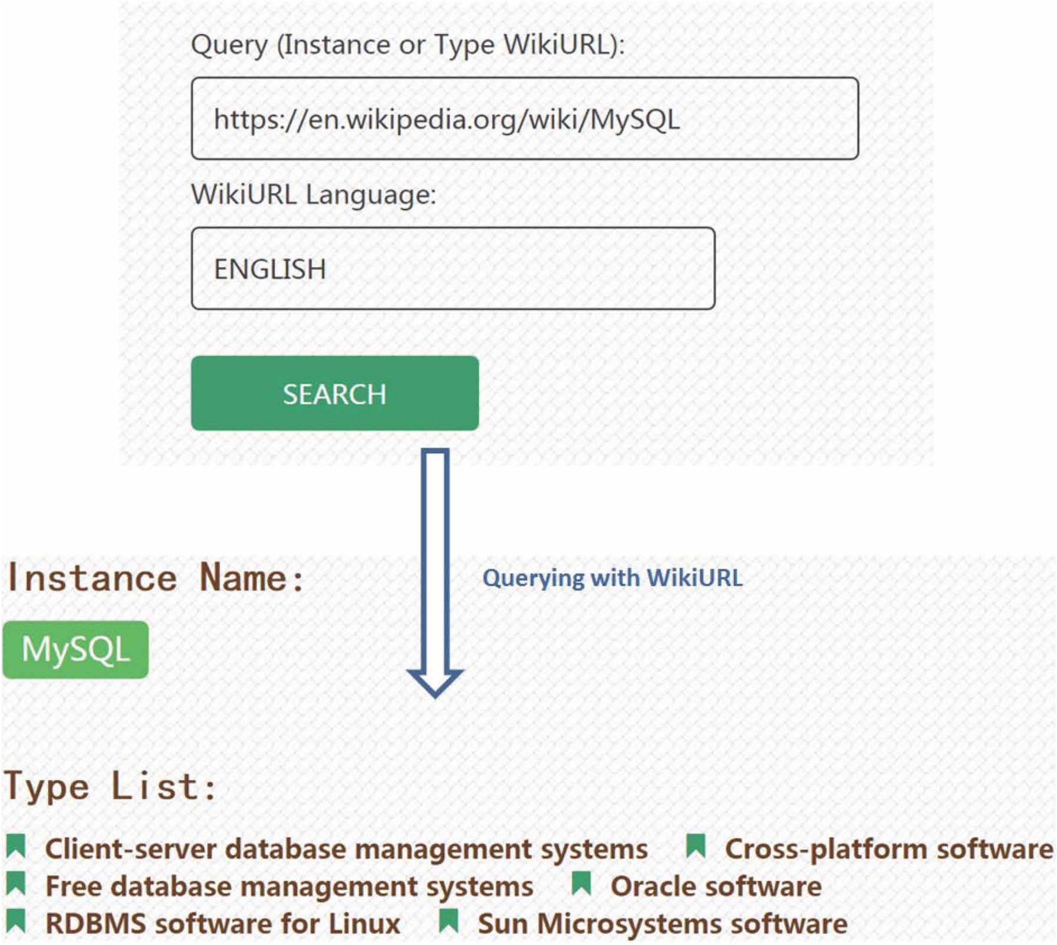
6. CONCLUSIONS AND FUTURE WORK

In this paper, we presented a language-independent approach to infer the types of instances from multilingual Wikipedia. The proposed approach contains two parts. The first part is attribute extraction, which aims to generate the attributes of instances directly from infoboxes, and the attributes of classes with a general iterative algorithm using several language-independent rules. The second part is type information generation, which uses an adaptation of the random walk algorithm to infer the types of instances. After applying the proposed approach to the English and Chinese Wikipedia, we got the first version of MulType, a knowledge base describing the types of instances from multilingual Wikipedia. Different ways of Web access to MulType have been provided to users, including the downloading of data dump, lookup service and SPARQL endpoint.

The experiments about evaluating the effectiveness of our approach have been conducted on two datasets of different languages. The results showed that our approach outperforms the state-of-the-art baselines in different evaluation metrics. We also designed some other experiments on evaluating the quality of type information in MulType. These results not only demonstrated that the type information in MulType achieves a high accuracy, but also verified that there exists a large scale of new English and Chinese type information which are not covered by the existing KBs.

In the future, we first consider applying our proposed approach to the Wikipedia of other languages, such as Arabic, Spanish and German. Besides, in order to add emerging type information into MulType, we will perform live update of MulType by continuous parsing the dumps of multilingual Wikipedia[11]. Regarding to our approach, since it relies on the coverage of the attributes of instances and those of classes, we plan to extract attributes for more instances and classes from Web documents and complement attributes leveraging crowdsourcing related techniques. To better acquire the results of type inference leveraging the proposed attribute-driven type inference assumption, we need to study how to measure the representativeness of attributes for each class without depending exclusively on the number of classes which each attribute is shared by.

Figure 8. An example of lookup service



Query (Instance or Type WikiURL): https://en.wikipedia.org/wiki/MySQL

WikiURL Language: ENGLISH

Instance Name:

MySQL

Type List:

- Client-server database management systems
- Cross-platform software
- Free database management systems
- Oracle software
- RDBMS software for Linux
- Sun Microsystems software

REFERENCES

Abhishek, A., Anand, A., & Awekar, A. (2017). Fine-grained entity type classification by jointly learning representations and label embeddings. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics* (Vol. 1, pp. 797-807). Association for Computational Linguistics. doi:10.18653/v1/E17-1075

Auer, S., & Lehmann, J. (2007). What have innsbruck and leipzig in common? extracting semantics from wiki content. In *European Semantic Web Conference* (pp. 503-517). Springer. doi:10.1007/978-3-540-72667-8_36

Berrueta, D., Phipps, J., Miles, A., Baker, T., & Swick, R. (2008). Best practice recipes for publishing rdf vocabularies (Working draft). *W3C*.

Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., & Hellmann, S. (2009). Dbpedia-a crystallization point for the web of data. *Journal of Web Semantics*, *7*(3), 154–165. doi:10.1016/j.websem.2009.07.002

Brown, L. D., Cai, T. T., & DasGupta, A. (2001). Interval estimation for a binomial proportion. *Statistical Science*, 101–117.

De Melo, G., & Weikum, G. (2010). Menta: Inducing multilingual taxonomies from Wikipedia. In *Proceedings of the 19th ACM International Conference on Information and Knowledge management* (pp. 1099-1108). ACM.

Del Corro, L., Abujabal, A., Gemulla, R., & Weikum, G. (2015). Finet: Context-aware fine-grained named entity typing. *Paper presented at the Proceedings of EMNLP.* Association for Computational Linguistics.

Ding, J., Ding, W., Hu, W., & Qu, Y. (2015). An ebmc-based approach to selecting types for entity filtering. *Paper presented at the Proceedings of AAAI* (pp. 88-94).

Dowty, D. R., Wall, R., & Peters, S. (1981). *Introduction to Montague semantics*. Springer Netherlands.

Fleischman, M., & Hovy, E. (2002). Fine grained classification of named entities. In *Proceedings of the 19th International Conference on Computational linguistics* (Vol. 1, pp. 1-7). Association for Computational Linguistics.

Fleiss, J. L. (1971). Measuring nominal scale agreement among many raters. *Psychological Bulletin*, *76*(5), 378–382. doi:10.1037/h0031619

Gangemi, A., Nuzzolese, A. G., Presutti, V., Draicchio, F., Musetti, A., & Ciancarini, P. (2012). Automatic typing of DBpedia entities. In *Proceedings of the International Semantic Web Conference* (pp. 65-81). Springer.

Hearst, M. A. (1992). Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th Conference on Computational linguistics* (Vol. 2, pp. 539-545). Association for Computational Linguistics. doi:10.3115/992133.992154

Hepp, M. (2008). Goodrelations: An ontology for describing products and services offers on the web. *Paper presented at the Proceedings of EKAW*. doi:10.1007/978-3-540-87696-0_29

Kalyanpur, A., Murdock, J. W., Fan, J., & Welty, C. (2011). Leveraging community-built knowledge for type coercion in question answering. In *Proceedings of the International Conference on Knowledge Engineering and Knowledge Management* (pp. 329-346). Springer. doi:10.1007/978-3-642-25093-4_10

Kliegr, T. (2015). Linked hypernyms: Enriching dbpedia with targeted hypernym discovery. *Journal of Web Semantics*, *31*, 59–69. doi:10.1016/j.websem.2014.11.001

Kliegr, T., & Zamazal, O. (2014). Towards linked hypernyms dataset 2.0: Complementing dbpedia with hypernym discovery and statistical type inference. In *Proceedings of The Ninth International Conference on Language Resources and Evaluation.*

Kliegr, T., & Zamazal, O. (2016). Lhd 2.0: A text mining approach to typing entities in knowledge graphs. *Journal of Web Semantics*, *39*, 47–61. doi:10.1016/j.websem.2016.05.001

Lee, T., Wang, Z., Wang, H., & Hwang, S.-w. (2013). Attribute extraction and scoring: a probabilistic approach. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)* (pp. 194-205). IEEE.

Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., & Bizer, C. et al. (2015). DBpedia - a large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web*, *6*(2), 167–195.

Lin, T., Mausam, & Etzioni, O. (2012). No noun phrase left behind: detecting and typing unlinkable entities. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning* (pp. 893-903). Association for Computational Linguistics.

Ling, X., & Weld, D. S. (2012). Fine-grained entity recognition. *Paper presented at the Proceedings of AAAI* (pp. 94-100).

Ma, Y., Cambria, E., & Gao, S. (2016). Label embedding for zero-shot fine-grained named entity typing. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers* (pp. 171-180)..

Mahdisoltani, F., Biega, J., & Suchanek, F. M. (2015). Yago3: A knowledge base from multilingual wikipedias. *Paper presented at the Proceedings of CIDR.*

Melo, A., Völker, J., & Paulheim, H. (2017). Type prediction in noisy RDF knowledge bases using hierarchical multilabel classification with graph and latent features. *International Journal of Artificial Intelligence Tools*, *26*(02), 1760011. doi:10.1142/S0218213017600119

Meyer, B. (1988). Object-oriented software construction (Vol. 2). New York: Prentice Hall.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *Paper presented at the ICLR Workshop.*

Nakashole, N., Tylenda, T., & Weikum, G. (2013). Fine-grained semantic typing of emerging entities. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics* (Vol. 1, pp. 1488-1497)..

Navigli, R., & Ponzetto, S. P. (2012). BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, *193*, 217–250. doi:10.1016/j.artint.2012.07.001

Nuzzolese, A. G., Gangemi, A., Presutti, V., & Ciancarini, P. (2012). Type inference through the analysis of Wikipedia links. *Paper presented at the Proceedings of LODW.*

Pasca, M. (2007). Organizing and searching the world wide web of facts-step two: harnessing the wisdom of the crowds. In *Proceedings of the 16th International Conference on World Wide Web* (pp. 101-110). ACM.

Pasca, M., & Van Durme, B. (2007). What you seek is what you get: extraction of class attributes from query logs. *Paper presented at the Proceedings of IJCAI* (Vol. 7, pp. 2832-2837).

Pasca, M., & Van Durme, B. (2008). Weakly-supervised acquisition of open-domain classes and class attributes from web documents and query logs. *Paper presented at the Proceedings of ACL* (pp. 19-27).

Paulheim, H., & Bizer, C. (2013). Type inference on noisy rdf data. In *International Semantic Web Conference* (pp. 510-525). Springer.

Paulheim, H., & Bizer, C. (2014). Improving the quality of linked data using statistical distributions. *International Journal on Semantic Web and Information Systems*, *10*(2), 63–86. doi:10.4018/ijswis.2014040104

Ponzetto, S. P., & Strube, M. (2007). Deriving a large scale taxonomy from Wikipedia. *Paper presented at the Proceedings of AAAI* (Vol. 7, pp. 1440-1445).

Ponzetto, S. P., & Strube, M. (2008). Wikitaxonomy: a large scale knowledge resource. *Paper presented at the Proceedings of ECAI* (Vol. 178, pp. 751-752).

Rahman, A., & Ng, V. (2010). Inducing fine-grained semantic classes via hierarchical and collective classification. In *Proceedings of the 23rd International Conference on Computational Linguistics* (pp. 931-939). Association for Computational Linguistics..

Ren, X., He, W., Qu, M., Huang, L., Ji, H., & Han, J. (2016). AFET: Automatic fine-grained entity typing by hierarchical partial-label embedding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* (pp. 1369-1378). doi:10.18653/v1/D16-1144

Stojanovic, L. (2004). *Methods and tools for ontology evolution* [PhD thesis]. Karlsruhe Institute of Technology, Germany.

Suchanek, F. M., Kasneci, G., & Weikum, G. (2008). Yago: A large ontology from Wikipedia and Wordnet. *Journal of Web Semantics*, *6*(3), 203–217. doi:10.1016/j.websem.2008.06.001

Tonon, A., Catasta, M., Demartini, G., Cudré-Mauroux, P., & Aberer, K. (2013). Trank: Ranking entity types using the web of data. In *International Semantic Web Conference* (pp. 640-656). Springer. doi:10.1007/978-3-642-41335-3_40

Wang, H., Wu, T., Qi, G., & Ruan, T. (2014). On publishing Chinese linked open schema. In *International Semantic Web Conference* (pp. 293-308). Cham: Springer.

Wu, T., Qi, G., & Wang, H. (2014) Zhishi. schema explorer: A platform for exploring Chinese linked open schema. In *Chinese Semantic Web and Web Science Conference* (pp. 174-181). Springer.

Yosef, M. A., Bauer, S., Hoffart, J., Spaniol, M., & Weikum, G. (2012). Hyena: Hierarchical type classification for entity names. In *Proceedings of COLING 2012: Posters* (pp. 1361-1370).

Yosef, M. A., Bauer, S., Hoffart, J., Spaniol, M., & Weikum, G. (2013). Hyena-live: Fine-grained online entity type classification from natural-language text. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations* (pp. 133-138).

ENDNOTES

[1] Note that class and category are synonyms in this work, and both of them refer to the topic of some instance (i.e. an article in Wikipedia). Given an instance, its classes (i.e. categories) are treated as candidate types, and if there exists a *TypeOf* relation from a class to an instance, then the class can be seen as the type of the instance.

[2] https://en.wikipedia.org/wiki/List_of_languages_by_total_number_of_speakers

[3] https://github.com/jxls080511/123105

[4] http://www.multype.org/Download.jsp

[5] http://www.w3.org/DesignIssues/LinkedData.html

[6] http://www.utf-8.com/

[7] http://www.multype.org/LookUp.jsp

[8] http://www.multype.org/SPARQL.jsp

[9] https://www.w3.org/TR/sparql11-protocol/

[10] http://www.franz.com/agraph/allegrograph/

[11] https://dumps.wikimedia.org/backup-index.html

Tianxing Wu received the master's degree in software engineering from Southeast University, China, in 2014. He is currently working toward the PhD degree in the School of Computer Science and Engineering, Southeast University, China. He is a recipient of the Chinese National Scholarship for PhD students. His research interests include Artificial Intelligence, Knowledge Graph and Semantic Web.

Guilin Qi is a professor working at Southeast University in China. He received his PhD in Computer Science from Queen's University of Belfast in 2006. Before he moved back to China, he has worked in Institute AIFB at University of Karlsruhe for three years. His research interests include knowledge representation and reasoning, uncertainty reasoning, and semantic web. He has published over 100 papers in these areas, many of which published in proceedings of major conferences or journals. He is the principal investigator of three national natural science foundation of China projects. He is in the editorial board of the Journal of Web Semantics, which is one of the top journals in Semantic Web. He is an associate editor of the Journal of Advances in Artificial Intelligence and has co-edited a special issue of Annals of Mathematics and Artificial Intelligence. He is a co-Chair of the poster and demo track of EKAW'14 and was a co-Chair of the poster and demo track of ISWC'11. He was invited to give a lecture at Reasoning Web summer school in Vienna in 2012 and was invited to give a couple of talks in some famous Chinese universities. He gave several tutorials in some top conferences, such as AAAI'10 and ISWC'11.

Bin Luo received the bachelor's degree in software engineering from Changshu Institute of Technology, China, in 2014. He is currently working toward a master's degree in the School of Computer Science and Engineering, Southeast University, China. His research interests include Knowledge Graph and NLP techniques.

Lei Zhang received his PhD degree from Karlsruhe Institute of Technology (KIT), Germany, in 2017. He has been mainly working on the topics of text mining, natural language processing, information retrieval and semantic search. His interdisciplinary work has been published in more than 20 top-level journals and conference proceedings and earned the second prize at the Semantic Web Challenge 2015 and the Best Student Research Paper at ISWC 2016.

Haofen Wang received his PhD from Shanghai Jiao Tong University in 2013. He worked as an assistant professor in East China University of Science and Technology from 2014 to 2016. Now he is the CTO of Shenzhen Gowild Robotics Co. Ltd., China. His research interests include Knowledge Graph, Chat Bots and Semantic Web. He has published more than 70 papers in high-level journals and conferences. He has earned the second prize in the Billion Triple Challenge of ISWC 2008 and the first prize on the task of instance matching of OAEI 2011. He is serving or ever served as a PC member for some leading international conferences, such as AAAI, WWW, ISWC, etc.