Cross-Lingual Type Inference

Bo Xu[1], Yi Zhang[1], Jiaqing Liang[1], Yanghua Xiao[1,2(✉)], Seung-won Hwang[3], and Wei Wang[1]

[1] School of Computer Science, Fudan University, Shanghai, China
{xubo,z_yi11,shawyh,weiwang1}@fudan.edu.cn, l.j.q.light@gmail.com
[2] Shanghai Internet Big Data Engineering and Technology Center, Shanghai, China
[3] Department of Computer Science, Yonsei University, Seoul, South Korea
seungwonh@yonsei.ac.kr

Abstract. Entity typing is an essential task for constructing a knowledge base. However, many non-English knowledge bases fail to type their entities due to the absence of a reasonable local hierarchical taxonomy. Since constructing a widely accepted taxonomy is a hard problem, we propose to type these non-English entities with some widely accepted taxonomies in English, such as DBpedia, Yago and Freebase. We define this problem as cross-lingual type inference. In this paper, we present CUTE to type Chinese entities with DBpedia types. First we exploit the cross-lingual entity linking between Chinese and English entities to construct the training data. Then we propose a multi-label hierarchical classification algorithm to type these Chinese entities. Experimental results show the effectiveness and efficiency of our method.

1 Introduction

With the boost of Web applications, WWW has been flooded with information on an unprecedented scale, most of which is readable only by human but not by machine. To make the machine understand the Web, great efforts have been dedicated to harvesting knowledge from the online encyclopedias, such as Wikipedia. A variety of knowledge graphs or knowledge bases thus have been constructed, such as Yago [21], DBpedia [2] and Freebase [3]. These knowledge bases contain different semantic relationships between entities and concepts (also known as types or categories).

A fundamental semantic information about entities is their types. The relationships between an entity and its types represent the instanceOf relationships. For example, William Shakespeare has the types Person, Writer, Poet, etc. Among these types, *fine-grained* types such as Writer and Poet are more important than *coarse-grained* types such as Person because they characterize

Y. Xiao—This paper was supported by the National NSFC (No. 61472085, 61171132, 61033010, U1509213), by National Key Basic Research Program of China under No. 2015CB358800, by Shanghai Municipal Science and Technology Commission foundation key project under No. 15JC1400900. Seung-won Hwang was supported by Microsoft.

S.B. Navathe et al. (Eds.): DASFAA 2016, Part I, LNCS 9642, pp. 447–462, 2016.
DOI: 10.1007/978-3-319-32025-0_28

an entity more accurately. Characterizing an entity with types especially with fine-grained types plays a more and more important role in many real applications, such as recommender systems [12], question answering [10], emerging entities discovering [11], named entity disambiguation [24], etc. We refer to this effort as *Entity Typing*. Different from traditional `Named Entity Recognition (NER)` task, which tends to classify entities into a small set of coarse types, such as `Person`, `Location` and `Organization` [17], `Entity Typing` task tends to assign *specific* types to their entities.

A direct solution to type an entity is assigning it to the type that the entity most likely belongs to. The likelihood can be estimated by the similarity between the entity and the entities already with the type. Clearly, this naïve solution relies on a conceptual hierarchy (which contains the *instanceOf* relation between entities and their categories) and the description about entities. However, for non-English entities, typing them is still difficult due to the lack of well-structured non-English knowledge bases, especially the *instanceOf* knowledge. Although many non-English knowledge bases contain category information about entities, it cannot serve as *instanceOf* knowledge for entity typing. We use Baidu Baike, the largest Chinese knowledge repository, as an example to illustrate their weaknesses:

- First, categories[1] in non-English knowledge bases such as Baidu Baike are actually entities' tags/topics instead of the exact types. One of the categories of entity `AK47` in Baidu Baike is 军事 (en: Military), which is the topic of the entity instead of its type.
- Second, some entities contain error categories. For example, one of the categories of entity `AK47` in Baidu Baike is 军事人物 (en: Military Person), which is wrong.
- Third, types in non-English knowledge bases can hardly be organized as a hierarchy. Even though some categories are really types of entities, many desired relationships among categories such as `subClassOf` are still sparse in non-English knowledge bases. As a result, we are blind about the granularity of a type as well as their `subClassOf` relationships, which makes the entity typing with specific types difficult.

We notice that many English knowledge bases have been available and they contain widely acknowledged `instanceOf` knowledge. Thus, we wonder *whether we can use the English knowledge bases to type an non-English entity*. We refer to this problem as *cross-lingual type inference*. In this paper, we focus on typing Chinese entities with English DBpedia knowledge bases.

[1] In this paper, we strictly differentiate "category" from "type". "Category" always refer to the part of the knowledge base such as Baidu Baike and Wikipedia named as "category". Most of these categories actually are only tags of an entity. Instead "type" always refer to the class that an entity can be classified into.

Motivation. We highlight that cross-lingual type inference is sufficiently motivated:

- First, reuse of types in English knowledge bases ensures the high quality of types. Many well-established knowledge bases are available in English, such as DBpedia [2] and Yago [21]. These knowledge bases provide richer and cleaner types, which are widely and successfully used in many real applications.
- Second, types from these well-established English knowledge bases form a hierarchy consisting of `subClassOf` relations between types. It enables specificity-aware entity typing, which has the flexibility to be adapted in different applications demanding types with different specificity.
- Third, cross-lingual type inference enables many cross-lingual search tasks. For example, suppose we are looking for `all birds from all over the world`. Since many birds may only appear in a local knowledge base, we need to integrate the knowledge from different knowledge repositories. However, due to knowledge bases in a specific language tend to name the type `birds` in its own language, such as 鸟 in Chinese. As a prerequisite, we need to type the entities in different languages with a uniform type. In our example, to retrieve all birds, we need to type the entities of `birds` in different local knowledge bases to the same English type `birds`.

Weakness of Previous Approaches. Many solutions have been proposed to type English entities with English Types. However, some of them [5,6,15] are language-dependent, and cannot be used to solve the cross-lingual type inference problem. `Tipalo` [5] uses natural language processing (NLP) tools to extract types from the definition, and `Yago` [15,21] also finds WordNet types from Wikipedia category names. For instance, they use NLP tools to find category `Michael Jackson albums` belonging to WordNet type `album`, hence all entities belonging to this category can be assigned to the type (`album`) in WordNet. However, with only English entities considered, it cannot be easily adapted to other languages. `SDType` [13] is a state-of-the-art type inference method and can be adopted to solve cross-lingual problem. However, they do not consider the type taxonomy and predict type label independently (i.e. not aware of the relationship between types), which in general will decrease the accuracy of some fine-grained type classifiers.

Challenges and Contributions. A direct solution to determine whether an entity belongs to a type is training a binary classifier for the type. However, this naive solution in general is not applicable in our setting. We still need to solve the following challenges:

- **Construction of Training Data.** To build the classifiers for each type, we need a massive amount of training data. For Chinese entities, there exist no DBpedia types. Manual labeling is costly and not applicable on a large scale of training data. We address this challenge in Sect. 4 by exploiting the cross-lingual linking between Chinese and English entities, and typing these Chinese ones with the types of their corresponding English ones.

- **Efficient-yet-effective Typing Solutions.** There exist hundreds of English types in DBpedia and tens of millions of Chinese entities to type. Classifying an entity using each classifier is obviously wasteful since an entity deserves a very small number of types. Hence, how to use the relationships between types to speed up the typing procedure while ensuring the accuracy of the typing is a challenge. We address this challenge in Sect. 5 by building hierarchical multi-label classifiers and designing a corresponding hierarchy-aware typing algorithm.
- **Sparseness of Types.** Note that *instanceOf* relations in DBpedia are still sparse. For example, *Tom Cruise* only has types *Thing, Agent* and *Person* in DBpedia and many other types such as *Artist* and *Actor* are missing. As a result, many types such as *Artist* in our example miss many members. The classifier built for *Artist* can hardly be effective due to the sparsity of its instances. To solve this problem, we first propose a type completion method as a preprocessing step to find more types for DBpedia entities in Sect. 3.2.

The rest of this paper is organized as follows. In Sect. 2, we formally define the problem of cross-lingual type inference and give an overview of our proposed system. In Sect. 3, we introduce the features we use for Chinese knowledge bases entities and the process of type completion of English DBpedia entities. In Sect. 4, we describe the construction of training data. In Sect. 5, we propose our multi-label hierarchical classification method. In Sect. 6, we present the experimental results. In Sect. 7, we review the related work and highlight the differences between our work and major existing methods. In Sect. 8, we conclude this work.

2 Overview

2.1 Problem Definition

In this section, we first formalize the cross-lingual type inference problem in Definition 1. There are many well-established knowledge bases, which have different taxonomy structures. For instance, the structure of DBpedia is a tree, while Yago is a Directed Acyclic Graph (DAG). According to [18], solutions vary from structure to structure. In this paper, we focus on typing Chinese entities with types in a tree-based taxonomy. Specifically, we type Chinese entities with DBpedia types.

Definition 1 (Cross-Lingual Type Inference). *Let $\mathcal{T}$ be the collection of all types in English knowledge base's taxonomy, and $\mathcal{E}$ be the collection of non-English knowledge base entities. Our problem is typing each non-English entity $e \in \mathcal{E}$ with a subset of types $T(e) \subset \mathcal{T}$.*

DBpedia taxonomy is a tree-based hierarchical type structure. Different from DAG structure, a node can only have one parent node. As shown in Fig. 1, types of DBpedia entities have three properties:

- First, entities have types with different granularities. For instance, `William Shakespeare` is not only a `Person`, but also a `Writer`.

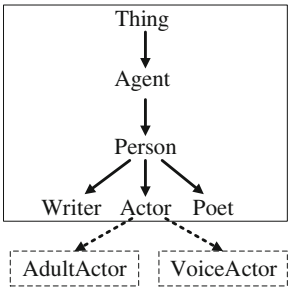


Fig. 1. Black solid square represents the types of entity `William Shakespeare` and their hierarchical relations. `AdultActor` and `VoiceActor` are the sub-types of Actor, but not the types of `William Shakespeare`.

- Second, entities may have multiple types at a certain granularity. For instance, `William Shakespeare` belongs to the types `Writer` and `Poet`, both of which are the sub-types of `Person`.
- Third, entities may not have the most specific types. For instance, `William Shakespeare` belongs to the type `Actor`, but not to any of its sub-types (`AdultActor` and `VoiceActor`).

Hence, typing Chinese entities with DBpedia types is a *tree-based, multi-label, non-mandatory leaf node, hierarchical classification problem.*

2.2 System Architecture

We introduce `CUTE`, which is short for **C**ross-ling**U**al **T**ype inf**E**rence method. Figure 2 is the system architecture of `CUTE`. We first exploit the cross-lingual entity linking between Chinese and English entities to construct the training data. Then we propose a multi-label hierarchical classification algorithm to type these Chinese entities.

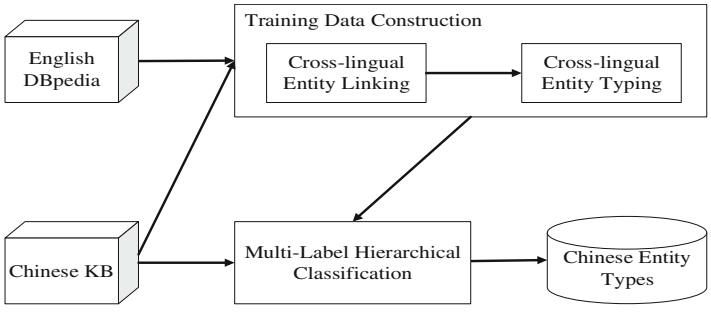


Fig. 2. System architecture of `CUTE`

3 Data

In this section, we first introduce the features used to characterize Chinese entities. Then we present the detail to find more types for DBpedia entities so that the constructed training data (in Sect. 4) is more complete.

3.1 Feature Set

To make our solution general enough so that it can be applicable to type entities in an arbitrary Chinese knowledge base, we only use features that exist in their knowledge bases. We tend to use many features in the structured data of a knowledge part (such as infobox templates, tags) instead of the features extracted from free text such as part-of-speech tags, dependency parsing results, etc. More formally, each entity $e \in \mathcal{E}$ is represented by an $|\mathcal{F}|$-dimension vector:

$$\boldsymbol{e} = (f_1, f_2, ...f_j, ..., f_{|\mathcal{F}|}) \tag{1}$$

where f_j represents the j-th feature of entity e and $\mathcal{F}$ is the full feature set we used. In this paper, each feature is a binary feature, that is we use 1 and 0 to represent the existence of the j-th feature in entity e. Next, we elaborate the three types of features we used: *entity category*, *entity attribute* and *entity attribute-value pair*:

Entity Category: Categories in encyclopedia websites are used to group similar articles, which has been widely used in ontology/taxonomy construction [16,21] and type inference [1]. For instance, Chinese entity 刘德华 (en: Andy Lau) belongs to many categories, such as 歌手 (en: Singer) and 演员 (en: Actors). Using these features, we can easily infer its types.

Entity Attribute: Attributes (also called properties) of entities also play an important role in inferring their types [7,13]. For instance, one may obtain the types `agent` and `person` by 刘德华 (en: Andy Lau)'s attribute 职业 (en: Occupation).

Entity Attribute-Value Pair: By using attribute features, one may get some coarse-grained types. When using attribute-value pair features, we can discover more fine-grained types. For instance, one can find the fine-grained type `Actor` from the attribute-value pair 职业-演员 (en: Occupation-Actor).

Note that some rare features might slow down the training phase and mislead the classifiers. Hence, we only choose the features which are shared by at least β entities. In our experiment, we set the value to 10.

3.2 Type Completion of English DBpedia Entities

Many types are missing for DBpedia entities. The absence will deteriorate the quality of the training data, and in turn hurt the effectiveness of our typing solution. In this subsection, we elaborate how we solve this problem.

As we know, types of DBpedia entities are derived from their infobox template names in Wikipedia [8]. However, many of them only have general infobox templates, hence they may lack some fine-grained types. We refer to it as `Type Incompleteness` problem. For instance, a number of entities, such as `Tom Cruise`, only belong to `Thing`, `Agent` and `Person` in DBpedia, while more specific types such as `Artist` and `Actor` are missing. As a result of using existing DBpedia types to construct the training data, many types such as `Artist` and `Actor` in our example have quite few members. The classifiers of these types will have bad performance. As a result, entity typing with these classifiers will lead to errors.

To find more types for DBpedia entities, we exploit the category information of entities in DBpedia to complete their types. Our basic idea is discovering the `subClassOf` relationship between DBpedia categories and types. If a category c is a subclass of type t, then all entities in category c would belong to type t. Specifically, the type completion process consists of two steps. The first step is to estimate the probability that category c is a subclass of type t ($Pr(c \subset t)$). The second step is to compute the probability that the entity e belongs to type t ($Pr(e \in t)$).

STEP 1: $Pr(c \subseteq t)$. There are two state-of-the-art methods to estimate the probability of $Pr(c \subseteq t)$. One is Yago [21], and the other is PARIS [20]. We employ both methods to estimate the probability.

We first determine the `subClassOf` relations by Yago. The procedure is as follows [21]: We first segment the pre-modifier, head compound and post-modifier of the category name c. After stemming the head compound, we check whether the concatenation of pre-modifier and head compound or the head compound alone is a name of DBpedia type t. If true, we consider category c as a subclass of t. For all categories and types, if category c is a subclass of type t, we set the probability of $Pr_1(c \subset t)$ to 1. Otherwise, we set the probability to 0. Take Wikipedia category `History museums in Ohio` as a example, its pre-modifier, head compound and post-modifier are `History`, `museums` and `in Ohio`, respectively. After stemming the head compound (i.e. `museum`), we find that `museum` is a name of DBpedia type. Thus we set the probability that `History museums in Ohio` belongs to type `museum` to 1.

Then we estimate the probabilistic `subClassOf` relations from PARIS. As shown in Eq. 2 [20], the probability is proportional to the number of entities of category c that belong to type t:

$$Pr_2(c \subseteq t) = \frac{\#c \cap t}{\#c} \tag{2}$$

where $\#c$ is the number of entities of category c, and $\#c \cap t$ is the number of entities of category c that belong to type t.

Finally, we choose the greater one as the final probability. Because any individual one is a strong signal to suggest the *subClassOf* relations among categories.

$$Pr(c \subseteq t) = \max(Pr_1(c \subseteq t), Pr_2(c \subseteq t)) \tag{3}$$

STEP 2: $Pr(e \in t)$. Then, we use a Noisy-or model [19] to estimate the probability that entity e belongs to type t in Eq. 4:

$$Pr(e \in t) = 1 - \prod_{c \in C(e)} (1 - Pr(c \subseteq t)) \tag{4}$$

where $C(e)$ is the categories that entity e belongs to. If the probability of $Pr(e \in t)$ is greater than or equal to a threshold θ $(0 \leq \theta \leq 1)$, we assign the type t to entity e.

4 Training Data Construction

To effectively learn the classifiers, we need a massive amount of labelled data. For Chinese entities, there exist no English DBpedia types, and it is not practical to use human labeling. Hence, we first link Chinese entities and English entities and then use the types of English entities as the label of Chinese entities. In this way, we have many Chinese entities as well as their English types as the training data.

Cross-lingual entity linking in general is non-trivial [4,22,23]. Fortunately, our goal is just linking entities across different languages to construct the training data instead of finding as more cross-lingual entity links as possible. Thus, we are only concerned with the *precision* of the entity linking instead of *recall*. We notice that some entities in English/Chinese knowledge base may have the same Chinese label name. Hence, we only need to compare their Chinese label names of Chinese and English entities. If they share the same label name, we establish a link between them. For example, Chinese entity 威廉·莎士比亚 and English entity `William Shakespeare` share the same Chinese label name 威廉·莎士比亚, and hence should be linked together. Given the linked entity pairs, we directly use English entities' types to label the corresponding Chinese entities. For instance, `William Shakespeare` has 6 types which are used as the English types of corresponding Chinese entity 威廉·莎士比亚.

5 Multi-label Hierarchical Classification

To solve the multi-label hierarchical classification problem, we propose a `local classifier per node` based approach [18]. Specifically, we first train a binary classifier for each type in the hierarchy (except the root node) in the *training phase*. Then we use a top-down search to *type* the entities.

5.1 Training Phase

We first define the set of positive and negative samples for each classifier. We adopt the `sibling policy` proposed in [18] for this purpose. However, the generic policy is designed for *mandatory* leaf node classification problem [18]

and should be customized for our *non-mandatory* leaf node classification problem. More specifically, for each type classifier $TC(t)$, we use all entities belonging to type t as positive samples, and those entities belonging to the sibling or super-type of t but not to type t as negative samples. For type `VoiceActor` in Fig. 1, entities belonging to its sibling `AdultActor` or its super type `Actor` but not to `VoiceActor` serve as negative samples. The rationality of sibling strategy is that in our hierarchical classification, we determine the membership of an entity to types from the root to specific types. That means when an entity e has been typed with type t, we need to find the best subtypes that e belongs to. Thus, we need samples from different subtypes to effectively classify entities into fine-grained types.

Then we need to train all the classifiers. Each of them is a binary classifier. We tested all possible binary classification models, such as Logistic Regression, Random Forest, and SVM. We obtained almost the same performance results by these models. As a result, considering the efficiency, we use the Logistic Regression model implemented by `scikit-learn` [14].

5.2 Typing Phase

Finally, we propose a top-down multi-label hierarchical classification algorithm to type Chinese entities with DBpedia types. The procedure is illustrated in Algorithm 1. The algorithm searches in the DBpedia taxonomy in a top-down manner. We use a queue data structure Q to store the candidate types of entities. For each entity e, we first push all types at level 1 into Q, which are the sub-types of root node of DBpedia taxonomy. Then for each candidate type t in Q, we run the classifier of t to test whether entity e belongs to the type t or not. If the result is `true`, we continue to search all its sub-types by appending its sub-types into Q. The search process ends when no more candidate types to be processed.

6 Experiments

In this section, we present the experimental results. Specifically, we verify the effectiveness of *type completion* and *cross-lingual type inference*.

6.1 Type Completion of English DBpedia Entities

We first justify type completion and determine the best threshold (θ) used for type completion. We use DBpedia2014[2] as our dataset. Specifically, we use `DBpedia Ontology`[3], `Entity Types`[4] and `Entity Categories`[5] as input, and return a set of new entity-type pairs. There are totally 4,191,094 entities with 14,993,020 types in DBpedia, with an average of 3.58 types per entity.

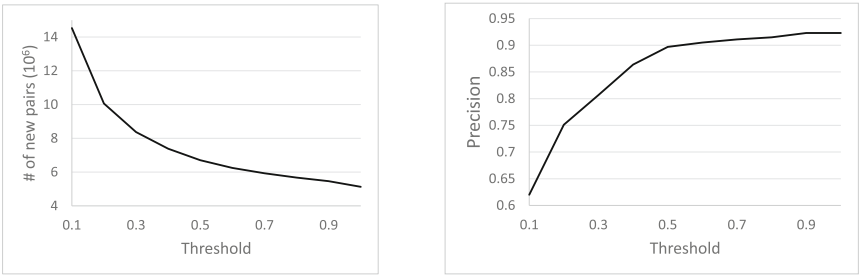
[2] http://oldwiki.dbpedia.org/Downloads2014.
[3] http://oldwiki.dbpedia.org/Downloads2014#dbpedia-ontology.
[4] http://oldwiki.dbpedia.org/Downloads2014#mapping-based-types.
[5] http://oldwiki.dbpedia.org/Downloads2014#articles-categories.

Algorithm 1. The Hierarchical Classification Algorithm.

Input: the feature vector $\boldsymbol{e} = (f_1, f_2, ..., f_{|\mathcal{F}|})$ of entity e, all types T in DBpedia taxonomy and trained classifiers $TC(t)$ for each type $t \in T$.
Output: a collection of types $T(e)$ of entity e

```
Initialize Q to be a queue containing all types at level 1
while Q is not empty do
    t ← Q.dequeue()
    if TC(t|e) == true then
        add t to T(e)
        if t has sub-types then
            add these sub-types to Q
        end if
    end if
end while
```

From Fig. 3(a), we can observe that the number of new entity-type pairs discovered by our approach decreased with the threshold. We also estimate how many newly extracted entity-type pairs are correct. Since no ground truth is available, we resort to using human judgement to evaluate the precision. We randomly select 1000 new entity-type pairs and ask volunteers to judge whether they are correct. From Fig. 3(b), we can see that the precision increases with the threshold. We notice that when the threshold is 0.9, our type completion approach achieves the highest precision (0.923) and finds more new entity-type pairs than threshold 1.0. Hence, we use 0.9 as the best threshold. Finally we obtain 5,463,462 new entity-type pairs.



(a) (b)

Fig. 3. Completion performance with different thresholds

6.2 Cross-Lingual Type Inference

Next we show the effectiveness and efficiency of our method. The Chinese knowledge base is from Baidu Baike[6] (the largest Chinese online encyclopedia)

[6] http://baike.baidu.com/.

articles which were crawled in January 2015. We obtain entity name, categories, attributes and attribute-value pairs information from each article. The English DBpedia data is from DBpedia2014, including `DBpedia Ontology`, `Entity Types`, and `Entity Chinese label names`[7]. Moreover, new entity-type pairs from type completion process are also used. There are overall 9,223,450 Chinese Baidu Baike entities and 4,191,094 English DBpedia entities. 344,576 of English entities have Chinese label names. After cross-lingual entity linking, we obtain English DBpedia types for 93,381 Chinese entities, which is large enough to train our classifiers. To evaluate the performance, we randomly select 50,000 entities for training, and 10,000 ones for test.

Metrics. Notice that our problem is hierarchical classification. Hence, we use hierarchical precision (hP), hierarchical recall (hR) and hierarchical f1-measure (hF) to evaluate the performances of our approach [18]. For an entity $e \in \mathcal{E}$, we denote the set of the ground truth types as t_e and the set of the types found by our approach as p_e. The metric we used for evaluation are defined as follows:

Hierarchical Precision (hP)

$$hP = \frac{\sum_{e \in \mathcal{E}} |t_e \cap p_e|}{\sum_{e \in \mathcal{E}} |p_e|}. \tag{5}$$

Hierarchical Recall (hR)

$$hR = \frac{\sum_{e \in \mathcal{E}} |t_e \cap p_e|}{\sum_{e \in \mathcal{E}} |t_e|}. \tag{6}$$

Hierarchical F1-measure (hF)

$$hF = \frac{2 * hP * hR}{hP + hR}. \tag{7}$$

Comparison with Baseline Methods. We compare our method with baseline methods and the state-of-the-art methods on the test data. The competitors include:

- **SDType.** SDType is a state-of-the-art method proposed by [13]. It uses the same features described in Sect. 3.1.
- **LR.** Logistic Regression Model (LR) is a strong baseline for many tasks. In this competitor, we does not consider the hierarchy structure of types, and train LR models for each type independently. The features are the same as in SDType.
- **CUTE.** The proposed method in this paper.

[7] http://data.dws.informatik.uni-mannheim.de/dbpedia/2014/zh/labels_en_uris_zh.nt.bz2.

Table 1. Comparison results on test data.

Method	hP	hR	hF
SDType	0.81	0.69	0.75
LR	**0.89**	0.69	0.78
CUTE	0.88	**0.71**	**0.79**

Effectiveness. The comparison result is shown in Table 1, from which we can see that our method CUTE performs sufficiently well on the test data, and achieves an hierarchical f1-measure of 0.79. SDType method has the worst performance, since it does not consider the hierarchical structure and has a poor prediction performance when an entity has multiple types at a certain granularity. Our method has a high recall and a comparable precision compared to LR. A closer look at the results reveal that our higher recall can be attributed to the sibling policy we used to define positive and negative samples. This policy helps discover more fine-grained types, leading to a higher recall. To see this, we give some examples of fine-grained types for entities found by LR and CUTE in Table 2.

Table 2. Fine-grained types for entities found by LR and CUTE.

Entity Name	LR	CUTE
长江基建	Agent, Organization	Agent, Organization, Company
功夫杀手	Work	Work, Film
林志玲	Agent, Person, Artist, Actor	Agent, Person, Artist, Actor, Model

Table 3. Running time (seconds) of LR and CUTE. *Predicting (1* K*)* is the runtime to type 1 K randomly selected entities. *Predicting (10* K*)* is the runtime to type all the 10 K entities in the test data set.

Method	Training	Predicting (1 K)	Predicting (10 K)
LR	3,144	40	399
CUTE	745	10	94

Efficiency. We also compare the runtime of LR with that of CUTE on test data. We only compare to LR because the above results show that it is a strong baseline. As shown in Table 3, CUTE is more efficient than LR in both model training and entity typing.

6.3 Typing Chinese Entities

Finally, we use CUTE to type all the Chinese Baidu Baike entities. In total, we get 22,725,365 types for 6,314,273 entities, with an average of 3.6 types per entity. We also estimated the precision (by Eq. 5) by manual evaluation on a sample of 1000 randomly selected entity-type pairs. The precision is 90.2 %.

Since we have typed Chinese entities with DBpedia types, it is possible to compare type distribution of Baidu Baike entities and DBpedia entities. This comparison gives us opportunities to study whether the content of a knowledge base is independent with language used to describe the knowledge. Table 4 shows the top-15 most frequent types of Baike entities. From the table we can see that, the type distributions in Baidu Baike and DBpedia are quite different. For instance, there are 1,056,106 books and 178,689 food entities in Baidu Baike, while there are only 31,029 and 6,337 counterparts, respectively, in DBpedia. This is because there are a large number of novels and food entities in Baidu Baike. Most of them are only famous in China thus are absent in DBpedia. Hence, many entities as well as their knowledge only exist in knowledge bases of local language.

Table 4. Top-15 most frequent types in Baidu Baike, and the type distribution of these types in both Baidu Baike and DBpedia.The percentage after the frequency is the proportion of entities with the type.

Types	Baike Frequency	Rank	DBpedia Frequency	Rank
dbo:Work	2,529,054 (40.1 %)	1	411,295 (9.8 %)	7
dbo:Agent	2,004,923 (31.8 %)	2	1,688,264 (40.3 %)	1
dbo:Person	1,217,988 (19.3 %)	3	1,445,104 (34.5 %)	2
dbo:Place	1,197,263 (19.0 %)	4	735,062 (17.5 %)	3
dbo:WrittenWork	1,098,019 (17.4 %)	5	56,212 (1.3 %)	4
dbo:Book	1,056,106 (16.7 %)	6	31,029 (0.7 %)	41
dbo:Organisation	790,974 (12.5 %)	7	241,286 (5.8 %)	12
dbo:PopulatedPlace	616,022 (9.8 %)	8	478,351 (11.4 %)	5
dbo:ArchitecturalStructure	492,580 (7.8 %)	9	150,254 (3.6 %)	16
dbo:Settlement	462,082 (7.3 %)	10	449,479 (10.7 %)	6
dbo:Building	454,448 (7.2 %)	11	68,582 (1.6 %)	25
dbo:Company	417,010 (6.6 %)	12	58,400 (1.4 %)	27
dbo:Species	211,536 (3.4 %)	13	252,166 (6.0 %)	10
dbo:Eukaryote	207,771 (3.3 %)	14	247,208 (5.9 %)	11
dbo:Food	178,689 (2.8 %)	15	6,337 (0.2 %)	105

7 Related Work

In this section, we review and summarize works that are most relevant to our research. These include works in named entity recognition, fine-grained entity mention recognition and fine-grained entity typing.

Named Entity Recognition. Named Entity Recognition (NER) is a widely used approach for typing problem. However, most of them only support a small set of coarse-grained types, such as Person, Location, and Organization [17]. However, our task focuses more on typing fine-grained types for entities.

Fine-Grained Entity Mention Recognition. One type of approaches for fine-grained entity mention recognition is based on Named Entity Linking (NEL). They first use NEL tools to link mentions to entities which exist in a knowledge base. Then, their types are obtained from the corresponding entity types. However, their performance mainly relies on the linking entities which exist in the knowledge base used [4]. In our setting, it is reported that there are only about 10 % (about 0.4 million) Chinese entities that have corresponding English entities [23]. Hence, it does not work to use this method alone. While in our work, we just use it to construct our training data.

Other approaches for fine-grained entity mention recognition use some handcrafted features (such as part-of-speech tags, dependency parsing results) and external resources (such as WordNet) to build a classifier to type mentions [9, 11, 24]. However, most of their features are extracted from sentences, while our work focuses on using entity-level features available in knowledge bases.

Fine-Grained Entity Typing. The state-of-the-art method for fine-grained entity typing is SDType [13]. However, it does not consider the hierarchical structure between types, and the weight value of features are fixed for all types, which is not suitable for multi-label classification problem as we empirically showed.

8 Conclusion

In this paper, we introduce `CUTE` for typing Chinese entities with DBpedia types. By using the cross-lingual entity linking method to construct the training data, we propose a multi-label hierarchical classification method. Extensive experiments have verified the efficiency and effectiveness of `CUTE`.

References

1. Palmero Aprosio, A., Giuliano, C., Lavelli, A.: Automatic expansion of DBpedia exploiting wikipedia cross-language information. In: Cimiano, P., Corcho, O., Presutti, V., Hollink, L., Rudolph, S. (eds.) ESWC 2013. LNCS, vol. 7882, pp. 397–411. Springer, Heidelberg (2013)

2. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.G.: DBpedia: a nucleus for a web of open data. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 722–735. Springer, Heidelberg (2007)
3. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp. 1247–1250. ACM (2008)
4. Dong, L., Wei, F., Sun, H., Zhou, M., Xu, K.: A hybrid neural model for type classification of entity mentions. In: Proceedings of the 24th International Conference on Artificial Intelligence, pp. 1243–1249. AAAI Press (2015)
5. Gangemi, A., Nuzzolese, A.G., Presutti, V., Draicchio, F., Musetti, A., Ciancarini, P.: Automatic typing of DBpedia entities. In: Cudré-Mauroux, P., Heflin, J., Sirin, E., Tudorache, T., Euzenat, J., Hauswirth, M., Parreira, J.X., Hendler, J., Schreiber, G., Bernstein, A., Blomqvist, E. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 65–81. Springer, Heidelberg (2012)
6. Hoffart, J., Suchanek, F.M., Berberich, K., Weikum, G.: Yago2: a spatially and temporally enhanced knowledge base from wikipedia. Artif. Intell. **194**, 28–61 (2013)
7. Lee, T., Wang, Z., Wang, H., Hwang, S.W.: Attribute extraction and scoring: a probabilistic approach. In: 2013 IEEE 29th International Conference on Data Engineering (ICDE), pp. 194–205. IEEE (2013)
8. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., et al.: DBpedia-a large-scale, multilingual knowledge base extracted from wikipedia. Semant. Web J. **5**, 1–29 (2014)
9. Ling, X., Weld, D.S.: Fine-grained entity recognition. In: AAAI. Citeseer (2012)
10. Murdock, J.W., Kalyanpur, A., Welty, C., Fan, J., Ferrucci, D.A., Gondek, D., Zhang, L., Kanayama, H.: Typing candidate answers using type coercion. IBM J. Res. Dev. **56**(3.4), 7:1–7:13 (2012)
11. Nakashole, N., Tylenda, T., Weikum, G.: Fine-grained semantic typing of emerging entities. In: ACL (1), pp. 1488–1497 (2013)
12. Passant, A.: dbrec — music recommendations using DBpedia. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part II. LNCS, vol. 6497, pp. 209–224. Springer, Heidelberg (2010)
13. Paulheim, H., Bizer, C.: Type inference on noisy RDF data. In: Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N., Welty, C., Janowicz, K. (eds.) ISWC 2013, Part I. LNCS, vol. 8218, pp. 510–525. Springer, Heidelberg (2013)
14. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al.: Scikit-learn: machine learning in python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
15. Pohl, A.: Classifying the wikipedia articles into the opencyc taxonomy. In: Proceedings of the Web of Linked Entities Workshop in Conjuction with the 11th International Semantic Web Conference, vol. 5, p. 16 (2012)
16. Ponzetto, S.P., Strube, M.: Deriving a large scale taxonomy from wikipedia. In: AAAI, vol. 7, pp. 1440–1445 (2007)
17. Ritter, A., Clark, S., Etzioni, O., et al.: Named entity recognition in tweets: an experimental study. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 1524–1534 (2011)

18. Silla Jr., C.N., Freitas, A.A.: A survey of hierarchical classification across different application domains. Data Min. Knowl. Discov. **22**(1–2), 31–72 (2011)
19. Srinivas, S.: A generalization of the noisy-or model. In: Proceedings of the Ninth International Conference on Uncertainty in Artificial Intelligence, pp. 208–215. Morgan Kaufmann Publishers Inc. (1993)
20. Suchanek, F.M., Abiteboul, S., Senellart, P.: Paris: probabilistic alignment of relations, instances, and schema. Proc. VLDB endowment **5**(3), 157–168 (2011)
21. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: Proceedings of the 16th International Conference on World Wide Web, pp. 697–706. ACM (2007)
22. Wang, Z., Li, J., Tang, J.: Boosting cross-lingual knowledge linking via concept annotation. In: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, pp. 2733–2739. AAAI Press (2013)
23. Wang, Z., Li, J., Wang, Z., Tang, J.: Cross-lingual knowledge linking across wiki knowledge bases. In: Proceedings of the 21st International Conference on World Wide Web, pp. 459–468. ACM (2012)
24. Yosef, M.A., Bauer, S., Hoffart, J., Spaniol, M., Weikum, G.: Hyena: hierarchical type classification for entity names (2012)