DBpedia Entity Type Inference Using Categories

Lu Fang, Qingliang Miao, Yao Meng

Fujitsu R&D Center Co., Ltd. Beijing 100025, China.

{fanglu, qingliang.miao, mengyao}@cn.fujitsu.com

Abstract. In this paper, we investigate how to identify entity type based on entity category information. In particular, we first calculate the statistical distribution of each category over all the types. And then we generate type candidates according to distribution probability. Finally we identify the correct type according to distribution probability, keywords in category and abstract. To evaluate the effectiveness of the approach, we conduct preliminary experiments on a real-world dataset from DBpedia. Experimental results indicate that our approach is effective in identifying entity types.

Keywords: DBpedia, Type Inference, Category

1 Introduction

DBpedia is a large-scale, multilingual knowledge base extracted from Wikipedia. In DBpedia, some entities have type information, e.g. "?s *rdf:type dbo:Organisation*[1]", but most of them are without type information. For example, only 21.9% entities in the Japanese DBpedia[2] have type information. Type information is very important in knowledge bases. Knowing what type a certain instance is, e.g., a person, organisation, place, etc., is key for some applications such as question answering, entity search, entity linking, etc. The main approaches for entity type inference are: content-based and linked-based methods. Gangemi A. et al. [1] use a natural language deep parser to produce definition sentences from Wikipedia pages, and then select types and type-relation from the RDF graph based on graph patterns. Similarly with [1], Kliegr T. and Kliegr O. [2] discover types from the first sentence of Wikipedia articles using Hearst pattern matching over part-of-speech annotated text. Paulheim H. and Bizer C. [3] utilize statistical distribution of types in the subject and object position of the property for entity's type prediction. Fossati M. et al. [4] convert the chaotic Wikipedia category system into an extensive general-purpose taxonomy through four-step processing pipeline.

Inspired by [3,4], we propose an approach to infer entity type according to entity's category information instead of using all the properties. Categories are chosen because they are predictive for entity type. For example, given a statement "*?s dcterms:subject*[3] *dbr-category:People_from_Tokyo*[4]", we may infer that *?s* is a person. Moreover, more than 54% entities of untyped entities have category information.

[1] dbo: standing for http://dbpedia.org/ontology

[2] All DBpedia examples in this paper use version 2015-04.

[3] http://purl.org/dc/terms/subject

[4] http://dbpedia.org/resource/Category:People_from_Tokyo

2 Approach

In DBpedia, a resource may have category information via the "*dcterms:subject*" property. We assume that an entity should belong to certain type if it has a certain category value. However, not all the type can be inferred from category literally. For example, given entity with category "*Place_of_birth_missing*", people may think its type is "*dbo:Place*". However, 95.6% of 5416 entities with category "*Place_of_birth_missing*" have a type "*dbo:Person*". Consequently, besides the textual content in category, we have to use the statistical distribution of category to identify entity types. Next we will introduce how to calculate the distribution probability.

2.1 Distribution Probability Calculation

For each category in DBpedia, there is a statistical distribution of entities with different types. For a given entity *e* with a certain category *c*, make use of the following statistic to measure how likely is it that entity *e* has a type *t*: P(*t*|*c*), that is, the conditional probability of finding an entity of type *t* having *c* as category. Equation 1 formally defines the probability:

$$\mathrm{P(t|c)} = \frac{|\{(s,p,c) \in DKB \wedge (s,a,t) \in DKB\}|}{|\{(s,p,c) \in DKB \wedge (s,a,o) \in DKB\}|} \quad (1)$$

It is the ratio between the number of times the category is used when the entity have a type *t* with its total number of uses except the subjects without a type. *DKB* means the DBpedia knowledge base composed of existing triples (*s, p, o*), (*s, a, t*) denotes that *s* is an entity of type *t*, (*s, a, o*) denotes that *s* has a type, and *p* = *dcterms:subject* .

2.2 Candidate Generation and Selection

In order to determine which entities have type *t*, we first use distribution probability of categories to generate some candidates, assume that *C* is the set of all the categories of a certain entity *e*, $C = \{c_1, c_2, \ldots, c_n\}$. For each $c_i \in C$, calculate P($t|c_i$), if one of the probabilities is larger than a certain threshold, add (*e*, *a*, *t*) as a candidate. To get more candidates, set the threshold as low as possible.

Next, candidates are selected or removed through the candidate selection process, which consists of a series of steps:

Probability Selection Here, we also use the probability P(*t*|*c*), as one can imagine, P(*t*|*c*) is close to 1 indicates that overwhelming majority of entities with category *c* have a type *t*. So for candidate (*e, a, t*), one of the categories of *e* has a probability P(*t*|*c*) larger than a certain threshold, select this candidate, otherwise move to the next step. This threshold is close to 1 and set based on experience.

Entity Name Selection We found that some entities obviously do not belong to the type according to their names. We can filter out some candidates by some simple rules. These rules are simply made that the entity names contain some special words, such as the word *list* for English, 一覧 for Japanese. Remove the candidates that match the rules. And then, we also use a simple string matching algorithm, which requires the category name contains the type name or its plural form in the type name list, such as *people* for *person*, *companies* for *company*. The subclass names are also in the name list. For example, if the category name contains the word *people*, we can keep the candidate (*e*, *a*, *person*). Likewise, if the category name contains *artist*, because *artist* is the subclass of *person*, keep the candidate (*e*, *a, person*) as well. For Japanese category names, translate them into English names before string matching because the type names are defined with English in DBpedia ontology. Some of the Japanese categories have the corresponding English categories linked by the property *owl:sameAs*, and translate the category names which do not have the corresponding English categories by machine translation tool. Remove all the candidates which are not kept in this step. Then move to the next step.

Abstract Selection We can easily get abstracts of entities from DBpedia using property *dbo:abstract*. The first sentence of the abstract is usually the definition of the entity, and definition usually indicates type. We do not need a complex algorithm to extract type from the sentence, just check if the first sentence contains the type name or sub-type name in a candidate, if the answer is true, select this candidate, or go to the next step.

Category List Selection At last, for a candidate (e, a, t), assume C is the set of all the categories of e, $C = \{c_1, c_2, \ldots, c_n\}$, and list L is the set for categories of all the entities with a type t, check the ratio between the number of c_i in list L with the number of C set. For example, there are four categories of entity e, and two of them are in the list L, so the ratio is 0.5. If the ratio is larger than 0.7, candidate (e, a, t) is selected, or it is removed.

3 Evaluation

In order to evaluate the effectiveness of the proposed approach, we conduct our experiments by using test data[5] from Japanese DBpedia. The data is randomly selected from Japanese DBpedia, and it includes three classes Person, Organization and Place. Each class contains 500 entities. Our gold-standard dataset is created from human annotations. In particular, three annotators with a background in information systems are recruited to annotate the test data. To construct the final gold standard, we adopted the following procedure. For entities that have received the same type from all three annotators, we assigned this agreed-upon type. For a small number of entities that have received differing assessment from these three annotators, we had all three annotators go through these entities and discuss their assessment with each other in a face-to-face meeting. We then used their consensual assessment as the final type. We use all the typed entity in DBpedia to calculate the category distribution probability over types. We use method in [3] as baseline and precision, recall and F-Measure are used to evaluate the results.

Table 1 shows the experiment results of entity type identification. From table 1, we can see our approach and baseline obtain relative high in precision. However, baseline method performs less effective in recall. Through careful analysis, we find low recall are caused by property sparsity. Only 11% un-typed entities have particular properties beside general purpose properties. While our method could capture more useful information from category information.

Table 1. Results for typing un-typed entities

Dataset	Method	Precision	Recall	F1-measure
Person	Ours	0.962	0.962	0.962
	Baseline	0.812	0.806	0.809
Organization	Ours	0.948	0.996	0.971
	Baseline	0.942	0.552	0.696
Place	Ours	0.930	0.998	0.963
	Baseline	0.945	0.516	0.668

5 The test data is available at http://36.110.45.46:8090/API/test_data.txt

To evaluate the effect of threshold in the probability selection step, we conduct experiment with different threshold. Table 2 shows the experiments. From table 2, we can see precision increases with the threshold increases, while recall drops a little.

Table 2. Results for different threshold

Dataset	Threshold	Precision	Recall	F1-measure
Person	0.90	0.962	0.962	0.962
	0.95	0.963	0.962	0.962
	0.98	0.965	0.962	0.963
Organization	0.90	0.948	0.996	0.971
	0.95	0.957	0.996	0.976
	0.98	0.959	0.996	0.977
Place	0.90	0.930	0.998	0.963
	0.95	0.938	0.996	0.966
	0.98	0.945	0.966	0.955

4 Conclusions

In this paper, we study how to infer entity type based on entity category information from DBpedia. Preliminary experiment results indicate the method is promising. The main difference between existing methods is that our approach focuses on category which contains rich type information. Moreover, since our method only process the category information, it is more efficient for large scale dataset such as DBpedia. Our method is language independent and we have identified type for 234 thousand and 471 thousand entities in Japanese and English DBpedia respectively. In the future work, we can also identify other entity types and fine-grained entity types using this method, namely, all the class and subclass in DBpedia ontology. For example, class "Person" and its sub class "Artist", "Actor" and many more. And we are going to identify entity type for entities without category information. Finding more predictive properties for entity type identification is another direction. We also plan to use machine learning method to solve this issue.

5 References

1. Gangemi, A., Nuzzolese A. G., Presutti, V., Draicchio, F., Musetti, A., Ciancarini, P.: Automatic typing of DBpedia entities. In: Proceedings of the International Semantic Web Conference (ISWC 2012), 65-81(2012).
2. Kliegr, T., Zamaza, O.: Towards linked hypernyms dataset 2.0: Complementing DBpedia with hypernym discovery and statistical type inference. In: Proceedings of the Conference on Language Resources and Evaluation, 3517-3523(2014).
3. Paulheim, H., Bizer, C.: Type inference on noisy RDF data. In: Proceedings of the International Semantic Web Conference (ISWC 2013), 510-525(2013).
4. Fossati, M., Kontokostas, D., Lehmann, J.: Unsupervised learning of an extensive and usable taxonomy for DBpedia. In: The International Conference, 177-184(2015).