Detect Missing Attributes for Entities in Knowledge Bases via Hierarchical Clustering*

Bingfeng Luo[1,**], Huanquan Lu[1,**],
Yigang Diao[2], Yansong Feng[3], and Dongyan Zhao[3]

[1] Department of Machine Intelligence, Peking University, Peking, China
luoluo123n@pku.edu.cn, huanquanlu@g.ucla.edu
[2] Technic and Commnication Technology Bureau, Xinhua News Agency, China
thomasfred@xinhua.org
[3] ICST, Peking University, Peking, China
{fengyansong,zhaody}@pku.edu.cn

Abstract. Automatically constructed knowledge bases often suffer from quality issues such as the lack of attributes for existing entities. Manually finding and filling missing attributes is time consuming and expensive since the volume of knowledge base is growing in an unforeseen speed. We, therefore, propose an automatic approach to suggest missing attributes for entities via hierarchical clustering based on the intuition that similar entities may share a similar group of attributes. We evaluate our method on a randomly sampled set of 20,000 entities from DBPedia. The experimental results show that our method can achieve a high precision and outperform existing methods.

Keywords: Missing Attributes, Automatic Knowledge Base Construction, Hierarchical Clustering.

1 Introduction

The proliferation of knowledge-sharing communities such as Wikipedia and the progress in scalable information extraction and machine learning techniques have enabled the automatic construction of knowledge bases (KBs) such as DBPedia, Freebase and YAGO[3]. However, these automatically constructed KBs often suffer from quality issues such as the lack of attributes, duplication of entities, incorrect classifications and et al. In this paper, we focus on the problem of lack of attributes for existing KB entities. Since the contents of KBs are often dynamic and their volume grows rapidly, manually detecting missing attributes is a labor-intensive and time consuming task. Therefore, it would be of great importance to automatically suggest possible missing attributes for entities in KBs.

* This work was supported by the National High Technology R&D Program of China (Grant No. 2012AA011101, 2014AA015102), National Natural Science Foundation of China (Grant No. 61272344, 61202233, 61370055) and the joint project with IBM Research. Corresponding author: Yansong Feng.

** Contributed equally to this work.

C. Zong et al. (Eds.): NLPCC 2014, CCIS 496, pp. 392–402, 2014.

Once the missing attributes are found, the automatic KB construction systems can use these information as important hints to support the work of automatic information extraction. In other words, they can find the corresponding attribute values from the web or via other approaches for entities in the KB.

In this paper, we propose an effective way to find missing attributes for entities in the KB, and the experimental results show that our method can achieve a relatively high precision. We will show some related work in the following section and the description of our method will be demonstrated in the third section. After that, we will show the performance of our method with an experiment on DBPedia. Finally, in the last part, the conclusion of the paper will be made and some possible future work will be discussed.

2 Related Work

We define our task as follows: Given a KB and an entity described by a set of attribute-value pairs, the task is to detect appropriate missing attributes for this entity, which are currently not in its attribute set.

Most related work focuses on finding missing attribute values through heuristic ways[5] or just focuses on finding attribute-value pairs in the information extraction way[6]. Although the attribute plays an important part in both of these methods, there is very limited work that focuses directly on finding missing attributes. Recent work that focuses on this problem tries to suggest missing attribute candidates using association rule mining[1]. The basic idea is that if an entity has attribute A, B and etc, then it may imply that this entity should also have attribute C. For example, if an entity (a person in this case) has attribute *music genre* and attribute *instrument*, it may imply that he or she should also have the attribute *record label*. This can be captured by the association rule $musicgenre, instrument \rightarrow recordlabel$, which means that that if we have learned the association rule , then it is reasonable to assume that entities which have attribute *music genre* and attribute *instrument* should also attribute *record label*. Using the method described above, Abedjan and Naumann(2013)[1] obtain top 5 or 10 candidates for each entity. Since the top 5 or 10 candidates usually contains only a relatively low ratio of correct predictions, human examination is required after the candidates are generated.

In essence, the association rules make sense mainly because the left part of the association rule implies that the entity belongs to some categories, and most entities in these categories tend to have the attribute lies on the right part of the association rule. To be specific, having attribute *music genre* and *instrument* implies that this entity is probably a singer or a bandsman. Therefore, it is reasonable to infer that this entity should also have the attribute *record label*. However, this association rule method can only lead to a limited number of categories since it doesn't take advantage of attribute values. For instance, the attribute *occupation* says nothing but this entity is a person in this case. However, the values of attribute *occupation* can tell us if this person is a musician, a politician or something else. Another drawback of this method is that it

implicitly maps attribute set to categories, which will introduce some unnecessary errors to the predicting system. Therefore, we introduce a clustering-based method to overcome these drawbacks. In this way, we focus directly on the categories themselves, and we can base our results on a large number of categories.

3 Our Approach

Rather than inducing missing attributes via existing attributes as described in [1], we turn to use categories that an entity belongs to. Our solution is based on the intuition that if an attribute is owned by the majority of entities in a category, i.e., a group of similar entities, then this attribute should probably be owned by other entities in this category. For example, as shown in figure 1, we can see that *Taylor Swift* (a famous American singer) belongs to category *pop singer*, and almost all the entities except for *Taylor Swift* in category *pop singer* have the attribute *record_label*. Therefore, *Taylor Swift* probably should also have the attribute *record_label*. Notice that all the attributes shown in figure 1 and other examples afterwards come from the attribute system of Wikipedia

In our system, as shown in figure 2, we first convert each entity in the KB into continuous vector representations. Then, we apply a hierarchical clustering method to group entities into clusters according to different attributes, yielding clusters on the bottom of figure 1. Finally, in each cluster that the entity belongs to, we use the method described in figure 2 to detect missing attributes.

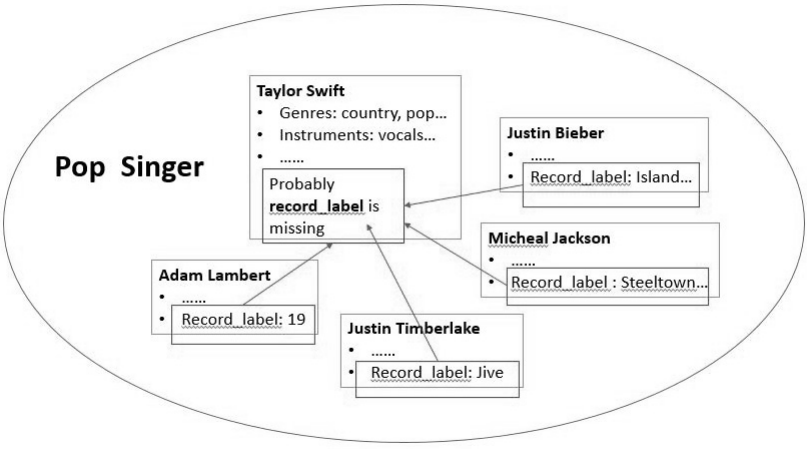

Fig. 1. An example of finding missing attributes for Taylor Swift

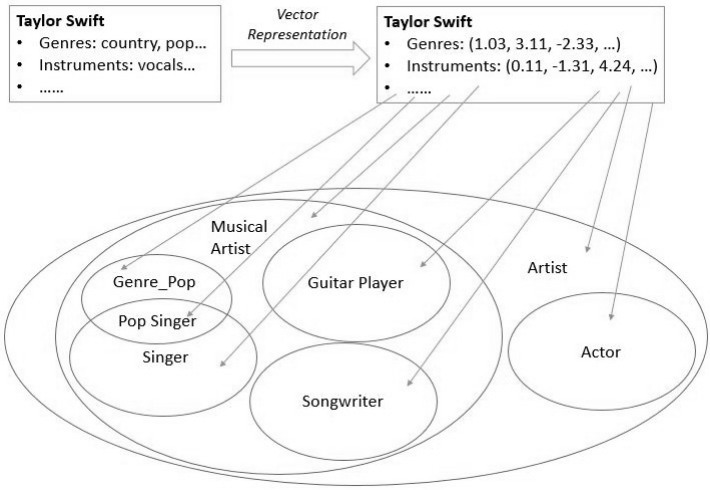

Fig. 2. Overview of our method

3.1 Preprocessing

Here we only make use of the attributes whose values are word strings when building cluster system. We learn a continuous vector representation (word embedding) for these values via RNNLM[2]. For example, the word string *actor* can be represented by a 200-dimension vector using RNNLM. After that, each entity is represented by a set of vectors, and each vector corresponds to a certain attribute of the entity. Notice that we simply abandon attributes with time and numerical values for the reason that they hardly yield useful clusters in this task. Actually, these values can be easily fitted into vector representation by using the first element to store the numeric value and setting other elements to be zero. As for time values, we can use the first element of a vector to represent year, the second to represent month, the third to represent day and the rest elements will be set to zero.

3.2 Entity Clustering

Why not Human-Made Category System: Since most existing KBs have their own category systems, using these existing category systems seems to be a plausible idea. However, these human made category systems have some disadvantages that make them unfit for our task.

First, the categories in these systems are not evenly distributed over all entities. These human made category systems tend to be very good in hot topics, but as for those less popular ones, these systems usually have very limited category information which is unlikely to grow in a short period of time. For example, the entity *Sinsharishkun*, an Assyrian King, has only 3 categories in Wikipedia

while the entity *Alexander the Great* has 16 categories. However, unfortunately, entities in these less popular topics are exactly the ones that are in strong need of missing attribute completion.

Second, the human-made category systems are not well-organized. Take Wikipedia again for example. The entity *Carson Henley*, an American, is assigned to the category *United States*. However, the category *United States* belongs to category *G20 nations*, which makes *Carson Henley* a nation. Actually, this kind of inconsistency can be seen everywhere in the Wikipedia category system, since many categories belong to a higher-level category that may lead to inconsistency like category *United States*.

Third, it is also hard to add new entities into these category systems automatically, which means that it is difficult to use these category systems in different KBs and for newly generated entities. On the one hand, the criterions that people used to make these categories are hard to be understood by machine. We can only use some contextual information to approximate the criterions, which is as difficult as building a new automatically-constructed category system since they all require transforming the contextual information into the way that can be understood by machine. On the other hand, people sometimes use the information that doesn't exist in both the attributes or the attribute values of the entity to create categories. Therefore, it is almost impossible to capture the criterions that people used when making these categories.

Considering all the disadvantages of human-made category systems, it is better to automatically build a category system on our own which is well organized, easy to update and possibly more expressive than human-made ones (we will discuss this later).

Clustering Algorithm: We use an overlapped hierarchical density-based clustering method (similar to [4]) to form our category system (since we are using clustering method, we use the term cluster system instead of category system afterwards). The method is an improved version of the traditional density-based clustering that it allows overlap as well as a hierarchical structure.

Algorithm 1. Overlapped Hierarchical Density-Based Clustering Algorithm.

Input:

The set of entities, E_n;
A group of thresholds, T_k;

Output:

Grouping entities with the same attribute value into the same cluster, yielding first-layer clusters, $C_{1.m_1}$;

1: Traditional density-based clustering on $C_{1.m_1}$, yielding clusters of the second layer, $C_{2.m_2}$;
2: Assigning isolated points and non-core points to their neighbor clusters if close enough;
3: Using mean vector as representative vector for $C_{2.m_2}$;
4: Repeat step 1-3 to get higher layer clusters until no new cluster is generated;

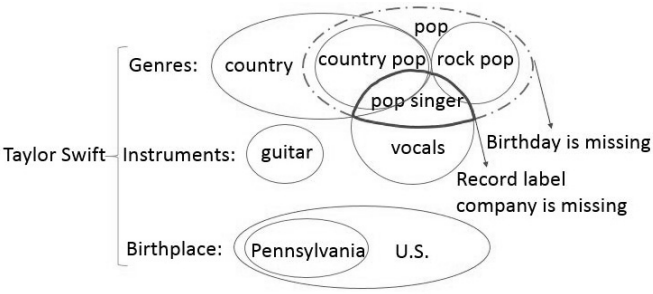

Fig. 3. An example for Taylor Swift's attributes

Instead of considering all the attribute vectors when clustering entities, we deal with each individual attribute respectively. As shown in the algorithm above, for each attribute we group entities with the same attribute value into the same cluster (one entity may belong to several clusters), which forms the first layer of our hierarchical cluster system. Then, to construct the second layer, we first use traditional density-based method to get clusters without overlap. After that, we check if the isolated points as well as the non-core points are close enough to any clusters, and assign these points to their neighbor clusters which are close enough (one point can be assigned to several neighbor clusters). Now, the points that lies in the overlap areas are found. After that, we continue performing this overlapped density-based clustering method over these newly found clusters using their mean vectors as their representative vectors to get a higher layer. Repeat the former step until the algorithm doesn't generate new clusters, then we get a hierarchy of clusters (***simple clusters***, to distinguish against *complex clusters* described afterwards) for each attribute, where each cluster contains entities that are similar to some extent. For example, figure 3 shows three attributes for *Taylor Swift*. According to her *genres*, she has been assigned to four clusters: *country*, *country pop*, *rock pop*, and *pop*. Note that *country pop* belongs to *country*, both *country pop* and *rock pop* belong to a higher-level cluster, *pop* and she is assigned to both *country pop* and *rock pop* for the reason that we introduced overlap into our algorithm.

Furthermore, notice that the intersection of different *simple clusters* is also meaningful. For example, the *pop singer* cluster is the intersection of *simple cluster pop* and *simple cluster vocals* in figure 3. Therefore, for each entity, we need to intersect combinations of its *simple clusters* to get ***complex clusters*** it may belong to. This intersecting step gives the cluster system great ability of expression that it can capture most of the categories in human-made category systems as well as those categories that people missed.

Obviously, it is inefficient to enumerate all combinations. We therefore apply a pruning strategy to improve the efficiency. The idea is that we can abandon small clusters when intersecting clusters. Because small clusters are not strong

enough to support our further induction. Empirically, intersecting five or more *simple clusters* rarely yields useful clusters, therefore, setting four as the maximum number of *simple clusters* to intersect is reasonable and is an effective way to speed up.

Good Properties of the Cluster System. First, our cluster system is well organized, which means there won't be any inconsistency described in the beginning of this section. This is guaranteed because we only cluster in the domain of one particular attribute once at a time, which makes us focus on just one aspect of the entities. On the contrary, human-made categories may focus on different aspects in different category level. In the example of section 3.2, the category system first focuses on nationality, then it changes to focus on international organization, which results in the inconsistency. In our cluster system, even if we will intersect these clusters afterwards, these new clusters still have clear and consistent focuses (previous focuses are always followed).

Second, our cluster system is possibly more expressive than human-made ones. This is achieved by applying intersection to *simple clusters*. On the one hand, as shown above, many human-made categories can be yielded by our algorithm. On the other hand, categories like *American Female Referee* (*complex clusters*) are only covered by human-made categories in a small proportion due to its huge amount. Therefore, our method can yield much more clusters than human-made ones. Furthermore, our method can easily yield clusters for less popular entities, which only have very limited number of human-made categories.

3.3 Detecting Missing Attributes

We define the attributes owned by the majority of entities of a cluster as ***common attributes***. Once we have found appropriate cluster assignments for each entity, based on the intuition, we first compute the support of each attribute in a cluster, as follows:

$$support(a_i) = \frac{number\ of\ entities\ that\ have\ a_i}{number\ of\ entities}$$

If the support of attribute a_i is strong enough, then we consider a_i as a *common attribute* of this cluster, and a_i should be owned by all the members of the cluster. For example, since most pop singers have the attribute of *record label*, we can reasonably infer that all entities in cluster *pop singer* should have this attribute.

As for entities already existing in the KB, we have already found clusters they belong to during the clustering step. Then, for each entity, if a *common attribute* of the clusters that the entity belongs to does not appear in the entity's attribute set, we should suggest this attribute as its missing attribute.

For those newly extracted entities which have fewer attributes, we first find the *simple clusters* they may belong to by simply calculating the vector similarities, and find *complex clusters* by intersecting *simple clusters*. Notice that once it is

assigned to a lower-layer *simple cluster*, it should also be assigned to higher-layer *simple clusters* which encompass the lower-layer one. Once we have found all these clusters, we can take similar steps as those of existing entities to detect their missing attributes.

4 Experiments and Discussions

We examine our model on the *Person* category of DBPedia which contains randomly sampled 20,000 entities. We cluster half of entities in the data set, regarded as entities existing in the KB and the other half as new entities. We regard points that has more than 3 neighbor points with a similarity higher than 0.6 to be core points. When doing overlap, we lower the similarity threshold to 0.55. After that, we use the same parameter to get higher layer clusters. In the finding-missing-attribute part, we abandon clusters whose size is smaller than 5 and consider attributes with a support higher than 0.6 to be *common attributes*. When fitting new entities into the cluster system, we only accept the match with a similarity higher than 0.85.

To evaluate the performance of our models, we use the same evaluation strategy as [1]. We randomly drop one attribute from each entity in the data set, then we rank the suggested missing attribute candidates for each entity according to the support of these attributes within its clusters. If the top k candidates contain the dropped attribute, we record this instance as correct. We evaluate our method in different ks: top1, top5 and top10. Although the data set of [1] and ours are not identical, they are both randomly sampled from the *Person* category of DBPedia with similar volumes. We hope the imperfect comparisons could still draw the skeleton of differences between our method and the method of [1]. As Table 1 demonstrates, our method can achieve a precision of 84.43% by considering the first candidate, 95.36% for top 5 candidates and 96.05% for top 10 candidates for existing entities, much higher than [1] whose precision computed over the *Person* category is 51% for top 5 candidates and 71.4% for top 10 candidates([1] didn't compute precision for the first candidate). We can see that our method performs much better than [1], which makes sense because we focus directly on the categories themselves (rather than implicitly contained in the association rule), and we can base our results on a large number of categories.

However, for new entities, our method can only achieve a precision of 33.86% for top 1 attributes, 45.45% for top 5 and 46.07% for top 10, all of which are not reported by [1]. The experimental results show that the performance on

Table 1. The performance of our proposed model

Precision	Top 1	Top 5	Top 10	Human Evaluation
Association Rules	Not Available	51%	71.4%	Not Available
Existing Entities	84.43%	95.36%	96.05%	96.72%
New Entities	33.86%	45.45%	46.07%	97.09%

Table 2. Some missing attributes suggested by our model

Entity	the Most Important DBPedian Category	Missing Attributes
Cho Beom-Hyeon	Baseball Player	team_label, throwingSide, activeYearStartDate, deathDate, statisticValue
Joseph Kariyil	Christian Bishop	birthPlace
Gerry Joly	Singer-Songwriter	recordLabel_label, occupation_label, activeYearsStartYear, hometown_label
William Thum	Office Holder	successor_label, almaMater_label, residence_label
Bob Turner	Congressman	termPeriod_label, nationality_label, occupation_label, battle_label, allegiance
Andrew H. Ward	Congressman	nationality_label, residence_label, termPeriod_label, occupation_label, otherParty_label
Li Haiqiang	Soccer Player	height, weight, shoots, number, league_label
Julian L. Lapides	Politician	almaMater_label, termPeriod_label
Tomo Yasuda	Musician	genre_label, occupation_label, recordLabel_label
Howard L. Vickery	Military Person	allegiance, occupation_label, militaryBranch_label, award_label

new entities decreases significantly. One of the reasons may be that the dropped attributes of new entities may never appear in the training data.

We should notice that, the evaluation criterion does not make full sense in the case when most of the attributes in top k are predicted correctly except for the imperfection that these k attributes don't contain the one we dropped before. And this situation occurs possibly because that the dropped attributes of

new entities may never appear in the training data. Therefore, we also perform human evaluation on the detected missing attributes.

In the human evaluation, we first randomly sample 2,000 entities, 1,000 from existing entities and 1,000 from the new ones. Then, we manually judge whether each suggested missing attribute is reasonable or not. As shown in table 2, we randomly selected 10 entities (the first 5 are existing entities and the last 5 are new entities) and chose the top 5 candidates for exhibition (some entities may have less than 5 candidates). We can see that most missing attributes are reasonable except for 3 imperfections. One is that we suggest *Bob Turner* should have the attribute *Battle*. This suggestion appears because he once served in the army for about 4 years. However, this service doesn't necessarily imply that he once attended a battle. Another one occurs in the results of *Andrew H. Ward*. The results contain the attribute *Other Party*, which apparently doesn't make sense here. The final one is associated with *Howard L. Vickery*, who has the missing attribute suggestion *Award*. This doesn't make full sense because not all military person receive an award. However, the result is right at this time.

In general, the accuracy for existing entities is 96.72% and the accuracy for new entities is 97.09%, which is high enough to support automatic knowledge base construction. Therefore, the first evaluation method actually underestimated the performance of our method. Apart from that, this human examination result to some extent supports our explanation for the relatively bad performance on new entities under the first evaluation criterion: the dropped attributes of new entities may never appear in the training data.

5 Conclusion and Future Work

In this paper, we propose to automatically suggest missing attributes for KB entities by examing *common attributes* owned by the majority of entities in their clusters obtained through a hierarchical clustering algorithm. The experimental results show that our model outperforms existing method by a large margin and human evaluation indicates the potentials of our model in practice.

However, we admit that the more challenging scenario for our model is to detect missing attributes in multi-language knowledge bases. Our next step is to adapt our ideas to Chinese data sets and investigate the possibility of applying to multi-language knowledge bases.

When missing attributes are found, we still need to fill the missing attribute values. Therefore, we will also try to combine our system with information extraction methods to find the missing attribute-value pairs for entities in the future.

Furthermore, our method actually forms the basis of solutions for other quality applications in the automatically constructed knowledge bases since the cluster system we built actually has more power than just detecting missing attributes. For instance, with our cluster system we can deal with errors in entity type classifications.

References

1. Abedjan, Z., Naumann, F.: Improving rdf data through association rule mining. Datenbank-Spektrum 13(2), 111–120 (2013)
2. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv: 1301.3781 (2013)
3. Suchanek, F., Weikum, G.: Knowledge harvesting in the big-data era. In: Proceedings of the 2013 International Conference on Management of Data, pp. 933–938 (2013)
4. Xu, X., Yuruk, N., Feng, Z., Schweiger, T.A.: Scan: A structural clustering algorithm for networks. In: Proceedings of the 13th ACM SIGKDD, pp. 824–833 (2007)
5. Grzymala-Bussem, J.W., Grzymala-Busse, W.J.: Handling Missing Attribute Values. In: Data Mining and Knowledge Discovery Handbook, pp. 33–51 (2010)
6. Wong, Y.W., Widdows, D., Lokovic, T., Nigam, K.: Scalable Attribute-Value Extraction from Semi-Structured Text. IEEE International Conference on Data Mining Workshops (2009)