RelClus: Clustering-based Relationship Search

Yanan Zhang, Gong Cheng, and Yuzhong Qu

State Key Laboratory for Novel Software Technology, Nanjing University,
Nanjing 210023, P.R. China
ynzhang@smail.nju.edu.cn, {gcheng, yzqu}@nju.edu.cn

Abstract. Searching and browsing relationships between entities is an important task in many domains. To support users in interactively exploring a large set of relationships, we present a novel relationship search engine called RelClus, which automatically groups search results into a dynamically generated hierarchy with meaningful labels. This hierarchical clustering of relationships exploits their schematic patterns and a similarity measure based on information theory.

Keywords: Association discovery, exploratory browsing, hierarchical clustering, path finding, relationship search.

1 Introduction

Many information needs in various domains can be met by using an information system that supports searching and browsing relationships (a.k.a. associations) between entities, which are represented as paths in RDF graph. Whereas path finding has been efficiently implemented (e.g. [4]), a major challenge that remains is how to organize the results, which could be a large set of relationships and cause information overload. To address this issue, efforts (e.g. [1]) have been made to rank the results according to various criteria. Another line of work such as [3], inspired by recent advances in exploratory search [2], provides faceted categories to organize search results into groups and serve as filters, each of which characterizes a common feature of the underlying relationships such as their length, or a type of a node (i.e. a class) or edge (i.e. a property) involved. Differently, in this demo we will present a relationship search engine called RelClus[1] that practices another implementation of exploratory search, namely clustering. RelClus measures the similarity between relationships based on their schematic patterns by using information theory, and returns a dynamically generated hierarchical clustering with meaningful labels, to effectively guide the exploration of search results. Figure 1 shows a screenshot of the system.

2 Design and Implementation

RelClus is based on the DBpedia data set (`dbpedia.org`), and consists of four components: keyword mapping, path finding, relationship clustering, and result presentation, which will be detailed in the following.

[1] `http://ws.nju.edu.cn/relclus/`.

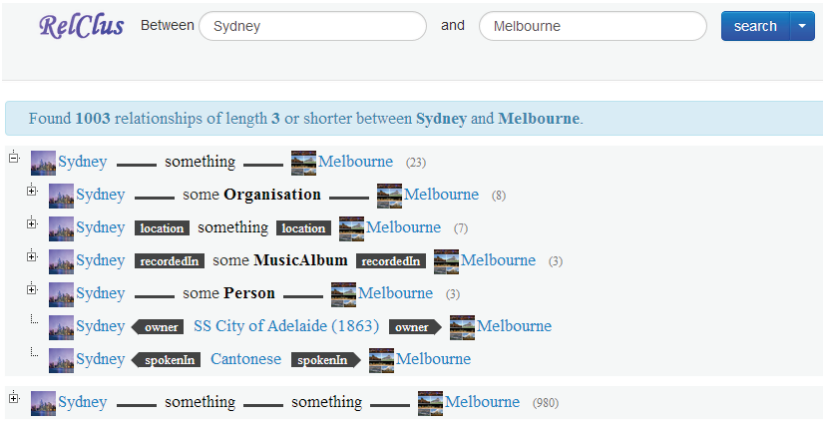

Fig. 1. A screenshot of RelClus.

2.1 Keyword Mapping

User interaction starts with two keyword phrases, e.g. *Sydney* and *Melbourne* in Fig. 1, describing two entities between which the relationships are requested. Featured by the autocomplete functionality implemented based on Apache Lucene (`lucene.apache.org`), RelClus can help the user conveniently determine the mapping from keyword phrases to entities. In addition, when necessary, the user can change the default length limit on the relationships to be returned, by choosing an appropriate value from the drop-down list next to the search button.

2.2 Path Finding

Once the search button is clicked, RelClus will start to find all the paths (subject to a length limit) between the two entities specified by the user. In particular, edges in a relationship are not required to go the same direction because the inverse of a property also has meaning for human readers. Figure 2 illustrates an RDF graph containing five relationships, R_1–R_5, from Sydney to Melbourne, as our running example.

Paths are found by using bidirectional breadth-first search (bi-BFS), which runs two simultaneous searches from the two entities given and finds paths when the two meet in the middle. According to our experimental results, bi-BFS is generally faster than a single BFS or DFS, though requiring more memory than DFS. To further reduce the time needed, our bi-BFS runs concurrently in multiple threads. Besides, a cache is used to avoid repeated path finding for repeated queries in the future.

2.3 Relationship Clustering

As a key step of RelClus, all the relationships found by bi-BFS will be clustered into a hierarchy. For simplicity, the relationships are firstly grouped by length,

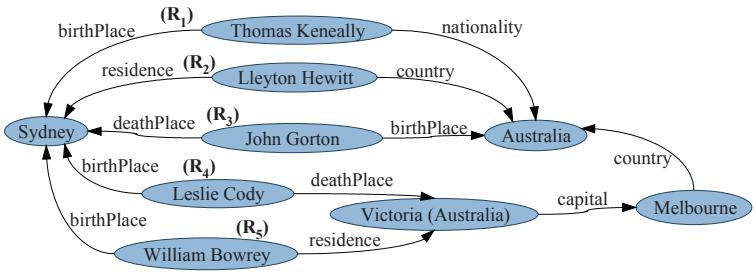

Fig. 2. An RDF graph containing five relationships from `Sydney` to `Melbourne`.

and then each group is processed individually. We will illustrate our clustering algorithm by using R_1–R_5 in Fig. 2, all of which are of length three.

Our clustering algorithm follows an agglomerative manner; that is, each relationship starts in its own cluster, and a hierarchy of clusters is built by progressively merging the most similar pair of clusters.

Before describing the similarity measure, we need to introduce how we assign a meaningful and representative label to each cluster. The label of a cluster is a *relationship pattern*, which is a high-level abstraction of relationships where the nodes can be either entities or classes, and the directions of edges are omitted; the label of a singleton cluster is just the unique relationship it contains. For instance, in Fig. 3, R_4 labels the singleton cluster $\{R_4\}$; P_1 is a relationship pattern that labels the cluster $\{R_4, R_5\}$, where $\top_{\mathbb{P}}$ denotes the top property that is a superproperty of all the properties.

We call P_1 a *superpattern* of R_4 and R_5 in the sense that for each entity, class, and property in R_4 and R_5, the element in the corresponding position in P_1 is either the same or its type, superclass, and superproperty, respectively; in particular, it is the *least common element* among the possibilities. For instance, Leslie Cody in R_4 and William Bowrey in R_5 are instances of both Person and Athlete, and we choose Athlete in P_1 because it is the least common type given Athlete being a subclass of Person, and thus contains the most *information content* as discussed in [5].

We define the similarity between two clusters as *the information content associated with the relationship pattern that labels their union*, which indicates how many commonalities the two clusters share. The information content associated with a relationship pattern is the sum of the information contents of its elements. So, at the beginning of the clustering process in our running example, $\{R_4\}$ and $\{R_5\}$ are merged before $\{R_1\}$ and $\{R_2\}$ because the label of $\{R_4, R_5\}$, which would be P_1, contains more information content than P_2 which would label $\{R_1, R_2\}$, mainly because P_2 contains one more top property, the information content of which is trivially zero.

In addition, after successively forming $\{R_4, R_5\}$ labeled with P_1 and $\{R_1, R_2\}$ labeled with P_2, we will immediately merge $\{R_1, R_2\}$ and $\{R_3\}$ into $\{R_1, R_2, R_3\}$, still labeled with P_2, because R_3 also matches this pattern. Finally, the two

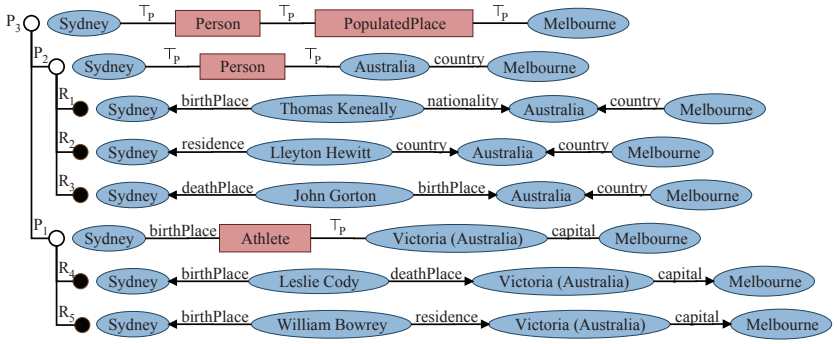

Fig. 3. A hierarchical clustering of five relationships from `Sydney` to `Melbourne`.

clusters labeled with P_1 and P_2 are merged into the root cluster labeled with P_3, and a hierarchy is formed.

2.4 Result Presentation

A hierarchical clustering is visualized as a collapsible/expandable tree, as illustrated in Fig. 1. Each entity is prefixed by a thumbnail if available; each class is prefixed by *some*; the top property is omitted; and the top class is shown as *something*. Each non-singleton cluster is suffixed by its size in parentheses, and sibling clusters are sorted by their sizes decreasingly.

Acknowledgments. This work was supported in part by the NSFC under Grant 61100040 and 61223003, and in part by the JSNSF under Grant BK2012723.

References

1. Aleman-Meza, B., Halaschek-Wiener, C., Arpinar, I.B., Ramakrishnan, C., Sheth, A.P.: Ranking Complex Relationships on the Semantic Web. IEEE Internet Comput. 9(3), 37–44 (2005)
2. Hearst, M.A.: Clustering versus Faceted Categories for Information Exploration. Comm. ACM 49(4), 59–61 (2006)
3. Heim, P., Lohmann. S., Stegemann, T.: Interactive Relationship Discovery via the Semantic Web. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010, Part I. LNCS, vol. 6088, pp. 303–317. Springer, Heidelberg (2010)
4. Janik, M., Kochut, K.: BRAHMS: A WorkBench RDF Store and High Performance Memory System for Semantic Association Discovery. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 431–445. Springer, Heidelberg (2005)
5. Resnik, P.: Using Information Content to Evaluate Semantic Similarity in a Taxonomy. In: 14th International Joint Conference on Artificial Intelligence, Volume 1, pp. 448–453. Morgan Kaufmann, San Francisco (1995)