Knowledge-Based Systems 141 (2018) 200–210

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

A hierarchical co-clustering approach for entity exploration over Linked Data

Liang Zheng, Yuzhong Qu*, Xinqi Qian, Gong Cheng

National Key Laboratory for Novel Software Technology, Nanjing University, China

ARTICLE INFO

Article history:
Received 20 February 2017
Revised 10 November 2017
Accepted 13 November 2017
Available online 14 November 2017

Keywords:
Linked Data
Entity exploration
Hierarchical co-clustering

ABSTRACT

With the increasing amount of Linked Data on the Web, large numbers of linked entities often make it difficult for users to find the entities of interest quickly for further exploration. Clustering as a fundamental approach, has been adopted to organize entities into meaningful groups. In general, link and entity class are semantically labelled and can be used to group linked entities. However, entities are usually associated with many links and classes. To avoid information overload, we propose a novel hierarchical co-clustering approach to simultaneously group links and entity classes. In our approach, we define a measure of intra-link similarity and intra-class similarity respectively, and then incorporate them into co-clustering. Our proposed approach is implemented in a Linked Data browser called CoClus. We compare it with other three browsers by conducting a task-based user study and the experimental results show that our approach provides useful support for entity exploration. We also compare our algorithm with three baseline co-clustering algorithms and the experimental results indicate that it outperforms baselines in terms of the Clustering Index score.

1. Introduction

The volume of Linked Data on the Web has increased rapidly [1]. It can be represented as a collection of triples by using the Resource Description Framework (RDF) [2]. In Fig. 1, there is an RDF description of Steven Spielberg in DBpedia [3]. These entity-centric structured data can be reused to facilitate a wide variety of applications such as Web search and business intelligence. However, with the enrichment of available entity-centric Linked Data, large numbers of linked entities often make it difficult for users to find the entities of interest quickly for further exploration. For instance, from Steven Spielberg in DBpedia, it is possible to navigate toward 117 entities. The user would have to browse a large number of entities and choose the desired ones from these entities. A common technique that has been adopted to organize the large amount of data is clustering [4,5]. Clustering refers to the process of grouping data objects into multiple groups based on different similarity measures such as text-based [6], CoCitation [7] and SimRank [8]. In the case of entity exploration over Linked Data, clustering linked entities can provide a good way to help users navigate and seek the needed entities.

Since link and entity class are two key entity facets, they are generally used to group linked entities. In addition, they can provide semantic labels for the generated clusters. By using different types of links, generic Linked Data browsers (e.g., Tabulator [9]) naturally group linked entities and then present users with a list of links. These links reflect the relationships between current entities and their linked entities. Suppose the user is looking for the works produced by Steven Spielberg. The user can select *producer* link to further explore the works he produced. On the other hand, since entity class provides users an intuitive understanding of entity, it can be used to distinguish and group entities. For instance, Aemoo [10] is an exploratory search application over Linked Data, which groups the entities' neighbours (i.e., their linked entities) based on entity classes and then displays the most relevant entity classes. Suppose the user is looking for the films related to Steven Spielberg. The user can find linked films by an entity class *Film*. Moreover, there are several advanced systems based on faceted browsing (e.g., /facet [11]). They provide different kinds of facets to group entities in multiple dimensions. In fact, facets are raw links and classes. Users can explore a collection of entities by selecting links and classes. For instance, the user can find the films produced by Steven Spielberg through the link *producer* and the class *Film*.

As mentioned above, link and entity class offer effective ways to group entities and thus can be used to help users orient and control entity exploration. However, entities are usually associated with many classes and links. Fig. 2 shows the context of browsing

* Corresponding author.
E-mail address: yzqu@nju.edu.cn (Y. Qu).

https://doi.org/10.1016/j.knosys.2017.11.017
0950-7051/

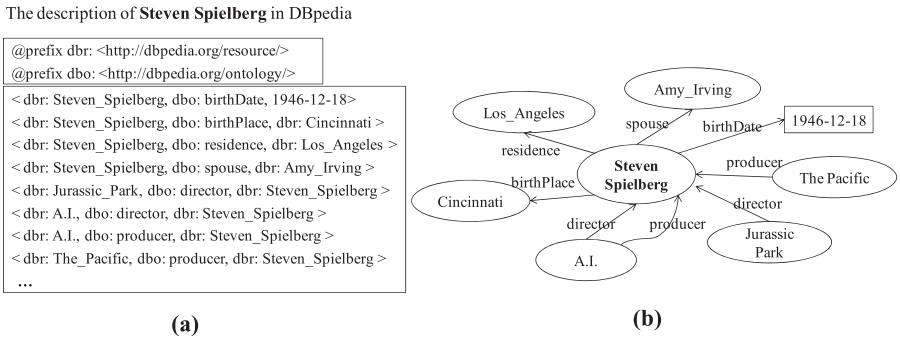

Fig. 1. An excerpt of the description of Steven Spielberg in DBpedia ((a): a set of RDF triples; (b): an RDF graph).

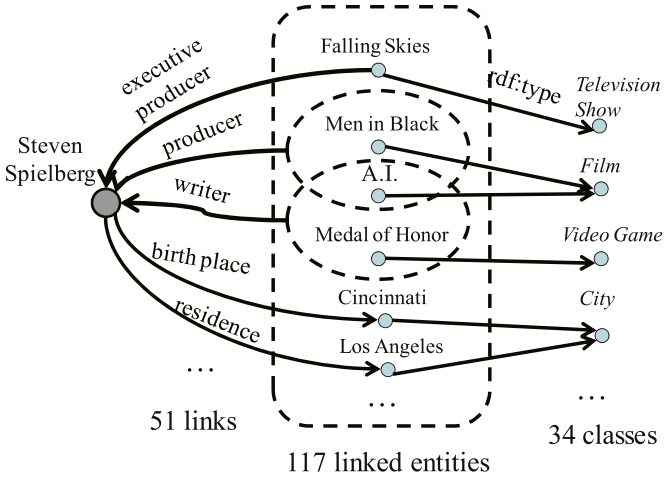

Fig. 2. The context of browsing Steven Spielberg.

Steven Spielberg. There are 117 linked entities, which are associated with 51 links and 34 classes. Users' direct interaction with a large number of links and entity classes could cause the problem of cognitive overhead. To avoid this problem, efforts have been made to rank links and entity classes based on certain criteria (e.g., link frequency and entropy [12]), or organize them as a hierarchy based on their structured features (e.g., SynopsViz [13]). Yet, existing studies take into account only the raw knowledge granularity (link and class) but ignore the intra-similarity among links or classes, as well as the potential relationships between links and classes. As shown in Fig. 2, the link *executive producer* and *producer* are similar based on the lexical similarity between their labels. The class *Television Show* and *Film* are similar based on the semantic similarity (they are subclasses of *Work*). Links and entity classes can be further clustered. In addition, there is a connection between the link *producer* and the class *Film* through an entity A.I. It is often desirable to simultaneously cluster both links and entity classes by exploiting their relationships.

To facilitate entity exploration, we propose a novel hierarchical co-clustering approach to simultaneously group links and entity classes. Our approach is implemented in a prototype system called CoClus.[1] We use the following example to illustrate our approach.

Suppose a user is exploring an entity Steven Spielberg and he wants to find some linked entities of interest for further exploration. The user inputs an entity URI[2] in CoClus (A), as shown in Fig. 3. The system enters a cluster interface. The left-hand side of the interface lists 3 link clusters, such as {*is relative of* ,...}, {*is foundedBy of* ,...} and {*is director of* ,...} (B). The right-hand side lists 3 class clusters, such as {*Place*,...}, {*Organisation*,...} and {*Work*,...} (C). The user knows that Steven Spielberg has three major kinds of linked entities based on links and classes, respectively. Each cluster has its subhierarchy. The user can choose a link cluster or class cluster to further observe its sub-clusters at will. Then, the user selects the {*is director of*,...} link cluster to further observe its sub-clusters. The interface refreshes and shows some sub-clusters such as {*is writer of*,...}, {*is director of*,...} and {*is executiveProducer of*,...} (D). Meanwhile, the interface shows some related class clusters such as {*Work*,...}, {*Software*,...} and {*TelevisionShow*,...} (E). The interface also shows the navigation path ("*is director of*") and supports rollback (F). The user selects a class cluster {*Software*,...} and there are 4 linked softwares (G). By iteratively exploring link and class clusters step-by-step, the user understands the overview of information and finally captures an interesting fact that Steven Spielberg is the *writer* of a *Video Game* Medal of Honor.

In our previous work [14], we began to study the co-clustering approach of entity exploration over Linked Data. The objective of this paper is to continue this line of research by studying more effective approaches, performing a more complete evaluation of their performance, and providing an effective approach to explore the Linked Data. In particular, the contributions of this paper are as follows:

- We detail some existing similarity measures and co-clustering algorithms. Moreover, we introduce a hierarchical co-clustering approach, which leverages the idea of hierarchical clustering and tackles every partition of the node as a process of co-clustering. It attempts to overcome the shortcoming of previous work: it could obtain various local minima when starting from an initial random partition, and need to compute the joint distribution over the whole data set during each iteration.
- We perform an empirical evaluation of the performance of our approach. We implement our approach in a Linked Data exploratory system called CoClus, and provide a detailed description of the system. In order to assess its usability, we compare CoClus with its two restricted versions and one conventional Linked Data system by conducting a task-based user study, and test the statistical significance of the results. We also compare our algorithm with three baseline co-clustering algorithms in terms of the Clustering Index score.

The rest of this manuscript is organized as follows. Section 2 reports related work. Section 3 introduces our proposed approach. Section 4 provides our experimentation. Section 5 concludes this paper.

2. Related work

Since our work leverages hierarchical co-clustering for entity exploration, in this section we will separately review the previous

[1] http://ws.nju.edu.cn/coclus/.
[2] http://dbpedia.org/resource/Steven_Spielberg.

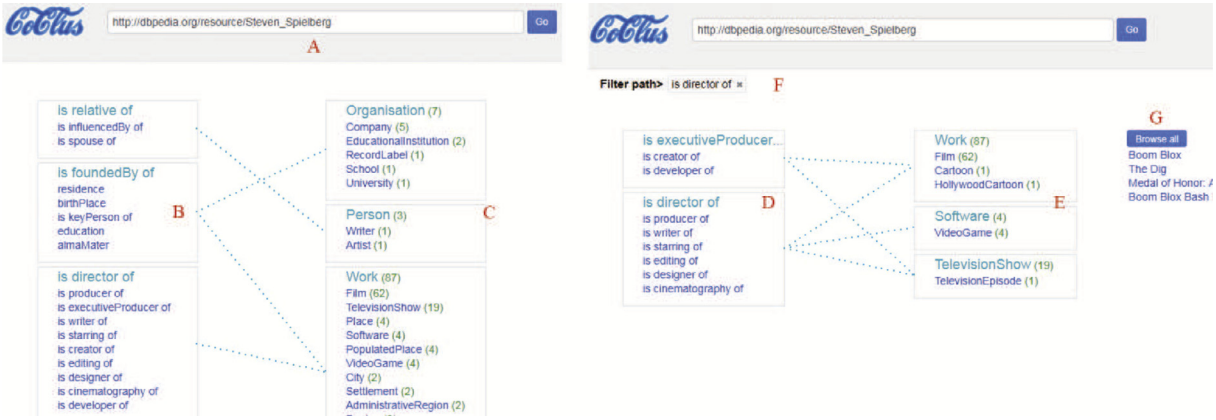

Fig. 3. A screenshot of our browser CoClus.

works on tradition clustering, co-clustering, hierarchical clustering and clustering over Linked Data.

Clustering and similarity measures. Clustering is a fundamental tool in unsupervised learning that is used to group together similar objects [15], and provide an effective mechanism for information exploration. In document search and retrieval, Cutting et al. [5] present a cluster-based approach for browsing large document collections. In Web search, Query/document clustering could be used by search engines to identify the most popular topics and synonyms, or to improve retrieval performance [16,17]. Kaki et al. [18] demonstrate experimentally that clustering can be useful for clarifying and sharpening a vague query by showing users the dominant themes of the returned results. In image recognition, image segmentation [19] simultaneously clusters image features and image pixels.

Finding good similarity functions and selecting better features are the main focus of these clustering works. Existing similarity measures can be classified into either content-based or link-based ones [20]. Content-based measures compute similarities among objects by comparing the contents of the objects involved, such as texts and multimedia. In the various types of contents, these measures mainly utilize the textual information, which is easier to analyze than other types. Measures for computing the similarities among objects using textual information are referred to as text-based similarity measures. Cosine similarity and SVD, LDA, LSI-based similarity measures belong to this category. On the other hand, link-based measures represent the relationships among objects as links and compute the similarities using the link information. The more neighbors the two objects have in common, the higher the similarity between the two becomes. Typical link-based measures include CoCitation [7] and SimRank [8]. There are also some similarity measures on the heterogeneous networks that consist of the objects and links of different types. SimFusion [21] uses an unified relationship matrix (URM) to represent the heterogeneous web objects and the inter-relationships among these web objects. Typical web objects include products, email users, web pages and the like. The similarity matrix of SimFusion is computed iteratively over URM, which helps overcome the data sparseness problem and detect the latent relationships among heterogeneous data objects. PathSim [22] is a meta path-based similarity measure, which provides a meta path to measure similarities from different perspectives. However, it is difficult for the users to choose an appropriate meta path especially when the network schema is unknown.

Inspired by these similarity measures, we measure the intra-link and intra-class similarity based on cosine similarity, lexical similarity and semantic similarity.

Co-clustering. Most clustering algorithms focus on one-way clustering, i.e., cluster one dimension of the table based on similarities along the second dimension. Co-clustering refers to clustering of more than one object type and partitions simultaneously them into subsets. This technique is already applied to a diverse set of applications such as gene expression analysis, text mining and recommendation systems. Dhillon proposes bipartite spectral graph partitioning approach (called BSGP) [23], and then information-theoretic approach (called ITCC) [24] to co-cluster words and documents. BSGP models the document collection as a bipartite graph between documents and words and the simultaneous clustering problem can be posed as a bipartite graph partitioning problem. ITCC views the co-occurrence table as an empirical joint probability distribution of two discrete random variables and poses the co-clustering problem as an optimization problem in information theory – the optimal co-clustering maximizes the mutual information between the clustered random variables subject to constraints on the number of row and column clusters. Recently there are many co-clustering researches. Pereira et al. [25] propose a hybrid recommendation approach called SCOAL (Simultaneous Co-Clustering and Learning), which can address the (pure) cold start problem in recommender systems. Pham et al. [26] develop an interval-valued fuzzy co-clustering for data classification.

More recently, Govaert and Nadif [27] offer two coherent frameworks for the problem of co-clustering. The first involves minimizing an objective function based on measures of association and in particular on phi-squared and mutual information. The second uses a model-based co-clustering approach, and they consider two models: the block model and the latent block model. Different with non-diagonal co-clustering algorithms (e.g., ITCC), Ailem et al. [28] propose a block-diagonal co-clustering algorithm, which is based on the direct maximization of graph modularity.

In this study, we use the well known Information-Theoretic Co-Clustering (ITCC). Furthermore, we define the intra-link and intra-class similarity, and incorporate them into co-clustering.

Hierarchical clustering. Hierarchical clustering is one basic kind of clustering methods, which groups data objects into a hierarchy or "tree" of clusters. It can offer natural facilitation of data navigation and browsing. Steinbach et al. [29] propose a hierarchical divisive version of K-means, called bisecting K-means, which recursively partitions the data into two clusters at each step. Hierarchical co-clustering combines the idea of co-clustering and hierarchical clustering. It aims at simultaneously constructing hierarchical structures for two or more data types, that is, it attempts to achieve the function of both hierarchial clustering and co-clustering. Hosseini et al. [30] introduce a hierarchical co-clustering algorithm for queries and URLs of a search engine log. All queries and URLs are

iteratively clustered for constructing hierarchical categorization. Xu et al. [31] propose a hierarchical divisive co-clustering algorithm to generate topic hierarchy and map each Web page in a large-scale collection onto the computed hierarchy.

Inspired by these works, we introduce a hierarchical co-clustering algorithm to improve the effectiveness of clustering. It tackles every partition of the node as a process of co-clustering.

Clustering over Linked Data. Clustering technique has been widely used for handling RDF data. Most applications organize entities into meaningful groups by using a category system, in order to help users make sense of the entities and decide what to do next. Aemoo [10] is an exploratory search application over the Semantic Web. It groups the entities' neighbours based on DBpedia entity types. Generic RDF browsers (e.g., Tabulator) aggregate linked entities via common semantic link and then presents users with a list of semantic links. Flamenco [32] uses a hierarchical faceted category to help users navigate an image collection.

Our approach is different from the above works. We leverage two entity facets (i.e., entity class and semantic link), and simultaneously group them by using a hierarchical co-clustering approach. To the best of our knowledge, we are the first to introduce co-clustering as an entity exploration mode.

3. Co-clustering links and classes hierarchically

In this section, we describe the details of our hierarchical co-clustering approach. In Section 3.1, we first define a navigation graph to model three aspects of exploration (i.e., linked entities, links and classes). In Section 3.2, we define a measuring scheme to compute the link similarity and class similarity. Then, we present a novel hierarchical co-clustering algorithm in Section 3.3. Finally, we introduce a method for cluster labeling in Section 3.4.

3.1. Constructing navigation graph

When publishing Linked Data on the Web, data is represented using the Resource Description Framework (RDF). It provides a graph data model and is built upon the notion of triple. An RDF triple has the following structure: $<$subject, predicate, object$>$. An entity description is represented as a set of triples. As shown in Fig. 1, there are some RDF triples such as $<$dbr: Steven Spielberg, dbo: birth place, dbr: Cincinnati$>$. An entity classes can be captured via the *rdf:type* predicate, such as $<$dbr: Cincinnati, rdf: type, dbr: City$>$.[3]

We define a navigation graph to describe the relationships among linked entities, links and classes. Let U be a set of entities and classes, L be a set of links including object properties and inverse of them[4], and $T \subseteq U \times L \times U$ be a set of triples.

Definition 1 (Navigation Graph)**.** Given a set of entities $S \subseteq U$ being the focus, a navigation graph G $=$ $<$H, E$>$ consists of a set of vertices $H = L' \cup R \cup C$ where $L' \subseteq L$ denotes a set of links, $R \subseteq U$ a set of linked entities and $C \subseteq U$ a set of classes, and a set of edges E $=$ $\{(l, r) \mid \exists\ s \in S, (s, l, r) \in T\} \bigcup \{(r, c) \mid (r, rdf{:}\ type, c) \in T\}$.

For instance, as shown in Fig. 2, let S= {Steven Spielberg}, L'= {l_1: executive producer^{-1}, l_2: producer^{-1}, l_3: writer^{-1}, l_4: birth place, l_5: residence}[5], R = {r_1: Falling Skies, r_2: Men in Black, r_3: A.I., r_4: Medal of Honor, r_5: Cincinnati, r_6: Los Angeles}, C = {c_1: Television Show, c_2: Film, c_3: Video Game, c_4: City}. Fig. 4 shows an example of navigation graph about Steven Spielberg.

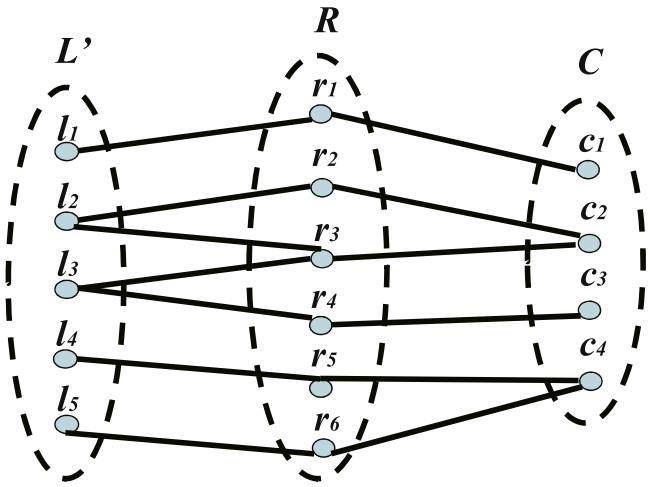

Fig. 4. An example of navigation graph.

Co-clustering links-classes problem can be described as follows: Given a navigation graph $G =< H, E >$ and a similarity function *sim*, find k disjoint clusters of links $\{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_k\}$ and l disjoint clusters of classes $\{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_l\}$ such that $\sum_{i=1}^{k} \sum_{x,x' \in \hat{x}_i} sim(x, x')$ and $\sum_{j=1}^{l} \sum_{y,y' \in \hat{y}_j} sim(y, y')$ are maximized.

Clearly the similarity function is a key facet in solving this problem and we define a measuring scheme in Section 3.2.

3.2. Measuring similarity

In our work, we measure the link similarity and class similarity based on cosine similarity, lexical similarity and semantic similarity. To get the total similarity, we combine these measures based on weighted linear combination.

3.2.1. Cosine similarity

Link and entity class can be represented as a vector over the set of entities based on vector space model [33]. For example, as shown in Fig. 4, the links $l_1 = (1, 0, 0, 0, 0, 0)$, $l_2 = (0, 1, 1, 0, 0, 0)$ and $l_3 = (0, 0, 1, 1, 0, 0)$. We define the link similarity based on the cosine function of angle between two vectors of l_i and l_j:

$$sim_{cos}(l_i, l_j) = \frac{l_i \cdot l_j}{||l_i|| \cdot ||l_j||} \tag{1}$$

and the value of sim_{cos} is in the range [0, 1]. $sim_{cos}(l_1, l_2) = 0$, $sim_{cos}(l_2, l_3) = \frac{1}{2}$. Likewise, the cosine similarity between two classes can be captured according to the above method.

3.2.2. Lexical similarity

The label of a link (class) is useful for human understanding. When the labels of two links (classes) share many common words, it indicates that they are similar. Given two links l_i and l_j, the lexical similarity is defined based on the edit distance between two strings of l_i's label (s_{l_i}) and l_j's label (s_{l_j}):

$$sim_{edit}(l_i, l_j) = \frac{1}{1 + editDist(s_{l_i}, s_{l_j})} \tag{2}$$

where the value of sim_{edit} is in the range (0, 1] and $editDist(s_{l_i}, s_{l_j})$ is the minimum number of character insertion and deletion operations needed to transform one string to the other [34].

[3] Prefix dbr: http://dbpedia.org/resource/ ; dbo: http://dbpedia.org/ontology/ ; rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#.

[4] Object properties link entities to entities; Inverse of properties: Properties have a direction. In practice, people often find it useful to define relations in both directions (e.g., persons own cars, cars are owned by persons). https://www.w3.org/TR/owl-ref/.

[5] We use l^{-1} to denote the inverse of link l.

For instance, $sim_{edit}(producer, executive\ producer) = \frac{1}{1+10} = 0.09$. $sim_{edit}(producer, writer) = \frac{1}{1+8} = 0.11$.

Likewise, the lexical similarity between two classes can be captured according to the above method.

3.2.3. Semantic similarity

In our work, semantic similarity refers to similarity between two concepts in a taxonomy. The semantic similarity is based on path lengths between concepts [35]. We define the semantic similarity between class c_i and c_j as follows:

$$sim_{sem}(c_i, c_j) = \frac{2 \cdot depth(LCA)}{depth(c_i) + depth(c_j)} \tag{3}$$

where the value of sim_{sem} is in the range (0, 1] and $depth(c_i)$ is the shortest path length from the root to c_i and *LCA* is the least common ancestor of c_i and c_j. For instance, the class *Television Show* and *Film* are subclasses of *Work*. $depth(Work) = 2$, $depth(TelevisionShow) = depth(Film) = 3$ and $sim_{sem}(TelevisionShow, Film) = \frac{2\times2}{3+3} = 0.67$.

Also, the semantic similarity between links can be computed according to the above method. For instance, the link *birthPlace* and *residence* are subproperties of *hasLocation*. $depth(hasLocation) = 3$, $depth(birthPlace) = depth(residence) = 4$ and $sim_{sem}(birthPlace, residence) = \frac{2\times3}{4+4} = 0.75$.

3.2.4. Combination of similarity

We discuss three similarity measures from different points of view. In order to get the total similarity, we combine these measures based on weighted linear combination. The similarity scoring function of the two classes c_i and c_j is defined as follows.

$$sim(c_i, c_j) = \alpha \cdot sim_{cos}(c_i, c_j) + \beta \cdot sim_{edit}(c_i, c_j) + \gamma \cdot sim_{sem}(c_i, c_j) \tag{4}$$

where $\alpha + \beta + \gamma = 1$ and $\alpha, \beta, \gamma \in [0, 1]$ indicate the weights for each similarity measure to be tuned empirically.

Also, the similarity function of the two links l_i and l_j can be captured according to the above method.

3.3. Hierarchical co-clustering

Co-clustering refers to clustering of more than one data type [23]. Here, leveraging the relationships between the links and entity classes, we simultaneously cluster both links and entity classes. There are many co-clustering techniques [36] and we use a well-known co-clustering algorithm based on information theory [24], which defines co-clustering as a pair of maps from rows to row-clusters and from columns to column-clusters. The optimal co-clustering is one that minimizes the difference ("loss") in mutual information between the original random variables and the mutual information between the clustered random variables.

Let $X = \{x_1, \ldots, x_m\}$ and $Y = \{y_1, \ldots, y_n\}$ be discrete random variables respectively. Let $p(X, Y)$ denote the joint probability distribution between X and Y. Let the k disjoint clusters of X be written as: $\hat{X} = \{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_k\}$, and the l disjoint clusters of Y be written as: $\hat{Y} = \{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_l\}$. An optimal co-clustering minimizes the loss of mutual information, defined as

$$I(X;Y) - I(\hat{X};\hat{Y}) = D(p(X,Y)||q(X,Y)) \tag{5}$$

where $I(X; Y)$ is the mutual information between sets X and Y, $D(\cdot\|\cdot)$ denotes the Kullback–Leibler (KL) divergence, and $q(X, Y)$ is a distribution of the form $q(x, y) = p(\hat{x}, \hat{y})p(x|\hat{x})p(y|\hat{y})$, $x \in \hat{x}$, $y \in \hat{y}$.

In our context, the link set L' can be considered as X and the class set C as Y. The joint probability distribution between links and classes can be captured based on entity set R over navigation graph $G = < L' \cup R \cup C, E >$. The joint probability of a link x and a class y is defined as follows:

$$p(x, y) = \frac{|\{r|\exists r \in R, (x, r) \in E\} \cap \{r|\exists r \in R, (r, y) \in E\}|}{|R|} \tag{6}$$

We illustrate an execution by means of a small example co-clusters. As shown in Fig. 4, $X = [l_1, l_2, l_3, l_4, l_5]$, $Y = [c_1, c_2, c_3, c_4]$. We can compute the joint probability of the link l_2 ($producer^{-1}$) and class c_2 (*Film*), $p(l_2, c_2) = \frac{2}{6}$ ($|R| = 6$). We can obtain the joint distribution matrix between X and Y,

$$p(X,Y) = \begin{pmatrix} \frac{1}{6} & 0 & 0 & 0 \\ 0 & \frac{2}{6} & 0 & 0 \\ 0 & \frac{1}{6} & \frac{1}{6} & 0 \\ 0 & 0 & 0 & \frac{1}{6} \\ 0 & 0 & 0 & \frac{1}{6} \end{pmatrix}$$

Looking at the row distributions it is natural to group the rows into three clusters: $\hat{x}_1 = \{l_1\}$, $\hat{x}_2 = \{l_2, l_3\}$ and $\hat{x}_3 = \{l_4, l_5\}$. Similarly the natural column clustering is: $\hat{y}_1 = \{c_1\}$, $\hat{y}_2 = \{c_2, c_3\}$ and $\hat{y}_3 = \{c_4\}$. The resulting joint distribution $p(\hat{X}, \hat{Y})$ is given by:

$$p(\hat{X},\hat{Y}) = \begin{pmatrix} \frac{1}{6} & 0 & 0 \\ 0 & \frac{3}{6} & 0 \\ 0 & 0 & \frac{2}{6} \end{pmatrix}$$

It can be verified that the mutual information lost due to this co-clustering is 0.1799, and that any other co-clustering leads to a larger loss in mutual information.

However, it neglects the intra-link and intra-class similarity. $\hat{x}_1$ and $\hat{x}_2$ can be clustered because the link l_1, l_2 and l_3 are similar based on semantic similarity. To improve co-clustering, many research studies utilize these intra-relationships [37,38]. We incorporate the intra-link and intra-class similarity into the information theoretic co-clustering. The Eq. (5) is rewritten as

$$I(X;Y) - I(\hat{X};\hat{Y}) + \lambda LCS + \mu CCS \tag{7}$$

$$LCS = \frac{2}{|\hat{X}|(|\hat{X}|-1)} \sum_{i>j} sim(\hat{x}_i, \hat{x}_j) \tag{8}$$

$$sim(\hat{x}_i, \hat{x}_j) = \sum_{x \in \hat{x}_i, x' \in \hat{x}_j} \frac{sim(x, x')}{|\hat{x}_i| \cdot |\hat{x}_j|} \tag{9}$$

$$CCS = \frac{2}{|\hat{Y}|(|\hat{Y}|-1)} \sum_{i>j} sim(\hat{y}_i, \hat{y}_j) \tag{10}$$

$$sim(\hat{y}_i, \hat{y}_j) = \sum_{y \in \hat{y}_i, y' \in \hat{y}_j} \frac{sim(y, y')}{|\hat{y}_i| \cdot |\hat{y}_j|} \tag{11}$$

LCS and *CCS* are the total similarity among link clusters and class clusters respectively. Obviously, the lower the *LCS* and *CCS* scores are, the better the clustering quality is. It denotes that the differences of resulted clusters are large. $\lambda + \mu = 1$ and $\lambda, \mu \in [0, 1]$ indicate the weights for the trade off among the loss of mutual information, *LCC* and *CCS*.

We iteratively optimize the objective function (Eq. (7)) by link clustering and class clustering. So minimizing (7) is equal to minimize Eq. (12), or minimize Eq. (13).

$$\sum_{\hat{x}} \sum_{x \in \hat{x}} D(p(Y|x)||q(Y|\hat{x})) + \lambda LCS \tag{12}$$

$$\sum_{\hat{y}} \sum_{y \in \hat{y}} D(p(X|y)||q(X|\hat{y})) + \mu CCS \tag{13}$$

We present our algorithm based on the information theoretic co-clustering (ITCC [24]) in Algorithm 1.

Continuing with the above example, the row distributions group the rows into two clusters: $\hat{x}_1 = \{l_1, l_2, l_3\}$ and $\hat{x}_2 = \{l_4, l_5\}$.

Algorithm 1: Co-clustering based on ITCC and intra-similarity.

Input: $p(X, Y)$: the joint probability distribution; k: the number of link clusters; l: the number of class clusters; parameters λ and μ

Output: $\hat{X}$: the link cluster sets; $\hat{Y}$: the class cluster sets

1 Initialize: Set $t = 0$. Start with some initial link and class cluster set $\hat{X}^{(0)}$ and $\hat{Y}^{(0)}$. Compute $q^{(0)}(\hat{X}, \hat{Y})$, $q^{(0)}(X|\hat{X})$, $q^{(0)}(Y|\hat{Y})$ and the distributions $q^{(0)}(Y|\hat{x})$, $1 \le \hat{x} \le k$, as done in [24].

2 Compute link clusters: For each link x, find its new cluster index as $\hat{X}^{(t+1)}(x) = argmin_{\hat{x}} D(p(Y|x)||q^{(t)}(Y|\hat{x})) + \lambda LCS$, resolving ties arbitrarily. Let $\hat{Y}^{(t+1)} = \hat{Y}^{(t)}$.

3 Compute distributions $q^{(t+1)}(\hat{X}, \hat{Y})$, $q^{(t+1)}(X|\hat{X})$, $q^{(t+1)}(Y|\hat{Y})$ and the distributions $q^{(t+1)}(X|\hat{y})$, $1 \le \hat{y} \le l$.

4 Compute class clusters: For each class y, find its new cluster index as $\hat{Y}^{(t+2)}(y) = argmin_{\hat{y}} D(p(X|y)||q^{(t+1)}(X|\hat{y})) + \mu CCS$, resolving ties arbitrarily. Let $\hat{X}^{(t+2)} = \hat{X}^{(t+1)}$.

5 Compute distributions $q^{(t+2)}(\hat{X}, \hat{Y})$, $q^{(t+2)}(X|\hat{X})$, $q^{(t+2)}(Y|\hat{Y})$ and the distributions $q^{(t+2)}(Y|\hat{x})$, $1 \le \hat{x} \le k$.

6 Stop and return $\hat{X} = \hat{X}^{(t+2)}$ and $\hat{Y} = \hat{Y}^{(t+2)}$ if the change in objective function value, that is, $D(p(X,Y)||q^{(t)}(X,Y)) - D(p(X,Y)||q^{(t+2)}(X,Y)) + \lambda(LCS^{(t)} - LCS^{(t+2)}) + \mu(CCS^{(t)} - CCS^{(t+2)})$, is "small"; Else set $t = t + 2$ and go to step 2.

Similarly the natural column clustering is: $\hat{y}_1 = \{c_1, c_2, c_3\}$ and $\hat{y}_2 = \{c_4\}$. The resulting joint distribution $p(\hat{X}, \hat{Y})$ is given by:

$$p(\hat{X}, \hat{Y}) = \begin{pmatrix} \frac{4}{6} & 0 \\ 0 & \frac{2}{6} \end{pmatrix}.$$

Although the mutual information lost is not changed (is still 0.1799), the scores of *LCS* and *CCS* become smaller.

We implement the ITCC algorithm with time complexity $O((nz(k+l) + km^2 + ln^2)\tau)$ provided by [37], where nz is the number of non-zeros in $p(X, Y)$ and τ is the number of iterations.

However, the above algorithm has the following limitations. It could obtain various local minima when starting from an initial random partition. Moreover, during each iteration, it needs to compute the joint distribution over the whole data set.

To improve the efficiency of clustering, we introduce a hierarchical co-clustering algorithm (HCC), which combines co-clustering and hierarchical clustering. It leverages the idea of hierarchical clustering and tackles every partition of the node as a process of co-clustering. Compared with the previous algorithm, this algorithm only computes the joint distribution over the partial data set. Furthermore, a hierarchy can enable users to explore information in a multilevel fashion, and users can decide whether to divide the nodes.

HCC employs a top-down strategy. It starts by placing all the links and classes in one cluster, which is the hierarchy's root node. It then divides the root cluster into several smaller subclusters, and recursively partitions those clusters into smaller ones. The partition of every node can be used by ITCC co-clustering algorithm (Algorithm 1). The partitioning process continues until each cluster at the lowest level is coherent enough, or it satisfies specified constraints (we specify the desired number of leaf nodes as a termination condition). At every partitioning process, we empirically use a smaller number of clusters k and l (e.g., $k, l = 2$). The detail of HCC is in Algorithm 2.

Fig. 5 shows an illustration of the link and class cluster hierarchy associated with Fig. 2. When we handle the node $\{\{l_1, l_2, l_3\}, \{c_1, c_2, c_3\}\}$, it can be divided into two nodes $\{\{l_1\}, \{c_1\}\}$ and $\{\{l_2, l_3\}, \{c_2, c_3\}\}$ by using Algorithm 1. Because the pair

Algorithm 2: Hierarchical co-clustering.

Input: X: the link set; Y: the class set; N: the number of clusters

Output: CH: the link and class cluster hierarchy

1 Create an empty hierarchy CH.
2 $TopNode \leftarrow X \cup Y, CH \leftarrow CH \cup TopNode$;
3 $LeafNode \leftarrow TopNode$;
4 **while** $|LeafNode| \le N$ **do**
5 **foreach** $node \in LeafNode$ **do**
6 $\hat{X} \leftarrow \emptyset, \hat{Y} \leftarrow \emptyset$; // the link and class clusters.
7 compute $p(X, Y)$ for $node$;
8 $\hat{X}, \hat{Y} \leftarrow$ Co-Clustering$(p(X, Y), k, l, \lambda, \mu)$;
9 $CH \leftarrow CH \cup \hat{X} \cup \hat{Y}$.
10 **if** *CH does not change* **then**
11 break ;
12 $LeafNode \leftarrow$ the leaf nodes of CH ;
13 **return** CH;

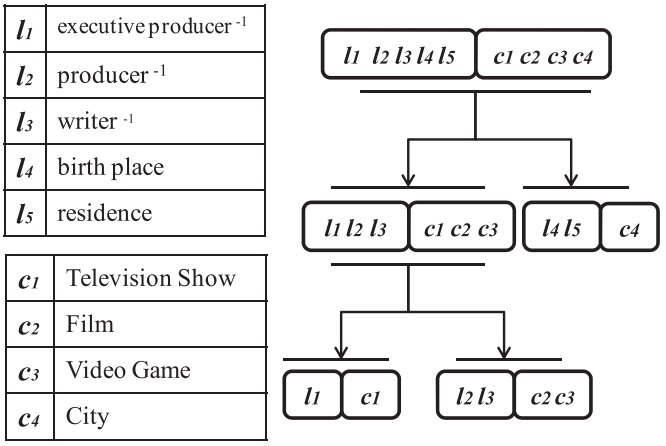

Fig. 5. The hierarchy of link and class clusters associated with Fig. 2.

of $l_2(producer^{-1})$ and $l_3(writer^{-1})$, and the pair of $c_2(Film)$ and $c_3(VideoGame)$ are most similar.

3.4. Making the clusters easy to browse

To help user decide at a glance whether the contents of a cluster are of interest, we aim to provide concise and accurate cluster description. A heuristic method is to find the cluster's centroid which is most similar with other elements in the cluster as the label of cluster.

Given a class cluster $\hat{y}$ and a class $y \in \hat{y}$, we define the centricity of y as follows:

$$centricity(y) = \sum_{y' \in \hat{y}} sim(y, y'). \tag{14}$$

Thus, the centroid of cluster $\hat{y}$ can be defined as follows:

$$centroid(\hat{y}) = \arg\max_{y \in \hat{y}} centricity(y) \tag{15}$$

Also, the link cluster's centroid can be captured according to the above method.

4. Evaluation

To evaluate exploration support provided by our approach, we implement it in a prototype system called CoClus, and compare

it with three baseline systems via a task-based user study in Section 4.1. Then, we evaluate the clustering performance of our approach in comparison with three representative co-clustering algorithms in Section 4.2. Furthermore, we discuss the scalability of our approach by measuring the average execution time in Section 4.3. Finally, we discuss evaluation results, limits and possible solutions to improve our approach in Section 4.4.

4.1. User study

To investigate how our approach help users explore entities in practice, we invited human users to carry out entity exploration tasks by using our system CoClus,[6] two restricted versions of CoClus and one conventional LD browser (Rhizomer[7]). By analyzing subjects' responses to questionnaires and their behavior during the experiment, we mainly aimed to test the following two hypotheses.

- Hypothesis 1: Hierarchical co-clustering links and entity classes provide a hierarchical organization with multi-granularity, and thus exploiting them can effectively help humans in entity exploration.
- Hypothesis 2: Link and entity class clusters are notably complementary in terms of usage in entity exploration, and thus providing both of them (as on CoClus) is more satisfying than only one of them (as on LinkClus and ClassClus).

4.1.1. Experimental setting

Tasks. We used DBpedia[8] covering a large amount of broad-ranging entities. It allowed us to design exploration tasks that were not targeted at a particular domain. Our entity exploration tasks were derived from 750 popular search terms in the Google trends 2015.[9] From these search terms, we chose 10 queried entities. These 10 entities can be found in DBpedia and had more than 5 different links each.

For each entity, we established the exploration tasks including entity-centric summarization and related entity finding. For instance, the exploration tasks about Jennifer Lopez[10] were designed as follows: *Suppose you will write a summary about Jennifer Lopez including at least three main aspects, and then find the films she produced.* Finally, the 10 tasks were to be used in the user study (Appendix A).

Participant systems. To evaluate whether co-clustering links and classes are really helping the user in his exploration, we deployed two restricted versions of our system: LinkClus (providing only the link clusters) and ClassClus (providing only the class clusters). We also compared one active LD browser Rhizomer [40], which provides a user interface with HTML and different facets (links and entity classes). Users can zoom into the linked entities by selecting different kinds of facets.

Procedure. We invited 24 student subjects majoring in computer science. They were familiar with the Web, but with no knowledge of our project. The evaluation procedure was performed as follows.

- Before the evaluation session, the subjects learned how to use the given systems through a 20 min tutorial. In the tutorial, for each system, we demonstrated its functionality. Then, the subject was given one task as a warmup. To avoid bias as far as possible, we used the same example to illustrate and subjects took the same task as a warmup. In addition, there was 10 min for free use and questions.
- Then, the subjects used each of the given systems arranged in random order. For each system, the subjects were randomly assigned to exploration tasks. Meanwhile, these tasks among the given systems were different. The subjects were asked to complete all the tasks in 30 min. We recorded their answers, activities, and the time they spent on each task, until the subject concluded the experiment with an explicit action of termination.
- Finally, with regard to each system, the subjects responded to the post-task questionnaire on a 5-point Likert scale, to give us feedback on the quality of exploration. The subjects also gave feedback and suggestions on the given systems.

Table 1
Post-task questionnaire.

	Questions
Q1:	System had an overview of all the information.
Q2:	System had appropriate size of navigation options.
Q3:	It was easy to reorient myself in the exploration.
Q4:	System was easy-to-use with no prior knowledge.
Q5:	System offered helpful support for this task.

Evaluation metrics. Three metrics were measured in the experiments: task time, number of user actions and user satisfaction. (1) Task time referred to the average time used by a subject for carrying out one task based on a given system. (2) Number of user actions was a complement to measures of usability. A small number of user actions indicated that the system could offer easy-to-use function without requiring too many user interactions. (3) User satisfaction was measured after performing the test by using a satisfaction questionnaire, as shown in Table 1.

4.1.2. Results

Post-task questionnaire Q1–Q5 captured subjects' exploration experience with different systems in Table 2. Repeated measures ANOVA revealed that the differences in subjects' mean ratings were all statistically significant ($p < 0.01$). LSD post-hoc tests ($p < 0.05$) revealed that, according to Q1, CoClus provided the best overview of all the information due to the use of both link and class clusters. Besides, LinkClus and ClassClus had a better overview than Rhizomer. According to Q2, Rhizomer provided top-k faceted filtering views in comparison with CoClus, LinkClus and ClassClus. According to Q3, Rhizomer helped subjects keep track of browsing and provided easy rollback. According to Q4, CoClus, LinkClus and ClassClus were easy to use without requiring prior knowledge. Finally, according to Q5, CoClus provided subjects with more helpful support for exploration than other three systems.

Fig. 6 shows the average time and number of actions spent in exploration tasks for the four systems respectively. Since CoClus provided an overview using link clusters and class clusters, subjects took far less time (30 s) and fewer interactions (17 times) to complete these tasks.

These results were consistent with subjects' experience and behavior reported previously. All of these collectively supported our hypotheses H1 and H2. Hierarchical co-clustering links and entity classes provided a multi-granular, progressive exploration assistance. Link and entity class clusters were complementary because they provided an alternative filtering mode to help refocus on a particular entity collection.

[6] http://ws.nju.edu.cn/coclus/. Also, our project code is uploaded and downloaded freely on https://github.com/waynezheng/coclus.
[7] http://rhizomik.net/html/rhizomer/.
[8] http://wiki.dbpedia.org.
[9] https://www.google.com/trends/topcharts.
[10] http://dbpedia.org/resource/Jennifer_Lopez.

Table 2
Results of post-task questionnaire.

	Response: Mean (SD)				$F(3, 92)$	LSD post-hoc
	Rhizomer	LinkClus	ClassClus	CoClus	(p-value)	($p < 0.05$)
Q1:	3.56 (0.85)	3.8 (1.056)	3.93 (0.95)	**4.18** (1.095)	11.543 (0.000)	CoClus > LinkClus, ClassClus > Rhizomer
Q2:	**3.67** (1.052)	3.45 (1.014)	3.34 (1.32)	3.02 (0.887)	9.56 (0.000)	Rhizomer, LinkClus, ClassClus > CoClus
Q3:	**3.83** (0.514)	3.667 (0.778)	3.78 (1.031)	3.5 (0.937)	11.67 (0.000)	Rhizomer > CoClus, LinkClus, ClassClus
Q4:	3.25 (0.793)	3.98 (0.855)	4.18 (1.055)	**4.33** (0.778)	10.36 (0.000)	CoClus, LinkClus, ClassClus > Rhizomer
Q5:	3.74 (0.717)	3.51 (0.582)	3.42 (1.02)	**4.02** (0.62)	13.65 (0.000)	CoClus > LinkClus, ClassClus > Rhizomer

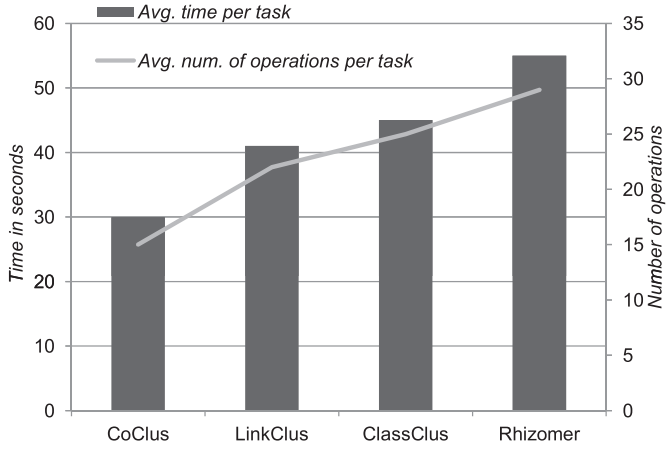

Fig. 6. Average time and number of user actions in exploration tasks for four systems.

4.2. Performance evaluation

In this experiment, we compared the performance of our algorithm (hierarchical co-clustering, HCC) with that of three representative co-clustering algorithms.

4.2.1. Experimental setting

Datasets. We performed experiments on two real-world datasets.

- DBpedia[11] (version 2015-10) is a central interlinking hub of the emerging RDF data on the Web. Specifically, the candidate entity collection was obtained from the *mapping-based properties* dataset. Classes of entities were obtained from the *mapping-based types* dataset. Class and link hierarchies were obtained from the *DBpedia ontology*. From the candidate entity collection, we collected those entities that had more than 15 different links each.
- YAGO[12] is a knowledge base which is extracted from Wikipedia and other sources. The candidate entity collection was obtained from the *yagoFacts* dataset. Classes of entities were obtained from the *yagoSimpleTypes* dataset. Class and link hierarchies were obtained from the *yagoTaxonomy* and *yagoSchema*. From the candidate entity collection, we collected those entities that had more than 6 different links each.

The detailed distribution of the three datasets is listed in Table 3.

Table 3
Statistics of experimental datasets.

	DBpedia	YAGO
Entities	18,329	15,035
Average number of linked entities per entity	217.5	103.2
Average number of links per entity	25.8	10.1
Average number of classes per linked entity	9.6	38.7

Evaluation baselines and parameter settings. We compare our algorithm (HCC) with three representative algorithms (ITCC, co-clustering via bipartite spectral graph partition(BSGP) and DiagCoClus).

- ITCC focuses on the relationship between row and column in the co-occurrence matrix from mutual information. In addition, we incorporate the intra-link and intra-class similarity into co-clustering (Algorithm 1) [14].
- BSGP [23] considers the co-clustering problem in term of finding minimum cut vertex partitions in a weighted bipartite graph. In our context, given a navigation graph $G =< L' \cup R \cup C, E >$, the link-class bipartite graph LC can be defined as follows: $LC = < V, E_{lc} >$ where $V = L' \cup C$ and $E_{lc} = \{(l, c) \mid \exists r \in R, (l, r) \in E \wedge (r, c) \in E\}$. We use the edge weighting method provided by Giannakidou et al. [39] and the weight between a link l and a class c is computed with

$$w(l, c) = \max_{c_i \in C_l} sim(c_i, c) + \max_{l_j \in L_c} sim(l, l_j) \tag{16}$$

  where $C_l = \{c' | \exists c' \in C, \ (l, \ c') \in E_{lc}\}$ and $L_c = \{l' | \exists l' \in L', \ (l', c) \in E_{lc}\}$.
- DiagCoClus [28] is a block-diagonal co-clustering algorithm which directly maximizes this modularity measure.[13] Different with the above algorithms, DiagCoClus does not take into account the intra-link and intra-class similarity.

For parameter settings, we investigated the sensitivity with respect to the disjoint clusters size k (=3, 5, 8) and the balanced parameters (α, β, γ, λ, μ). As to the parameters (α, β, γ) in similarity Eq. (4), we empirically set $t_1 = (1, 0, 0)$, $t_2 = (0, 1, 0)$, $t_3 = (0, 0, 1)$, $t_4 = (0.33, 0.33, 0.33)$, $t_5 = (0.6, 0.1, 0.3)$ and $t_6 = (0.3, 0.1, 0.6)$ in computing link similarity and class similarity. As to the parameters (λ, μ) of loss in mutual information Eq. (7), we set $\lambda = \mu = 0.5$ based on equity. More parameters settings will be experimented in future work. Besides, we set the number of iterations $\tau = 20$ which is enough for convergence [24].

[11] http://wiki.dbpedia.org.

[12] http://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/downloads/.

[13] The algorithm is available in online coclust package. We use the function "load_doc_term_data" to load our experimental data, and use the function "CoclustMod" to execute co-clustering by direct maximization of graph modularity. To avoid confusion between the "Coclus" and "Coclust" package, Coclust is called DiagCoClus in this paper. (https://pypi.python.org/pypi/coclust).

Table 4
The *CI* scores with different k and (α, β, γ) on DBpedia.

		t_1	t_2	t_3	t_4	t_5	t_6
k=3	HCC	0.69	0.62	0.45	0.54	**0.77**	0.65
		0.54	0.58	0.62	0.48	0.68	0.72
	ITCC	0.65	0.57	0.4	0.23	0.7	0.63
		0.52	0.41	0.49	0.33	0.62	**0.73**
	BSGP	0.56	0.56	0.37	0.28	0.46	0.43
		0.29	0.26	0.43	0.35	0.58	0.63
	DiagCoClus			0.61			
				0.65			
k=5	HCC	0.65	0.69	0.47	0.53	**0.76**	0.64
		0.51	0.54	0.59	0.41	**0.74**	0.73
	ITCC	0.57	0.48	0.42	0.37	0.71	0.47
		0.43	0.34	0.47	0.32	0.69	0.51
	BSGP	0.49	0.27	0.21	0.21	0.52	0.38
		0.39	0.39	0.57	0.19	0.51	0.42
	DiagCoClus			0.72			
				0.68			
k=8	HCC	0.67	0.66	0.57	0.63	**0.83**	0.78
		0.58	0.59	0.65	0.67	0.81	**0.86**
	ITCC	0.63	0.56	0.51	0.34	0.78	0.71
		0.52	0.49	0.59	0.42	0.76	0.82
	BSGP	0.58	0.38	0.38	0.27	0.59	0.56
		0.54	0.34	0.61	0.33	0.61	0.66
	DiagCoClus			0.74			
				0.78			

Table 5
The *CI* scores with different k and (α, β, γ) on YAGO.

		t_1	t_2	t_3	t_4	t_5	t_6
k=3	HCC	0.66	0.65	0.64	0.71	0.78	**0.79**
		0.54	0.51	0.57	0.67	**0.73**	0.64
	ITCC	0.61	0.61	0.59	0.69	0.73	0.71
		0.51	0.55	0.64	0.63	0.61	0.56
	BSGP	0.62	0.59	0.37	0.58	0.68	0.63
		0.51	0.48	0.28	0.55	0.51	0.55
	DiagCoClus			0.75			
				0.69			
k=5	HCC	0.69	0.62	0.63	0.75	**0.81**	0.7
		0.53	0.53	0.57	0.67	**0.76**	0.66
	ITCC	0.64	0.61	0.54	0.69	0.77	0.67
		0.51	0.52	0.53	0.55	0.71	0.59
	BSGP	0.53	0.55	0.51	0.55	0.69	0.56
		0.52	0.34	0.56	0.32	0.5	0.49
	DiagCoClus			0.71			
				0.72			
k=8	HCC	0.7	0.61	0.68	0.79	**0.87**	0.73
		0.55	0.67	0.61	0.71	**0.79**	0.66
	ITCC	0.67	0.57	0.62	0.73	0.71	0.69
		0.55	0.61	0.57	0.62	0.74	0.61
	BSGP	0.69	0.53	0.58	0.67	0.61	0.61
		0.54	0.54	0.45	0.53	0.51	0.58
	DiagCoClus			0.78			
				0.71			

For each experimental dataset, we randomly selected 200 entities. As to every selected entity, we empirically conducted 10 runs using four algorithms (HCC, ITCC, BSGP and DiagCoClus) and reported the average CI score.

Evaluation metric. In order to determine the efficiency of each clustering algorithm, we used the evaluative criteria of Clustering Index (CI) [41]. Intuitively, since the most efficient clusters are the ones containing objects close to each other within the cluster, while sharing a low similarity with objects belonging to different clusters, CI focuses on increasing the first measure (intra-cluster similarity) while decreasing the second (inter-cluster similarity). The Clustering Index is defined as:

$$CI = \frac{\overline{\sigma}^2}{\overline{\sigma} + \overline{\delta}} \tag{17}$$

where $\overline{\sigma}$ is the average intra-cluster similarity and $\overline{\delta}$ the average inter-cluster similarity. Obviously, the higher the CI score is, the better the clustering quality is.

Given a cluster set $O = \{O_1, \ldots, O_k\}$ of N elements, the average intra-cluster similarity $\overline{\sigma}$ and the average inter-cluster similarity $\overline{\delta}$ are defined as follows.

$$\overline{\sigma} = \frac{1}{k}\sum_{i=1}^{k}\sigma(O_i),\ \ \sigma(O_i) = \frac{\sum_{o\in O_i, o'\in O_i} sim(o, o')}{|O_i|\cdot(|O_i|-1)} \tag{18}$$

$$\overline{\delta} = \frac{2}{k(k-1)}\sum_{i>j}\delta(O_i, O_j),\ \ \delta(O_i, O_j) = \frac{\sum_{o\in O_i, o'\in O_j} sim(o, o')}{|O_i|\cdot|O_j|} \tag{19}$$

The similarity function *sim* is defined in Eq. (4), and we use CI to evaluate the quality of link clusters and class clusters respectively.

Results. Tables 4 and 5 show the comparisons of clustering quality in datasets DBpedia and YAGO respectively. Each cell shows the Clustering Index (CI) score of clustering results (the upper of cell denotes CI score of link clusters, and the lower denotes CI score of class clusters). Specially, DiagCoClus had two CI scores under different balanced parameters (α, β, γ), since it does not consider the intra-link and intra-class similarity. A comparative analysis based on these tables is given below.

(1) In most cases, HCC performs better than other three algorithms. DiagCoClus behaves slightly better than HCC in some cases. DiagCoClus outperforms ITCC and BSGP in most cases on all datasets, and ITCC has a better performance than BSGP.
(2) As to the influence of the three parameters (α, β, γ) in similarity computing, we investigate how the three parameters influence the CI scores under the same k and algorithm. The better CI scores of link clusters are observed when the cosine or lexical similarity has higher weight (t_1 and t_2). On the contrary, the better CI scores of class clusters are observed when the semantic similarity has higher weight (t_3 and t_6). It shows that the cosine and lexical similarity are a better fit for link clustering, and the semantic similarity seems to be better fit to class clustering in our experimental datasets. More datasets will be experimented in future work.

4.3. Efficiency evaluation

We evaluated the scalability of HCC, ITCC, BSGP and DiagCoClus by measuring the average execution time for varying size of entities denoted by m. We randomly selected 500 entities from our experimental datasets. The four algorithms were implemented in Java and carried out on an Intel Core2 Quad 2.66 GHz CPU, Windows 7 with 1.2GB JVM.

As can be seen from Fig. 7, DiagCoClus is the fastest, and BSGP is faster than HCC and ITCC. Because HCC and ITCC need multiple iterations and recompute the distributions on every iteration. Besides, HCC computes the link cluster similarity and class cluster similarity respectively on every iteration. In all, with the increasing of m, HCC is not significantly slower than other algorithms (it took at most 4 s for 200 linked entities). It can be further optimized by pre-indexing the values of link and class similarity.

4.4. Discussions

We use human labor to assess our approach in user study. We summarized all the major comments of the subjects. The results show that our approach not only allow exploration of linked entities in a hierarchical multiscale fashion, but also provide a useful support for an alternative exploration mode.

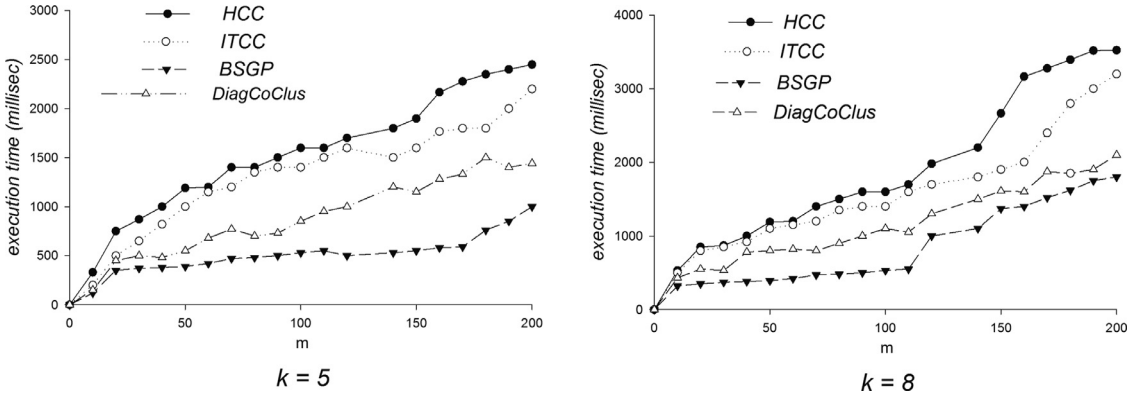

Fig. 7. Execution time of four algorithms with varying k and m.

To validate the efficiency of our algorithm, we compare it with other three algorithms using the clustering index (CI) score. The results show that our algorithm outperforms other algorithms in most cases on all datasets. By analyzing the average execution time of our algorithm, it is reasonably fast in practice.

There are some potential threats to the validity of our study. For instance, to avoid completing the tasks in an unserious way, we ask the subjects to perform the tasks under our supervision. To avoid subjects' understanding bias towards our tasks, we explain the goal of our tasks through a free question and answer period. To avoid mishandling the experimental systems, we ask the subjects to learn how to use the given systems through a tutorial. Furthermore, we use the same task to illustrate the systems and the subjects take the same task as a warmup. These solutions could reduce the validity threats as far as possible.

5. Conclusion

In this paper, we propose a hierarchical co-clustering approach to facilitate entity exploration. In our approach, we first define a measure of intra-link similarity and intra-class similarity respectively. Secondly, we incorporate them into co-clustering. Finally, we implement our approach in a Linked Data browser CoClus. Extensive evaluations demonstrate that our approach provides useful support for entity exploration.

There are some directions for future work. We will investigate more cluster labeling approaches (e.g., Topic Modeling) to make users easily understand the results. Furthermore, we will study "human factors" in the context of entity navigation. For example, users' preference on link and class can be collected and then leveraged to rank the clusters more intelligently.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grant No. 61772264 and 61572247. We are also grateful to all the participants in the experiments of this work.

References

[1] T. Heath, C. Bizer, Linked data: evolving the web into a global data space, Synthesis Lectures on the Semantic Web: Theory and Technology, Morgan & Claypool Publishers, 2011.
[2] G. Klyne, J.J. Carroll, Resource description framework (RDF): concepts and abstract syntax–w3c recommendation, 2004.
[3] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, S. Hellmann, DBpedia-a crystallization point for the web of data, J. Web Sem. 7 (3) (2009) 154–165.
[4] M.A. Hearst, Clustering versus faceted categories for information exploration, Commun. ACM 49 (4) (2006) 59–61.
[5] D.R. Cutting, D.R. Karger, J.O. Pedersen, J.W. Tukey, Scatter/gather: a cluster-based approach to browsing large document collections, in: Proceedings of 15th Annual International ACM SIGIR Conference Research and Development in Information Retrieval, 1992, pp. 318–329.
[6] S.H. Yoon, S.W. Kim, J.S. Kim, W.S. Hwang, On computing text-based similarity in scientific literature, Proceedings of 20th International Conference on World Wide Web (2011) 169–170.
[7] H. Small, Cocitation in the scientific literature: a new measure of the relationship between two documents, J. Am. Soc. Inf. Sci. 24 (4) (1973) 265–269.
[8] G. Jeh, J. Widom, Simrank: a measure of structural-context similarity, in: Proceedings of 8th ACM SIGKDD, 2002, pp. 538–543.
[9] T.B. Lee, Y. Chen, L. Chilton, D. Connolly, R. Dhanaraj, J. Hollenbach, A. Lerer, D. Sheets, Tabulator: exploring and analyzing linked data on the semantic web, in: Proceedings of the 3rd International Semantic Web User Interaction Workshop, 2006.
[10] A. Musetti, A.G. Nuzzolese, F. Draicchio, Aemoo: exploring knowledge on the web, in: WebSci, 2013, pp. 272–275.
[11] M. Hildebrand, J.v. Ossenbruggen, L. Hardman, /facet: a browser for heterogeneous semantic web repositories, in: 14th International Semantic Web Conference, 2006.
[12] E. Oren, R. Delbru, S. Decker, Extending faceted navigation for rdf data, in: Proceedings of the 14th International Semantic Web Conference, 2006.
[13] N. Bikakis, M. Skourla, G. Papastefanatos, rdf:synopsviz: a framework for hierarchical linked data visual exploration and analysis, in: Proceedings of ESWC, 2014.
[14] L. Zheng, J. Xu, J. Jiang, Y. Qu, G. Cheng, Iterative entity navigation via co–clustering semantic links and entity classes, in: Proceedings of ESWC, 2016, pp. 168–181.
[15] A.K. Jain, R.C. Dubes, Algorithms for Clustering Data, Prentice-Hall, Englewood Cliffs, NJ, 1988.
[16] D. Beeferman, A. Berger, Agglomerative clustering of a search engine query log, in: 6th ACM SIGKDD, 2000, pp. 407–416.
[17] J. Wen, J.Y. Nie, H. Zhang, Query clustering using user logs, in: ACM Transactions on Information Systems, 20, 2002, pp. 59–81.
[18] M. Kaki, Findex: search result categories help users when document rankings fail, ACM SIGCHI, 2005.
[19] T. Hofmann, J. Puzicha, J.M. Buhmann, Unsupervised texture segmentation in a deterministic annealing framework, in: IEEE Transactions on Pattern Analysis and Machine Intelligence, 20, 1998, pp. 803–818.
[20] F. Menczer, Combining link and content analysis to estimate semantic similarity, in: Proceedings of 13th International Conference World Wide Web, 2004, pp. 452–453.
[21] W. Xi, E.A. Fox, W. Fan, B. Zhang, Z. Chen, J. Yan, et al., Simfusion: measuring similarity using unified relationship matrix, in: Proceedings of SIGIR, 2005, pp. 130–137.
[22] Y. Sun, J. Han, X. Yan, P.S. Yu, T. Wu, Pathsim: meta path-based top-k similarity search in heterogeneous information networks, in: PVLDB, 4, 2011, pp. 992–1003.
[23] I.S. Dhillon, Co-clustering documents and words using bipartite spectral graph partitioning, in: 7th International Conference on Knowledge Discovery and Data Mining, 2001, pp. 269–274.
[24] I.S. Dhillon, S. Mallela, D.S. Modha, Information-theoretic co-clustering, in: 9th International Conference on Knowledge Discovery and Data Mining, 2003, pp. 89–98.
[25] A.L.V. Pereira, E.R. Hruschka, Simultaneous co-clustering and learning to address the cold start problem in recommender systems, Knowl. Based Syst. 82 (2015) 11–19.

[26] V.N. Pham, L.T. Ngo, W. Pedrycz, Interval-valued fuzzy set approach to fuzzy co-clustering for data classification, Knowledge-Based Systems, 2016.
[27] G. Govaert, M. Nadif, Mutual information, phi-squared and model-based co–clustering for contingency tables, in: Advances in Data Analysis and Classification, 2016, pp. 1–34.
[28] M. Ailem, F. Role, M. Nadif, Graph modularity maximization as an effective method for co-clustering text data, Knowl. Based Syst. 109 (2016) 160–173.
[29] M. Steinbach, G. Karypis, V. Kumar, A comparison of document clustering techniques, KDD Workshop on Text Mining, 2000.
[30] M. Hosseini, H. Abolhassani, Hierarchical co-clustering for web queries and selected urls, in: Proceedings on Web Information Systems Engineering, 2007, pp. 653–C662.
[31] G. Xu, W.Y. Ma, Building implicit links from content for forum search, in: Proceedings of 29th Annual International ACM SIGIR Conference, New York, 2006, pp. 300–307.
[32] K.P. Yee, K. Swearingen, K. Li, M. Hearst, Faceted metadata for image search and browsing, in: Proceedings of CHI., 2003.
[33] C.D. Manning, P. Raghavan, H. Schütze, Introduction to Information Retrieval, Cambridge University Press, 2008.
[34] D. Lin, An information-theoretic definition of similarity, in: International Conference on Machine Learning, 1998, pp. 296–304.
[35] Z. Wu, M. Palmer, Verbs semantics and lexical selection, in: 32nd Annual Meeting on Association for Computational Linguistics, 1994, pp. 133–138.
[36] S. Busygin, O. Prokopyev, P.M. Pardalos, Biclustering in data mining, Comput. Oper. Res. 35 (9) (2008) 2964–2987.
[37] J.S. Wu, J.H. Lai, C.D. Wang, A novel co-clustering method with intra-similarities, in: 11th International Conference on Data Mining Workshops, 2011, pp. 300–306.
[38] B.K. Bao, W. Min, T. Li, C. Xu, Joint local and global consistency on interdocument and interword relationships for co-clustering, IEEE Trans. Cybern. 45 (1) (2015) 15–28.
[39] E. Giannakidou, V.A. Koutsonikola, A. Vakali, Y. Kompatsiaris, Co-clustering tags and social data sources, in: 9th International Conference on Web-Age Information Management, IEEE, 2008, pp. 317–324.
[40] R. Garcia, J.M. Gimeno, F. Perdrix, et al., Building a usable and accessible semantic web interaction platform, World Wide Web. 13 (1–2) (2010) 143–167.
[41] J. Taeho, L. Malrey, The evaluation measure of text clustering for the variable number of clusters, in: Advances in Neural Networks, 2007, pp. 871–879.