Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)

# HIEDS: A Generic and Efficient Approach to Hierarchical Dataset Summarization

**Gong Cheng***, **Cheng Jin, Yuzhong Qu**
National Key Laboratory for Novel Software Technology,
Nanjing University, Nanjing 210023, China

## Abstract

The rapid growth of open data on the Web promotes the development of data portals that facilitate finding useful datasets. To help users quickly inspect a dataset found in a portal, we propose to summarize its contents and generate a hierarchical grouping of entities connected by relations. Our generic approach, called HIEDS, considers coverage of dataset, height of hierarchy, cohesion within groups, overlap between groups, and homogeneity of groups, and integrates these configurable factors into a combinatorial optimization problem to solve. We present an efficient solution, to serve users with dynamically configured summaries with acceptable latency. We systematically experiment with our approach on real-world RDF datasets.

## 1 Introduction

Increasingly many datasets have been made available as open data on the Web. To cope with their explosive growth, open data portals are established, to collect datasets from the Web and provide users with a single point access. Users, when browsing a dataset found in a portal, are usually fed with its metadata but lack for an insight into its contents, thereby having difficulty in determining its relevance to the needs.

To solve it, methods have been proposed to automatically generate small-sized, high-level abstraction of data, to summarize the contents of a dataset for quick inspection. Given data modeled as entity-property-value triples such as Resource Description Framework (RDF), schema-based methods [Benedetti *et al.*, 2014] group entities by their classes (i.e., values of the Type property) and connect groups by relations (i.e., properties with entity values). Such methods, limited by their expressivity, may unfortunately generate identical summaries for datasets in the same domain that share an ontological schema. More expressive methods are partition-based [Tian *et al.*, 2008], which partition entities into disjoint groups based on all properties beyond Type. However, they may fail to obtain many natural but overlapping groups, e.g., actors and directors.

*Corresponding author: Gong Cheng (gcheng@nju.edu.cn).

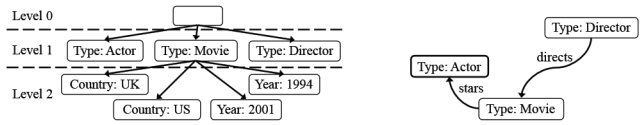
In this paper, a generic approach called HIEDS is proposed to summarize a dataset. Compared with existing schema-based methods that focus on the Type property, HIEDS exploits all the properties to divide entities into groups, and can also be configured to use a single property. Compared with partition-based methods that aim at disjoint groups, HIEDS allows groups to overlap to configurable degrees. More importantly, HIEDS organizes groups into a hierarchy, to provide abstraction at different granularities that can be incrementally explored. Our technical contribution is twofold.

- We propose to generate hierarchical dataset summaries. Our generic approach configures coverage of dataset, height of hierarchy, cohesion within groups, overlap between groups, and homogeneity of groups, all of which are formulated in a combinatorial optimization problem.
- We present an efficient solution to the problem. Our implementation allows approximate results depending on the computational resources available, and thus can serve users with dynamically configured summaries with acceptable latency even for large datasets.

## 2 Constitution of Summary

We expect the constitution of a dataset summary to exhibit three features. Firstly, a summary should *provide multi-granular abstraction of data* to be incrementally explored, since a single granularity could be either too coarse to offer adequate information or too fine to manage information overload. Secondly, a summary should *preserve the structural nature of a dataset*, i.e., to consist of not only its divisions but also connections between them. Thirdly, a summary should *be comprehensible*, i.e., to precisely and concisely characterize divisions and connections by human-readable labels.

To achieve these goals, we propose to iteratively divide the entities in a dataset into subgroups, forming a hierarchical grouping of entities to realize multiple granularities to be incrementally explored, as illustrated in Fig. 1(a). Each group is precisely characterized by a set of property-value pairs that are shared by all the entities in the group. Specifically, the root of the hierarchy (i.e., the only group at the 0-th level) contains all the entities in the dataset. Each group at the $l$-th level is characterized by a set of $l$ property-value pairs, $l-1$ of which are inherited from its ancestors in the hierarchy (i.e., groups on the path to the root of the hierarchy)




(a) A hierarchical grouping of entities.

(b) Relations connecting sibling groups.

Figure 1: An example dataset summary.

and are omitted from presentation; it is distinguished from its siblings by the remaining property-value pair as its label, and sibling groups are allowed to overlap (i.e., containing common entities) when some entity has all of these property-value pairs (e.g., a person who is both an actor and a director). The contents of a group are presented as a graph, in which its subgroups are connected by relations, as illustrated in Fig. 1(b); such a relation indicates that some entities in one subgroup have this relation to some entities in the other subgroup. In this way, both entity groups and the connections between them are labeled with concise terms that exactly come from the dataset, ensuring their comprehensibility.

# 3 Generation of High-quality Summary

Entities in a dataset can be grouped into different hierarchies; different relations can be selected from the dataset to connect groups. These lead to many and various summaries. In this section, we define a high-quality summary from five aspects, and obtain it by building a hierarchy of groups in a top-down manner via iterative subdivisions. Our solution, to be invoked iteratively, deals with a group of entities, and focuses on *which subgroups the input group should be divided into*, *connected by which relations*. We formulate a subdivision as a combinatorial optimization problem, and present an efficient solution to find the desired subgroups and relations.

## 3.1 Quality of Summary

To avoid overloading users with information, an input group should be divided into a limited number of subgroups, connected by selected relations. Their quality is fivefold.

**Coverage of Data.** The subgroups altogether should cover as many entities in the input group as possible, and thus each subgroup should cover a considerable portion. Analogously, selected relations should connect many pairs of entities.

**Height of Hierarchy.** Steadily dividing groups into large and overlapping subgroups to strive for coverage may generate a deep hierarchy of groups, leading to low efficiency of users' incremental exploration. To achieve a trade-off, subgroups should be not too large but moderate-sized.

**Cohesion within Groups.** The cohesion of a subgroup measures the extent to which the entities in it form a united whole. It is embodied in the common property-value pairs of the entities in the subgroup, which should not be generic ones shared by a broad range of entities in the dataset.

**Overlap between Groups.** Sibling subgroups are allowed to overlap, to help generating more natural subgroups. Nevertheless, their overlap should be controllable, to provide users with diverse directions in exploration.

**Homogeneity of Groups.** To avoid subdivision from different dimensions and reduce cognition burden, users may prefer homogeneous subgroups that are distinguished by different values of the same property. It should be configurable.

These five aspects are jointly formalized in the following.

## 3.2 Problem Formulation

Let $E$, $P$, and $V$ be the sets of all entities, properties, and values in a dataset, respectively. Let $R \subseteq P$ be the set of all relations, i.e., properties having entities as values. A dataset is a set of entity descriptions; the description of an entity $e \in E$ is a set of property-value pairs $D(e) \subseteq (P \times V)$.

When building a hierarchy of entity groups in a top-down manner, a group $G \subseteq E$ having been placed at the $l$-th level of the hierarchy is characterized by a set of $l$ property-value pairs $PV(G) \subseteq (P \times V)$, which are shared and are only shared by the entities in $G$:

$$G = \{e \in E : PV(G) \subseteq D(e)\}\,. \tag{1}$$

$E$ is the only group at the 0-th level, as the root of the hierarchy and characterized by an empty set of property-value pairs, i.e., $PV(E) = \emptyset$. When $l > 0$, among the $l$ property-value pairs in $PV(G)$, $l-1$ are inherited from $G$'s ancestors in the hierarchy, and the remaining one that distinguishes $G$ from its siblings is called $G$'s *label*, denoted by $lbl(G) \in PV(G)$. As illustrated in Fig. 1(a), the group labeled with Country:US is characterized by {Type:Movie, Country:US}.

In a top-down building process, our solution takes a group $G \subseteq E$ as input, and aims to divide it into not more than $k$ subgroups and obtain a high-quality summary as discussed in Sect. 3.1. Since groups are characterized by property-value pairs, *generating $k$ subgroups of $G$ amounts to selecting $k$ property-value pairs (other than $PV(G)$) from descriptions of the entities in $G$, each as the label of one subgroup*.

We formulate it as a combinatorial optimization problem. Specifically, $k$ property-value pairs will be selected from

$$CPV(G) = \bigcup_{e \in G} D(e) \setminus PV(G)\,. \tag{2}$$

Prior to selection, from these candidates, those corresponding to small subgroups containing smaller than a threshold proportion of entities are filtered out, to ensure considerable coverage of entities; the remaining candidates are

$$CPV'(G) = \{pv \in CPV(G) : \tau_E \leq \frac{|lbl^-(pv)|}{|G|} < 1\}\,, \tag{3}$$

where $\tau_E \in [0, 1)$ is a parameter to be tuned, and $lbl^-(pv)$ returns the candidate subgroup labeled with $pv$:

$$\begin{aligned} lbl^-(pv) &= \{e \in E : (PV(G) \cup \{pv\}) \subseteq D(e)\} \\ &= \{e \in G : pv \in D(e)\}\,. \end{aligned} \tag{4}$$

Given $n = |CPV'(G)|$, we number the property-value pairs in $CPV'(G)$ from $pv_1$ to $pv_n$, and introduce a series of binary variables $x_i$ for $i = 1..n$ to indicate whether $pv_i$ is selected into a solution to the following combinatorial optimization problem:

$$\begin{aligned}
&\text{maximize } \sum_{i=1}^{n} x_i \cdot (\alpha \cdot modsize(pv_i) + \beta \cdot coh(pv_i)) \\
&\text{subject to } ① \; x_i \in \{0,1\}, \text{ for } i = 1..n, \\
&\qquad ② \; \sum_{i=1}^{n} x_i \leq k, \text{ and} \\
&\qquad ③ \; x_i + x_j \leq 1, \text{ for } i,j = 1..n \text{ subject to} \\
&\qquad\qquad i < j \text{ and } ovlp(pv_i, pv_j) \geq \delta \,.
\end{aligned} \tag{5}$$

In the objective function, $modsize(pv_i) \in [0,1]$ is larger when the proportion of entities in $G$ contained in $lbl^-(pv_i)$ is closer to $\frac{1}{k}$, to achieve a trade-off between coverage of entities (which favors large subgroups) and height of hierarchical grouping (which favors small subgroups):

$$modsize(pv_i) = \exp(-\frac{|\frac{|lbl^-(pv_i)|}{|G|} - \frac{1}{k}|}{\min(\frac{|lbl^-(pv_i)|}{|G|}, \frac{1}{k})}) \,, \tag{6}$$

and $coh(pv_i) \in [0,1]$ is larger when $pv_i$ is shared by fewer entities in $E$, indicating a higher cohesion of $lbl^-(pv_i)$:

$$\begin{aligned}
coh(pv_i) &= \frac{-\log \frac{freq(pv_i)}{|E|}}{\log |E|}, \text{ where} \\
freq(pv) &= |\{e \in E : pv \in D(e)\}| \,.
\end{aligned} \tag{7}$$

According to information theory, $coh(pv_i)$ actually measures the normalized information content associated with the probabilistic event that $pv_i$ describes an entity in $E$. Since one single solution can rarely simultaneously maximize $modsize$ and $coh$, a linear scalarization quantifying a trade-off between them is maximized, with weights $\alpha, \beta \in [0,1]$ to be tuned.

In the constraints, ① and ② require that a solution selects at most $k$ property-value pairs; ③ requires that any two generated subgroups $lbl^-(pv_i)$ and $lbl^-(pv_j)$ cannot have an overlap larger than a threshold $\delta \in [-1,1]$ to be tuned:

$$ovlp(pv_i, pv_j) = \frac{\log \frac{|lbl^-(pv_i) \cap lbl^-(pv_j)|/|G|}{(|lbl^-(pv_i)|/|G|) \cdot (|lbl^-(pv_j)|/|G|)}}{-\log(|lbl^-(pv_i) \cap lbl^-(pv_j)|/|G|)} \,, \tag{8}$$

which is in $[-1,1]$ and, according to information theory, actually measures the normalized pointwise mutual information (which is a kind of association) between the probabilistic event that $pv_i$ describes an entity in $G$ and the probabilistic event that $pv_j$ describes an entity in $G$.

If homogeneous subgroups labeled with the same property are required, we extend Eq. (5) by adding a new constraint:

$$④ \; x_i + x_j \leq 1, \text{ for } i,j = 1..n \text{ subject to } i < j \text{ and } p_i \neq p_j \,, \tag{9}$$

where $p_i, p_j \in P$ are the properties in $pv_i, pv_j$, respectively.

Last but not least, every ordered pair of generated subgroups $G_i$ and $G_j$ are connected by relations selected from those connecting entities in $G_i$ to entities in $G_j$:

$$\begin{aligned}
CR(G_i, G_j) = &\{r \in R : \exists e \in G_i, \exists e' \in G_j, \\
&(\langle r, e' \rangle \in D(e))\} \,.
\end{aligned} \tag{10}$$

A relation $r \in CR(G_i, G_j)$ will be selected if it connects more than a threshold number of entity pairs from $G_i$ to $G_j$:

$$\frac{|\{\langle e, r, e' \rangle : e \in G_i, e' \in G_j, \langle r, e' \rangle \in D(e)\}|}{\min(|G_i|, |G_j|)} \geq \tau_R \,, \tag{11}$$

where $\tau_R \in [0,1]$ is a parameter to be tuned.

### 3.3 Problem Solution

The combinatorial optimization problem in Eq. (5) (and (9)) can be seen as a multidimensional knapsack problem (MKP), which is NP-hard [Kellerer *et al.*, 2004]. Specifically, each candidate property-value pair $pv_i$ is treated as an item in MKP, and $\alpha \cdot modsize(pv_i) + \beta \cdot coh(pv_i)$ is seen as its profit; ② and ③ (and ④, when necessary) form multiple constraints in which the weight of an item is 1 in ②, and is 1 (presence) or 0 (absence) in ③ (and ④). We solve it using a greedy strategy, considering candidate property-value pairs one after the other and selecting one into the solution if that would not violate any constraint. Candidates are sorted in decreasing order of the following heuristic [Kellerer *et al.*, 2004]:

$$\frac{\alpha \cdot modsize(pv_i) + \beta \cdot coh(pv_i)}{cons(pv_i)} \,, \tag{12}$$

where $cons(pv_i)$ is the number of constraints where $pv_i$ (i.e., $x_i$) presents, i.e., with a weight of 1.

Even with a greedy strategy, an efficient implementation is non-trivial because some operations above are computationally expensive over a large dataset. The following implementation manipulates inverted indexes, and allows approximate results depending on the computational resources available.

**Finding Candidates.** Considering efficiency, our implementation refers to a group via the set of property-value pairs characterizing it, not explicitly storing the entities in it but retrieving them when needed. So given $G$, or more precisely $PV(G)$, $CPV'(G)$ is obtained using Algorithm 1, which retrieves the entities in $G$ based on $PV(G)$ (line 2) and then collects and filters their property-value pairs (line 3–10).

Specifically, to retrieve the entities in $G$, an inverted index $idx_E$ is created based on Lucene (lucene.apache.org), which is usually used in information retrieval to index terms in documents. In $idx_E$, each entity $e \in E$ and each property-value pair in $D(e)$ are indexed as a document and a term in the document, respectively. Then, the entities in $G$ are obtained by submitting to $idx_E$ a conjunctive query consisting of the property-value pairs in $PV(G)$ as clauses (line 2).

The entities in $lbl^-(pv)$ are obtained in a similar way (line 6). Its size is checked according to Eq. (3) (line 7). We notice that $|lbl^-(pv)| \leq freq(pv)$, inspiring us to precompute $freq$ for each property-value pair in the dataset; then, the precomputed $freq(pv)$ is checked (line 5) before line 6–7, since that is very fast and may save a query against $idx_E$.

We limit the size of $CPV'(G)$ to a maximum of $\mu_E$ (line 9–10), depending on the computational resources available, to ensure that the resulting MKP will be solved in a reasonable time. To avoid missing subgroups containing important entities due to the limit, the entities retrieved in line 2 are sorted in order of importance, which is measured by PageRank in a directed entity-relation graph derived from the

**Algorithm 1:** Computing $CPV'(G)$

**Data:** $PV(G)$.
**Result:** $CPV'(G)$.

```
1  CPV'(G) ← ∅;
2  G ← Query idx_E with a conjunction of PV(G);
3  for the i-th entity e_i ∈ G do
4      foreach pv ∈ D(e_i) do
5          if freq(pv)/|G| ≥ τ_E then
6              lbl⁻(pv) ← Query idx_E with a conjunction
                 of PV(G) ∪ {pv};
7              if τ_E ≤ |lbl⁻(pv)|/|G| < 1 then
8                  CPV'(G) ← CPV'(G) ∪ {pv};
9                  if |CPV'(G)| = μ_E then
10                     Break the outermost loop;
11 return CPV'(G);
```

**Algorithm 2:** Finding Relations that Satisfy Eq. (11)

**Data:** $PV(G_i)$ and $PV(G_j)$.
**Result:** Subset of $CR(G_i, G_j)$ that satisfy Eq. (11).

```
1  R_ij ← ∅;
2  T ← Query idx_R with a conjunction of PV(G_i) for F_S
     and a conjunction of PV(G_j) for F_T;
3  for the m-th result in T do
4      r_m ← F_R of the m-th result;
5      T_m ← Query idx_R with a conjunction of PV(G_i)
         for F_S, PV(G_j) for F_T, and r_m for F_R;
6      if |T_m| / min(|G_i|,|G_j|) ≥ τ_R then
7          R_ij ← R_ij ∪ {r_m};
8      if m = μ_R then
9          Break the loop;
10 return R_ij;
```

dataset, where vertices are the entities in $E$ and an arc connects $e_i \in E$ to $e_j \in E$ if $e_j$ is a property value of $e_i$, i.e., $e_i$ has a relation to $e_j$. PageRank scores are precomputed and added to $idx_E$, to be used by Lucene's sorting function.

**Establishing Constraints.** To establish the constraints, whereas ①②④ are trivial, ③ checks every pair of property-value pairs in $CPV'(G)$ and computes, for each pair, $ovlp(pv_i, pv_j)$ according to Eq. (8). That involves $lbl^-(pv_i)$, $lbl^-(pv_j)$, and $lbl^-(pv_i) \cap lbl^-(pv_j)$, which are obtained by querying $idx_E$ with a conjunction of $PV(G) \cup \{pv_i\}$, $PV(G) \cup \{pv_j\}$, and $PV(G) \cup \{pv_i, pv_j\}$, respectively.

**Computing Heuristic Function.** In Eq. (12), $cons(pv_i)$ is obtained by going over all the constraints; to compute $modsize(pv_i)$ according to Eq. (6), $lbl^-(pv_i)$ is obtained by querying $idx_E$ as described above; $coh(pv_i)$ is obtained according to Eq. (7) based on the precomputed $freq(pv_i)$.

**Selecting Relations.** Given two subgroups $G_i$ and $G_j$, or more precisely $PV(G_i)$ and $PV(G_j)$, relations connecting them are found using Algorithm 2, which retrieves candidate relations $CR(G_i, G_j)$ based on $PV(G_i)$ and $PV(G_j)$ (line 2–4) and then selects those satisfying Eq. (11) (line 5–9).

Specifically, an inverted index $idx_R$ is created based on Lucene, in which each entity-relation-entity triple $\langle e, r, e' \rangle \in (E \times R \times E)$ in the dataset subject to $\langle r, e' \rangle \in D(e)$ is indexed as a document with three fields: each property-value pair in $D(e)$ is indexed as a term in the field $F_S$, $r$ is indexed as a term in the field $F_R$, and each property-value pair in $D(e')$ is indexed as a term in the field $F_T$. Then, the relations in $CR(G_i, G_j)$ are obtained by submitting to $idx_R$ a conjunctive query consisting of the property-value pairs in $PV(G_i)$ for $F_S$ and those in $PV(G_j)$ for $F_T$ as clauses (line 2), and collecting $F_R$ of the retrieved results (line 3–4).

The set of entity-relation-entity triples in the numerator of Eq. (11) is obtained (as $T_m$) by querying $idx_R$ in a similar way (line 5). Its size is checked according to Eq. (11) (line 6–7), where $|G_i|$ and $|G_j|$ have been computed in Algorithm 1.

We limit the number of relations to a maximum of $\mu_R$ (line 8–9), depending on the computational resources available. To avoid missing relations connecting important entities due to the limit, the results retrieved in line 2 (which are entity-relation-entity triples) are sorted in order of the total PageRank score of the two entities. Scores are precomputed and added to $idx_R$, to be used by Lucene's sorting function.

# 4 Experiments

We have implemented an online prototype of our HIEDS approach.[1] In this section, we empirically study the quality of summaries for real-world RDF datasets generated by HIEDS under various configurations, compare it with a baseline method, and report its running time.

## 4.1 Datasets

Two extensively used RDF datasets were tested.

- SWDF[2] (Semantic Web Dog Food) offers 200K entity-property-value triples describing 20K entities in the research domain (e.g., papers, researchers).
- LinkedMDB[3] offers 6M entity-property-value triples describing 0.6M movie-related entities (e.g., actors).

## 4.2 Metrics

Given a hierarchical grouping of entities generated by HIEDS, three metrics were measured: weighted average coverage, average cohesion, and height.

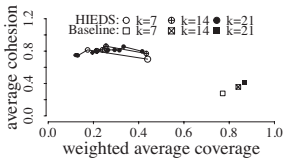
For each non-leaf group $G$ in a hierarchical grouping, the coverage of $G$ achieved by its subgroups $sg(G)$ is the proportion of $G$ covered by the union of $sg(G)$:

$$cov(G) = \frac{|\bigcup_{G' \in sg(G)} G'|}{|G|}. \tag{13}$$

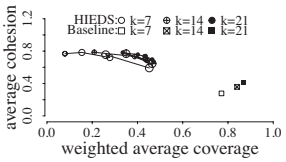
The *weighted average coverage* of a hierarchical grouping, in $[0, 1]$, is the average of coverage achieved over all the non-leaf groups ($NLG$) weighted by the logarithm of their sizes:

$$\text{weighted average coverage} = \frac{\sum_{G \in NLG} \log |G| \cdot cov(G)}{\sum_{G \in NLG} \log |G|}. \tag{14}$$

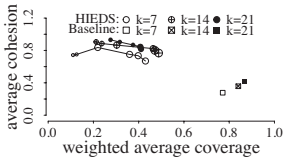
[1] http://ws.nju.edu.cn/hieds/
[2] http://data.semanticweb.org/
[3] http://data.linkedmdb.org/



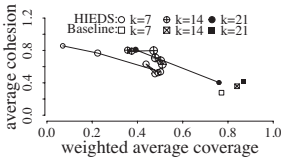

(a) SWDF (not s.t. ④).




(b) LinkedMDB (not s.t. ④).

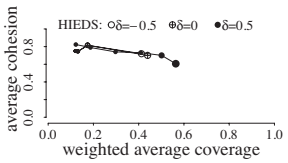



(c) SWDF (s.t. ④).

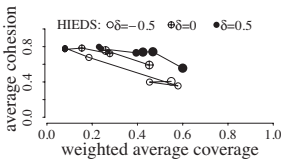



(d) LinkedMDB (s.t. ④).

Figure 2: Weighted average coverage (x-axis), average cohesion (y-axis), and height (diameter), under different $k, \alpha, \beta$.




(a) SWDF.




(b) LinkedMDB.

Figure 3: Weighted average coverage (x-axis), average cohesion (y-axis), and height (diameter), under different $\delta, \alpha, \beta$.

The *average cohesion* of a hierarchical grouping, in $[0, 1]$, is the average of cohesion of all the non-root groups ($NRG$):

$$\text{average cohesion} = \frac{\sum_{G \in NRG} coh(lbl(G))}{|NRG|}, \qquad (15)$$

which reuses $coh$ defined by Eq. (7); $lbl(G)$ is the property-value pair that labels $G$.

The *height* of a hierarchical grouping is the number of edges on a longest path between the root and a leaf.

## 4.3 Results and Discussion

In this experiment, groups were iteratively subdivided until each leaf group contained not more than 50 entities. We fixed $\mu_E = +\infty$ and $\tau_E = 0.01$ (c.f. Algorithm 1), to focus on the effects of other parameters.

**Effects of $\alpha$, $\beta$, $k$, and ④.** Firstly we tested the effects of $\alpha$ and $\beta$ (c.f. Eq. (5)), $k$ (i.e., the maximum number of subgroups of each non-leaf group), and whether to consider constraint ④ (c.f. Eq. (9)), under fixed $\delta = 0$ (c.f. Eq. (5)) for simplicity. Figure 2 shows the weighted average coverage, average cohesion, and height of hierarchical grouping generated under different settings of $k$, corresponding to different curves; the six points in each curve corresponded to different values of $\alpha$ from 0.0 to 1.0 in 0.2 increments and $\beta = 1 - \alpha$.

In Fig. 2(a) and 2(b) where constraint ④ was not introduced, when increasing $\alpha$ relative to $\beta$, weighted average coverage rose rapidly and average cohesion fell gradually, because a large value of $\alpha$ would favor large groups covering many entities whose cohesion was relatively low. However, weighted average coverage was below 0.5 under all the configurations on both datasets, being not very large, because except for Type, the two datasets lacked categorical properties that could induce groups of a considerable size; it would motivate us to seek novel ways of characterizing a group beyond property-value pairs in future work. When increasing $k$, both weighted average coverage and average cohesion were improved, because more small, cohesive groups were added to the summary; however, for those benefits, users had to spend more time reading many groups (and relations connecting them) at a time. Similar results were observed when constraint ④ was introduced to require homogeneous subgroups, as shown in Fig. 2(c) and 2(d).

Height was generally very small in the experiment because the size of group decreased exponentially with height, and thus would not bore users with too much interaction. On SWDF it was usually 2–4, and on LinkedMDB 3–5.

**Effects of $\delta$.** Then we tested the effects of $\delta$, under fixed $k = 7$ and not introducing constraint ④ for simplicity. Figure 3 shows the weighted average coverage, average cohesion, and height of hierarchical grouping generated under different settings of $\delta$, corresponding to different curves; the six points in each curve corresponded to different values of $\alpha$ from 0.0 to 1.0 in 0.2 increments and $\beta = 1 - \alpha$.

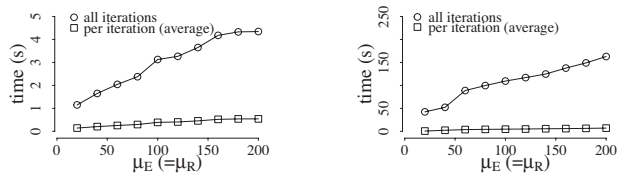
Under the same value of $\alpha$ and $\beta$, weighted average coverage was generally larger and average cohesion was slightly lower when $\delta$ was larger, because the allowed degree of overlap between groups was raised so that some large groups of relatively low cohesion were added to the summary.

Since larger values of $\delta$ led to more large groups, the height of hierarchical grouping rose accordingly. It hit a peak of 6 on LinkedMDB when $\alpha \geq 0.6$ and $\delta = 0.5$.

**Comparison with Baseline.** We compared HIEDS with LODeX [Benedetti *et al.*, 2014], a baseline schema-based method for summarizing a dataset. LODeX grouped entities by their classes, and the $k$ largest groups would form a summary; it could be conceived as a specific configuration of HIEDS that was biased towards coverage. The flat grouping it generated achieved higher coverage than HIEDS, as shown in Fig. 2. However, the cohesion of those groups was rather low, like Type:Person on LinkedMDB describing an overly broad concept. The results also included notably overlapping groups offering redundant information, like Type:Person and Type:Chair on SWDF; by comparison, HIEDS could be configured to control the degree of allowed overlap.

## 4.4 Running Time

We tested the running time of HIEDS on an Intel E3-1225 v3 with 30G memory for our Java program. All the relevant data resided in the memory, expect for Lucene indexes on disk. In HIEDS, groups were iteratively subdivided until each leaf group contained not more than 50 entities; the following parameters were fixed for simplicity: $\alpha = \beta = 0.5$, $\delta = 0$, $\tau_E = \tau_R = 0.01$, $k = 7$, and not introducing constraint ④.




(a) SWDF. (b) LinkedMDB.

Figure 4: Running time.

Note that the running time of HIEDS is not explicitly related to $\alpha$, $\beta$, $\delta$, or $\tau_R$, though it may be affected by $\tau_E$ and $k$.

Figure 4 shows the running time under different values of $\mu_E$ and $\mu_R$ (c.f. Algorithm 1 and 2, respectively) from 20 to 200 in 20 increments. The running time generally increased linearly when increasing $\mu_E$ and $\mu_R$; in practice, they could be tuned depending on the computational resources available to an application, to achieve a trade-off between quality of summary and running time. When $\mu_E = \mu_R = 200$, HIEDS took only 4s to compute all the iterations and generate a hierarchical summary for SWDF, but took 163s for LinkedMDB, mostly used to establish constraint ③ which queried the inverted index $idx_E$ for every pair of candidate property-value pairs in $CPV'$.

Actually, since a hierarchical summary was to be incrementally explored, it might be not necessary to compute all the iterations; an iteration could be computed just prior to exploring it, and thus a more important metric would be the average running time per iteration. As shown in Fig. 4, when $\mu_E = \mu_R = 200$, the averages were 0.5s and 7s for SWDF and LinkedMDB, respectively; it would be even faster under smaller values of $\mu_E$ and $\mu_R$, or by parallelizing our implementation, thereby being capable of serving users with dynamically configured summaries with acceptable latency.

## 5 Related Work

**Dataset Summarization.** To summarize a dataset for quick inspection, schema-based methods group entities by their classes and connect groups by relations, to form a conceptual graph (or a set of tables [Yan *et al.*, 2016]) as a high-level abstraction of data. Such a small-sized graph focuses on classes and relations either frequently used [Basse *et al.*, 2010; Böhm *et al.*, 2012] or centrally located [Presutti *et al.*, 2011; Christodoulou *et al.*, 2013] in the data, and its visualization is often associated with statistical information such as frequency of classes and relations [Benedetti *et al.*, 2014; Palmonari *et al.*, 2015]. By comparison, partition-based methods exploit all the properties beyond Type to group entities. Partition is based on commonly related entities [Khatchadourian and Consens, 2010], common relations [Tian *et al.*, 2008], or a variety of features [Campinas *et al.*, 2013]. Whereas both schema- and partition-based methods generate flat summaries, our HIEDS approach organizes groups into a hierarchy providing multi-granular abstraction of data to be incrementally explored. As a generic approach, HIEDS can be configured to either generate homogeneous groups based on a single property like schema-based methods (which only use Type) or exploit all the properties like partition-based methods; HIEDS can also be configured to allow different degrees of overlap between groups, and thus can generate natural but overlapping groups whereas partition-based methods require disjoint groups.

Different from the above browsing-oriented methods, summaries have also been generated to help efficiently answering structured queries posed to a dataset [Rietveld *et al.*, 2014; Dolby *et al.*, 2007] or testing whether a query has any answers against a dataset [Čebirić *et al.*, 2015]. Compared with our work, these query-oriented methods pursue different goals, and their techniques are orthogonal to ours.

**Ontology Summarization.** The ontological schema of a dataset may also be very large; its summary will be used in ontology search engines for quick inspection. Existing methods adopt extractive strategies, to select a subset of salient classes and properties [Wu *et al.*, 2008; Peroni *et al.*, 2008] or their descriptions [Zhang *et al.*, 2007; Troullinou *et al.*, 2015] from an ontology. By comparison, we deal with not the schema but the contents of a dataset. Our HIEDS approach adopts non-extractive strategies, to not extract a subset but generate a high-level abstraction of data.

**Faceted Search.** Faceted search also provides incremental exploration of a set of entities. It often assumes that a user enters a query stating a precise information need, and thus selects facets biased towards the query [Roy *et al.*, 2008], whereas our HIEDS approach is query-independent and addresses browsing-related concerns. Compared with existing browsing-oriented methods [Wagner *et al.*, 2011], HIEDS is distinguished not only by its generic nature comprising five aspects and the efficient implementation thereof, but also by its consideration of relations connecting groups in a summary.

## 6 Conclusion

To summarize the contents of a dataset for quick inspection, our HIEDS approach integrates five configurable aspects into a combinatorial optimization problem to solve, and generates a hierarchical grouping of entities connected by relations to be incrementally explored. It finds applications in open data portals, and our efficient implementation is proven to serve users with dynamically configured summaries with acceptable latency. We plan to extend our online prototype and develop such a portal featuring dataset summaries generated by HIEDS, and conduct a user study based on that.

Our work can be improved in several directions. It would be useful to seek novel ways of characterizing a group beyond property-value pairs, and to find an index-free implementation that directly operates against a SPARQL endpoint. Besides, it would be interesting to explore how to summarize a dataset covering a wide range of topics like DBpedia, which has not been effectively resolved by existing methods.

## Acknowledgments

This work was supported in part by the 863 Program under Grant 2015AA015406, in part by the NSFC under Grant 61572247 and 61223003, and in part by the Fundamental Research Funds for the Central Universities.

## References


[Basse *et al.*, 2010] Adrien Basse, Fabien Gandon, Isabelle Mirbel, and Moussa Lo. DFS-based frequent graph pattern extraction to characterize the content of RDF triple stores. In *Proceedings of the 2010 Web Science Conference*, Raleigh, North Carolina, April 2010. Web Science Trust.

[Benedetti *et al.*, 2014] Fabio Benedetti, Laura Po, and Sonia Bergamaschi. A visual summary for linked open data sources. In *Proceedings of the ISWC 2014 Posters & Demonstrations Track*, pages 173–176, Riva del Garda, Italy, October 2014. CEUR-WS.org.

[Böhm *et al.*, 2012] Christoph Böhm, Gjergji Kasneci, and Felix Naumann. Latent topics in graph-structured data. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pages 2663–2666, Maui, Hawaii, October–November 2012. ACM.

[Campinas *et al.*, 2013] Stéphane Campinas, Renaud Delbru, and Giovanni Tummarello. Efficiency and precision trade-offs in graph summary algorithms. In *Proceedings of the 17th International Database Engineering & Applications Symposium*, pages 38–47, Barcelona, Spain, October 2013. ACM.

[Čebirić *et al.*, 2015] Šejla Čebirić, Franćois Goasdoué, and Ioana Manolescu. Query-oriented summarization of RDF graphs. *Proceedings of the VLDB Endowment*, 8(12):2012–2015, August 2015.

[Christodoulou *et al.*, 2013] Klitos Christodoulou, Norman W. Paton, and Alvaro A. A. Fernandes. Structure inference for linked data sources using clustering. In *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, pages 60–67, Genoa, Italy, March 2013. ACM.

[Dolby *et al.*, 2007] Julian Dolby, Achille Fokoue, Aditya Kalyanpur, Aaron Kershenbaum, Edith Schonberg, Kavitha Srinivas, and Li Ma. Scalable semantic retrieval through summarization and refinement. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence*, pages 299–304, Vancouver, Canada, July 2007. AAAI Press.

[Kellerer *et al.*, 2004] Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack Problems*. Springer-Verlag, Berlin Heidelberg, Germany, 2004.

[Khatchadourian and Consens, 2010] Shahan Khatchadourian and Mariano P. Consens. ExpLOD: Summary-based exploration of interlinking and RDF usage in the linked open data cloud. In *Proceedings of the 7th Extended Semantic Web Conference, Part II*, pages 272–287, Heraklion, Greece, May–June 2010. Springer.

[Palmonari *et al.*, 2015] Matteo Palmonari, Anisa Rula, Riccardo Porrini, Andrea Maurino, Blerina Spahiu, and Vincenzo Ferme. ABSTAT: Linked data summaries with ABstraction and STATistics. In *Proceedings of the ESWC 2015 Satellite Events*, pages 128–132, Portoroz, Slovenia, May–June 2015. Springer.

[Peroni *et al.*, 2008] Silvio Peroni, Enrico Motta, and Mathieu d'Aquin. Identifying key concepts in an ontology, through the integration of cognitive principles with statistical and topological measures. In *Proceedings of the 3rd Asian Semantic Web Conference*, pages 242–256, Bangkok, Thailand, December 2008. Springer.

[Presutti *et al.*, 2011] Valentina Presutti, Lora Aroyo, Alessandro Adamou, Balthasar Schopman, Aldo Gangemi, and Guus Schreiber. Extracting core knowledge from linked data. In *Proceedings of the 2nd International Workshop on Consuming Linked Data*, Bonn, Germany, October 2011. CEUR-WS.org.

[Rietveld *et al.*, 2014] Laurens Rietveld, Rinke Hoekstra, Stefan Schlobach, and Christophe Guéret. Structural properties as proxy for semantic relevance in RDF graph sampling. In *Proceedings of the 13th International Semantic Web Conference, Part II*, pages 81–96, Riva del Garda, Italy, October 2014. Springer.

[Roy *et al.*, 2008] Senjuti B. Roy, Haidong Wang, Gautam Das, Ullas Nambiar, and Mukesh Mohania. Minimum-effort driven dynamic faceted search in structured databases. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, pages 13–22, Napa Valley, California, October 2008. ACM.

[Tian *et al.*, 2008] Yuanyuan Tian, Richard A. Hankins, and Jignesh M. Patel. Efficient aggregation for graph summarization. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 567–580, Vancouver, Canada, June 2008. ACM.

[Troullinou *et al.*, 2015] Georgia Troullinou, Haridimos Kondylakis, Evangelia Daskalaki, and Dimitris Plexousakis. RDF digest: Efficient summarization of RDF/S KBs. In *Proceedings of the 12th European Semantic Web Conference*, pages 119–134, Portoroz, Slovenia, May–June 2015. Springer.

[Wagner *et al.*, 2011] Andreas Wagner, Günter Ladwig, and Thanh Tran. Browsing-oriented semantic faceted search. In *Proceedings of the 22nd International Conference on Database and Expert Systems Applications*, pages 303–319, Toulouse, France, August–September 2011. Springer.

[Wu *et al.*, 2008] Gang Wu, Juanzi Li, Ling Feng, and Kehong Wang. Identifying potentially important concepts and relations in an ontology. In *Proceedings of the 7th International Semantic Web Conference*, pages 33–49, Karlsruhe, Germany, October 2008. Springer.

[Yan *et al.*, 2016] Ning Yan, Sona Hasani, Abolfazl Asudeh, and Chengkai Li. Generating preview tables for entity graphs. In *Proceedings of the 2016 ACM SIGMOD International Conference on Management of Data*, San Francisco, California, June–July 2016. ACM.

[Zhang *et al.*, 2007] Xiang Zhang, Gong Cheng, and Yuzhong Qu. Ontology summarization based on RDF sentence graph. In *Proceedings of the 16th International Conference on World Wide Web*, pages 707–716, Banff, Canada, May 2007. ACM.