Effective and Efficient Retrieval of Structured Entities

Ruihong Huang
Tsinghua University
Beijing, China
hrh16@mails.tsinghua.edu.cn

Shaoxu Song
Tsinghua University
Beijing, China
sxsong@tsinghua.edu.cn

Yunsu Lee
Samsung Research
Seoul, South Korea
yunsu16.lee@samsung.com

Jungho Park
Samsung Research
Seoul, South Korea
j0106.park@samsung.com

Soo-Hyung Kim
Samsung Research
Seoul, South Korea
sooh0721.kim@samsung.com

Sungmin Yi
Samsung Research
Seoul, South Korea
sungmin.yi@samsung.com

ABSTRACT

Structured entities are commonly abstracted, such as from XML, RDF or hidden-web databases. Direct retrieval of various structured entities is highly demanded in data lakes, e.g., given a JSON object, to find the XML entities that denote the same real-world object. Existing approaches on evaluating structured entity similarity emphasize too much the structural inconsistency. Indeed, entities from heterogeneous sources could have very distinct structures, owing to various information representation conventions. We argue that the retrieval could be more tolerant to structural differences and focus more on the contents of the entities. In this paper, we first identify the unique challenge of parent-child (containment) relationships among structured entities, which unfortunately prevent the retrieval of proper entities (returning parents or children). To solve the problem, a novel hierarchy smooth function is proposed to combine the term scores in different nodes of a structured entity. Entities sharing the same structure, namely an entity family, are employed to learn the coefficient in aggregating the scores, and thus distinguish/prune the parent or child entities. Remarkably, the proposed method could cooperate with both the bag-of-words (BOW) and word embedding models, successful in retrieving unstructured documents, for querying structured entities. Extensive experiments on real datasets demonstrate that our proposal is effective and efficient.

PVLDB Reference Format:
Ruihong Huang, Shaoxu Song, Yunsu Lee, Jungho Park, Soo-Hyung Kim and Sungmin Yi. Effective and Efficient Retrieval of Structured Entities. *PVLDB*, 13(6): 826-839, 2020.
DOI: https://doi.org/10.14778/3380750.3380754

1. INTRODUCTION

A structured entity is a set of connected nodes, often in a tree structure, labeled with types and contents. For example, T_1 in Figure 1 denotes a structured entity consisting of 4 nodes with types such as Institute and contents,

Proceedings of the VLDB Endowment, Vol. 13, No. 6
ISSN 2150-8097.
DOI: https://doi.org/10.14778/3380750.3380754

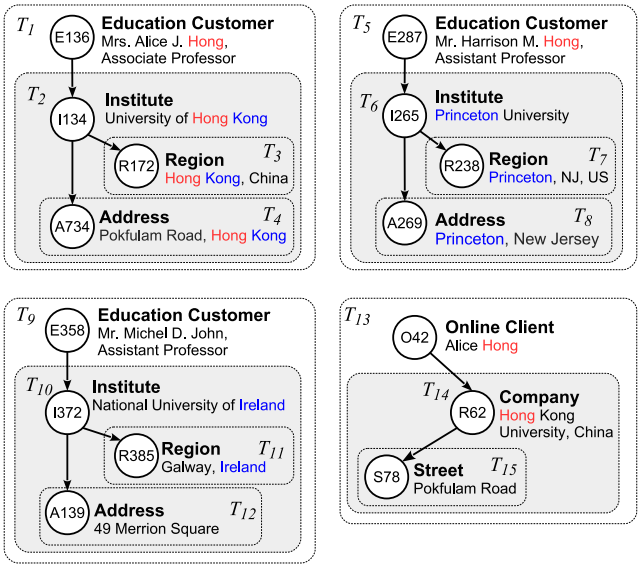

Figure 1: Motivation example of structured entities T in different customer channels in a corporation

e.g., University of Hong Kong. Such structured entities are prevalent in XML, RDF and hidden-web databases [18]. A structured entity in relational databases is a collection of tuples connected via primary-foreign keys, a.k.a. joined tuple tree [42]. XML entities are naturally organized in a tree structure [41]. In RDF data, entities could be interpreted as subgraphs [43]. In key-value stores, such as MongoDB, documents are encoded in JSON as semi-structured data.

In the emerging scenario of data lakes, direct retrieval on various source types becomes an important problem. Given a structured entity (query), the *structured entity retrieval* problem is to find the structured entities that are most relevant to the query, e.g., denoting the same real-world objects. For instance, given the structured query entity T_{13} in Figure 1, say from a relational database in the online channel, a result entity T_1 is expected (from some XML document in the education channel with a distinct structure but denoting the same person as T_{13}).

1.1 Challenges on Parent-Child Relationships

It is notable that the structured entity is **different** from the information object consisting of multiple documents with-

out any structure [36]. Owing to the hierarchical structure of entities, one structured entity may *contain* another, e.g., T_1 known as the parent contains T_2 as a child in Figure 1.

Some instances of shared terms are due to the unnormalized nature of the data, e.g., common in data lakes. While shared terms may appear in properly normalized data as well, we primarily target the former. For example, in Figure 1, Hong Kong appears not only in the Address of T_4 but also in the Region of T_3. However, such shared properties seem not entirely avoidable. For instance, the Institute of University of Hong Kong locates in the Region of Hong Kong.

It leads to the essential question: is a term more representative in the parent entity or the child entity? In other words, is the term able to distinguish whether the parent or child entity should be returned to a query entity?

Overlapping Contents. One may represent entities as term vectors, i.e., the bag-of-words (BOW) model [34], or embedding vectors, e.g., using word embeddings learned by the continuous bag-of-words (CBOW) model [28], which are successful in retrieving unstructured documents. A straightforward idea is to simply merge the contents of a structured entity as an unstructured document. Structured entities are then represented (and compared) by term scores, or by aggregating embedding vectors weighted by the term scores.

To apply the state-of-the-art retrieval technique FSDM [44] (considering term dependencies), all the structured entities (both parents and embedded children) need to be materialized as separate documents. It does not only waste storage cost, but also incur huge retrieval time cost.

Rather than enumerating all the structured entities, a natural idea is to *dynamically* aggregate the scores of nodes by **sum** as the score of a term in a structured entity (often considered in keyword search in graphs [26, 19]). Unfortunately, this simple aggregation leads to a dogmatic assertion that a term is always more representative in the parent, i.e., always returning the parent entity in retrieval while the child entity could never be retrieved (Proposition 1).

Structural Inconsistency. Existing studies [16, 32, 29] on evaluating the **graph (tree) edit distance** [31] may not be the best way of retrieving structured entities. As illustrated in Figure 1, entities T_1 and T_{13} from heterogeneous data sources have very distinct structures, but indeed denote the same person in real-world.

The structured entity retrieval problem is **different** from the keyword search problem over structured data [11], where the results are online assembled, and thus cannot utilize the entities sharing the same structure in answer ranking. As illustrated below, the predetermined structure of entities is essential in effectively and efficiently retrieving structured entities, i.e., properties could be learned from entities sharing the same structure, and enable pruning.

1.2 Opportunity on Same Structured Entities

Instead of simple sum, in this paper, we propose to devise a weighted aggregation of term scores among nodes in a structured entity. To determine a node's weight in a score combination, the idea is to explore the entities sharing the same structure, namely an *entity family* (which is not available in unstructured entity resolution). For instance, entities T_5 and T_9 have the same structure with T_1, i.e., the same node types Education Customer, Institute and so on, as well as the same connections among these types such as between Education Customer and Institute.

Intuitively, if the nodes of a type frequently share terms with their children, terms are likely to be representative in the parent entities, i.e., assigning a high weight of the node in combination. For instance, in Figure 1, Institute type nodes, I134, I265 and I372, share a large number of terms with their children, e.g., Hong and Kong appear both in I134 and R172, and Princeton in both I265 and R238. Thereby, term Hong is representative in T_2, and similarly, term Princeton is representative in T_6.

On the other hand, if a node type barely shares contents with its children, terms might not be representative in the parent entities, i.e., probably assigning a low weight of the node in combination. For instance, E136, E287 and E358 barely share contents with their children. Term Hong should not be representative in T_1 (having low term score).

Following the intuition, we observe a coefficient ρ_F (in Formula 7) on how large a type of nodes shares with their children, and utilize the coefficient to advise the score combination (in Formula 8). The parent and child entities are then distinguished by checking whether they share contents proportional to the coefficient ρ_F of the other same-structured entities.

1.3 Contributions

To the best of our knowledge, this is the first work on distinguishing the representative power of terms for structured entities with parent-child (containment) relationships. Remarkably, the proposed term score combination method could cooperate with both the bag-of-words (BOW) and word embedding models for retrieving structured entities. Our main contributions are summarized as follows.

We identify the infeasibility of applying the existing term score combination techniques to structured entities (in Section 4). While the sum combination always favors the parent entities (Proposition 1), the combination with normalization might never return the parent entities in certain cases (Proposition 2).

We propose a novel hierarchy smooth combination function for structured entities (in Section 5). The idea is to learn the coefficient ρ_F of aggregating scores in a structured entity from those entities sharing the same structure (following the aforesaid intuition). While the coefficient ρ_F is shared among all the same-structured entities in an entity family F, one may still want to tune individually the importance of the head node in an entity and distinguish parent from its children, by a smooth factor λ. The parameter λ in the score combination, is not only theoretically interesting (the sum combination is indeed a special case of the proposed combination with $\lambda = 0$ as shown in Proposition 6), but also can be practically determined by observing the difference on similarity scores of returned entities (Figure 10).

We develop indexing and pruning techniques to improve the efficiency of retrieving the (top-k) entities with the highest similarity scores (in Section 6). The monotonicity of the proposed score function is first illustrated (Proposition 7). It enables strategies for efficient ranking and pruning.

We conduct an extensive experiment on real data sets to demonstrate both effectiveness and efficiency of the proposed techniques (in Section 7).

Table 1: Notations

Symbol	Description
T	entity
F	entity family
t	node
t_0	head node of entity T
w	term
$\mathrm{s}(w,t)$	score of a term w in node t
$\mathrm{s}(w,T)$	score of a term w in entity T (Formula 1)
ρ_F	ratio of expected scores, indicating how most entities in an entity family F share terms in head node and children (Formula 7)
λ	smooth factor, indicating importance of head node in an entity and distinguishing (parent) entity from its children

2. RELATED WORK

To retrieve structured entities, e.g., from databases, a straightforward approach is to treat the structured entity as a "big joined tuple", then perform schema matching and entity matching [20]. Since the structural information is ignored, the performance of retrieval is limited. We list below several attempts of utilizing structural information that may be adapted to structured entity retrieval.

2.1 Difference to Entity Matching

Entity resolution is considered over entities that are connected as graphs, e.g., by using references among entities [6]. Structural information from relationship graphs is leveraged in entity matching tasks in different knowledge bases [23] or Linked Open Data [7]. To retrieve event entities, the occurrence order relationships among events are considered [46]. Referential relationships among tuple entities in databases can also be utilized as authority information in ranking [5].

All the aforesaid entities (events, tuples) are still single nodes, while the links among entity nodes assist the retrieval. In contrast, the structured entity considered in this paper involves multiple connected nodes. It leads to the unique challenge that one (parent) entity may contain another (child). The structured entity retrieval needs to distinguish whether the parent or child entity should be returned. As illustrated in the experiments in Section 7, our proposal shows higher retrieval accuracy compared to the existing entity resolution by considering the links among entities.

2.2 Difference to Keyword Search

Keyword search over structured/semi-structured data [11] dynamically assembles answers referring to the queried keywords, e.g., joined tuples in databases [42] or across databases [35], subtrees w.r.t. lowest common ancestor in XML [10], or subgraphs in RDF data [24].

The main differences between structured entity retrieval and keyword search are as follows. (1) While structured entities are all clearly defined w.r.t. node types and represented, the results of keyword search, e.g., joining network of tuples, are often dynamically assembled according to the query keywords. As shown in Section 5.1, the pre-identified structure enables entity score combination to utilize knowledge from other entities sharing the same structure. (2) Without exploring the same structured entities, the sum combination of tuple scores (or with size normalization) is often adopted in keyword search [27, 19, 26]. As discussed in Section 4, sum (with normalization) may fail in retrieving structured entities with parent-child containment relationships.

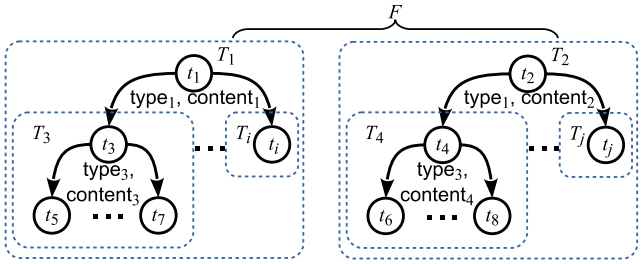

Figure 2: Structured entities with parent-children

2.3 Difference to Vectorizing Taxonomies

Motivated by the keyword search studies in databases, such as ObjectRank [5], the structural information paired with content information has been studied for vectorizing concepts in taxonomies [22, 12]. The CP/CV method [22] assigns different weights to keywords, depending on the structural information embedded in the taxonomy tree/graph. In addition to the structured entity retrieval studied in this paper, taxonomy vectorization is also used in other applications, e.g., to make a corpus-aware adaptation of existing taxonomies according to the change of contents in a document collection [8] or to build online reverse dictionary [38].

The novelty of our proposal, compared to the graph/tree vectorization [22, 12], lies in two aspects. (1) When the (weighted) concepts propagate between adjacent nodes and contribute to their characterization, "since the parent concept subsumes all child concepts, the propagation degrees from children to parent is set to 1.0" [22]. That is, the sum of child weights is considered, which may prefer the parent entities and fail in retrieving child entities, as analyzed in Section 4. Instead of a simple sum, in this paper, we devise a novel hierarchy smooth combination function with a coefficient ρ_F, denoting the expected ratio of scores between parents and children in aggregation. Remarkably, such a coefficient ρ_F could be different for various structured entities. (2) It leads to the second novelty. In order to determine a proper coefficient ρ_F for aggregating scores, we investigate the entities sharing the same structure, known as an entity family, which is not considered in vectorizing taxonomy trees/graphs. Such an entity family with the same structure is not only important in learning the parameters ρ_F for effective score combination (in Section 5) but also useful in pruning entities for efficient entity retrieval (in Section 6).

3. STRUCTURED ENTITY RETRIEVAL

In this section, we (1) introduce the sources of structured entities, (2) discuss the term score of structured entities, and (3) present the retrieval of structured entities.

3.1 Structured Entities

Definition 1. *An* entity T *is a directed acyclic graph* (N, E) *with a single root (having no parents), called* head node t_0

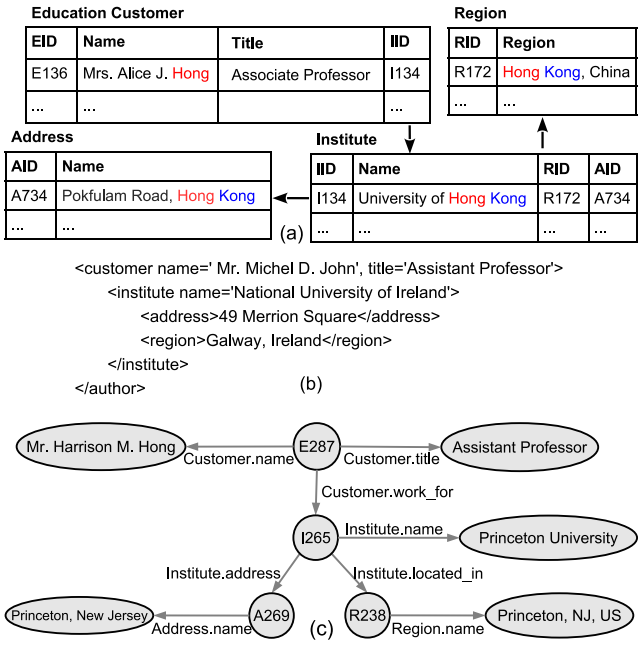

Figure 3: Structured entities in (a) database, (b) XML, (c) RDF

of T. Each node $t \in N$ is labeled with a (type, content) pair, and each edge $v_1 \to v_2 \in E$ denotes that v_1 is a parent of v_2.

Figure 2 presents a typical structured entity T_1 with head node t_1. Although our proposed technique could support structures with multiple roots in a directed acyclic graph (DAG), we do not find examples of such structures in real scenarios. For the same reason, we assume that nodes in an entity do not share the same type.

We call T_i a child entity of T_1, if the head node t_i of T_i is a child of t_1 (the head node of T_1), having an edge $t_1 \to t_i$. For instance, T_3 is also a child entity of T_1 in Figure 2.

Example 1. Figure 3 illustrates possible sources of structured entities in Figure 1, including (a) relational databases, (b) XML and (c) RDF. In Figure 3(a), each tuple corresponds to a node in the structured entity T_1 in Figure 1. The relation name serves as the type of a node, while the content of a tuple corresponds to the node's content. The connection between nodes, e.g., E136→I134, denotes the primary-foreign key relationship. The XML document in Figure 3(b) could be naturally represented in a tree structure, which corresponds to entity T_9 in Figure 1. An element nested in the XML document, e.g., institute, appears as a child entity T_{10} of T_9. RDF triples, represented as a graph in Figure 3(c), are transformed to more concise structured entities, i.e., T_5 in Figure 1. For instance, Customer.name is interpreted as the content of a node, while Customer.work_for denotes connections. Such entities can be obtained by template queries [43] or clustering [45] over a big RDF graph [47]. □

Analogous to tuples in a relation having the same set of attributes, we consider entities sharing the same structure.

Definition 2. *An* entity family *F is a set of entities sharing the same structure w.r.t. node types, i.e., for any two entities $T_1(N_1, E_1), T_2(N_2, E_2) \in F$, there is a bijective mapping between nodes N_1 and N_2 that preserves edges and node types.*

In other words, we have $|N_1| = |N_2|$, $|E_1| = |E_2|$ and for each $v_1 \to u_1 \in E_1$ there is a $v_2 \to u_2 \in E_2$ such that v_1, v_2 sharing the same type and u_1, u_2 sharing the same type. It also implies that for each $v_2 \to u_2 \in E_2$ there is a $v_1 \to u_1 \in E_1$ such that v_1, v_2 sharing the same type and u_1, u_2 sharing the same type, i.e., exactly identical structure for all entities in the same entity family F.

For example, entities T_1 and T_2 belonging to entity family F in Figure 2 share the same structure on node types. That is, nodes t_1 and t_2 have the same type_1, while their children t_3 and t_4 share the same type_3.

In this sense, entity family F could also be interpreted as the "schema" defining the structure of all entities in F. As presented in Section 5, the same structured entity family is essential in effectively retrieving structured entities. We learn properties from the entities sharing the same structure and enable pruning in Section 6.

3.2 Term Score

Following the bag-of-words model [34], we also consider the term scores of structured entities. Below, we start from the determination of term scores on nodes, the most elementary level in a structured entity, by treating them as unstructured documents. Then, the node scores of the same terms are combined as entity scores of an entity.

3.2.1 Node Score

By interpreting the content of a node as an unstructured document, we represent each node by a vector of terms. The score of term w in node t, denoted by $\mathrm{s}(w, t)$, could be evaluated by any IR score function in document retrieval such as TF-IDF [40]. Since it is not the focus of this paper to study the score evaluation on the unstructured document level, please refer to [39] for more details of the score function.

3.2.2 Entity Score

Once the scores of nodes are obtained, we combine them together as the scores of entities.

Definition 3. *The score of term w in entity $T(N, E)$ combines the scores of all the nodes $t \in N$ appearing in T,*

$$\mathrm{s}(w, T) = \mathsf{comb}_{t \in N}\big(\mathrm{s}(w, t)\big). \tag{1}$$

Designing the score combination function $\mathsf{comb}(\cdot)$, however, is non-trivial. It has to consider the unique entity containment relationship, i.e., a parent entity T may contain another child entity T_i. The term scores should be able to distinguish such entities that one contains another. We discuss this main challenge of score combination in structured entities in the following Section 4.

3.3 Entity Similarity

The *structured entity retrieval* problem is thus: given an entity T_q, to find the entities T, whose similarity scores $\mathrm{sim}(T_q, T)$ as defined below are the highest.

3.3.1 Term Vector Similarity

Given two entities T_1 and T_2 with the corresponding term scores, their similarity is given by

$$\mathrm{sim}(T_1, T_2) = \sum_{w \in W} \mathrm{s}(w, T_1)\,\mathrm{s}(w, T_2), \tag{2}$$

where W is the space of all terms. It is usually interpreted as the cosine of the angle between vectors [13].

3.3.2 Embedding Vector Similarity

Similar to sentence embedding [25], we can aggregate the word embeddings v_w weighted by term scores $\mathrm{s}(w, T)$ to form the structured entity embedding v_T,

$$v_T = \frac{1}{|T|} \sum_{w \in W} \mathrm{s}(w, T) v_w \tag{3}$$

where $|T|$ is the number of terms in entity T, W is the space of all terms, and v_w is the word embedding of term w, e.g., GloVe [33]. Given the entity embeddings of two entities T_1 and T_2, their similarity can be computed by

$$\mathrm{sim}(T_1, T_2) = \frac{v_{T_1} \cdot v_{T_2}}{||v_{T_1}||\ ||v_{T_2}||},$$

where $||v_{T_1}||, ||v_{T_2}||$ are the lengths of the vectors.

Overview

In the remaining of this paper, we will first illustrate how the existing score combination techniques fail in retrieving structured entities in Section 4. Our novel hierarchy smooth combination function is then presented in Section 5. Efficient indexing and pruning techniques are further developed upon the monotonicity property of the proposed score combination function in Section 6.

4. SIMPLE COMBINATIONS

In this section, we show that existing score combination strategies may fail to distinguish the entities with parent-child relationships. That is, the score of a term in the parent entity, for instance, is always higher than that of the child entity, and thus the child entity could never be retrieved.

4.1 Score Combination by Sum

To combine the scores of term w in an entity $T(N, E)$, a straightforward strategy is to sum up the scores of all nodes $t \in N$ appearing in T,

$$\mathrm{s}(w, T) = \mathsf{combSUM}_{t \in N}\big(\mathrm{s}(w, t)\big) = \sum_{t \in N} \mathrm{s}(w, t), \tag{4}$$

which is known as combSUM combination, *the sum of scores* [37]. Unfortunately, this combination does not work in representing entities with parent-child relationships. A direct result is that entity T as the parent always has a higher score than any child entity T_i.

Proposition 1. *A term w is always more representative in a parent entity T than any of its children T_i, i.e., $\mathrm{s}(w, T) \geq \mathrm{s}(w, T_i)$, evaluated by* combSUM *in Formula 4.*

In entity retrieval, referring to the similarity function in Formula 2, only the parent entities with higher scores will be returned as the results. That is, *the child entities can never be retrieved*, which is obviously a defeated attempt.

Example 2. Consider node R172 in T_3 in Figure 1. For simplicity, we assume that each term appearing in the content in a node has score 1, e.g., s(Hong, R172)=1 and s(Kong, R172)=1. Similar scores on terms Alice, University, and so on could also be obtained for nodes A734, I134 and E136.

To compute the entity score of T_2, we combine the scores of nodes R172, A734 and I134. The sum combination in Formula 4 is employed, having s(Hong, T_2)=3, s(Kong, T_2)=3, s(University, T_2)=1, and so on. Similarly, for T_1, we have s(Hong, T_1)=4, s(Kong, T_1)=3, s(University, T_1)=1, ... Referring to Proposition 1, for any term, the score in the parent is always higher (more representative) than that of child entity, e.g., s(Hong, T_1)=4 > s(Hong, T_2)=3.

Consider a query entity T_{14} whose term scores are computed similarly, having s(Hong, T_{14})=1, s(Kong, T_{14})=1, s(University, T_{14})=1, and so on. Terms not appearing in the entity have score 0. Referring to Formula 2, we have

$$\begin{aligned}&\mathrm{sim}(T_{14}, T_2) = 3 + 3 + 1 + 1 + 1 + 1 = 10\\ <\ &\mathrm{sim}(T_{14}, T_1) = 4 + 3 + 1 + 1 + 1 + 1 = 11.\end{aligned}$$

That is, the child entity T_2, denoting the same real world entity as T_{14}, cannot be retrieved, since its parent entity T_1 gets a higher similarity score with T_{14} by combSUM. □

For the same reason, combMNZ combination, which multiplies the combSUM by the number of non-zero scores and shows good performance in meta search [15], does not work in our problem either. Although a variation is studied in [26] with the consideration of the maximum node score, it is still based on combSUM with the problem of favoring parents.

4.2 Score Combination by Size Normalization

Instead of simple sum, the combination with normalization over the size of T is computed as follows, known as the combANZ combination, the average of non-zero scores [37],

$$\mathrm{s}(w, T) = \mathsf{combANZ}_{t \in N}\big(\mathrm{s}(w, t)\big) = \frac{\sum_{t \in N} \mathrm{s}(w, t)}{|N|}, \tag{5}$$

where $|N|$ is the number of nodes in entity T.

Unfortunately, this combANZ function might not be able to effectively distinguish entities with parent-child relationships either, e.g., in Figure 2.

Proposition 2. *Let T_m be the child of T with the maximum score, i.e., $m = \arg\max_{i \in [1,n]} \mathrm{s}(w, T_i)$. If*

$$\mathrm{s}(w, t_0) < \mathrm{s}(w, T_m), \tag{6}$$

where t_0 is the head node of T, then parent entity T always has score $\mathrm{s}(w, T) < \mathrm{s}(w, T_m)$ by combANZ *in Formula 5.*

Given the condition in Formula 6, term w in the parent entity T can never be as representative as in the children. In entity retrieval, the child with larger $\mathrm{s}(w, T_m)$ will always rank higher than the parent with lower $\mathrm{s}(w, T)$, referring to the similarity function in Formula 2. In this case, *the parent entity can never be retrieved.*

Example 3. Consider again the entity T_2 in Figure 1. Similar to Example 2, we compute the entity scores by using combANZ in Formula 5, having s(Hong, T_2)=$\frac{3}{3}$=1, s(Kong, T_2)=$\frac{3}{3}$=1, s(University, T_2)=$\frac{1}{3}$=0.33, and so on. Similar combination applies to T_1, having s(Hong, T_1)=$\frac{4}{4}$=1, s(Kong, T_1)=$\frac{3}{4}$=0.75, s(University, T_1)=$\frac{1}{4}$=0.25, ... The same score s(Hong, T_2)=s(Hong, T_1)=1 means that the term Hong cannot distinguish the child T_2 from the parent T_1.

Consider another query entity T_{13}, s(Hong, T_{13})=$\frac{2}{3}$=0.66, s(Kong, T_{13})=$\frac{1}{3}$=0.33, s(University, T_{13})=$\frac{1}{3}$=0.33, ... Referring to the similarity function in Formula 2, we have

$$\begin{aligned}\mathrm{sim}(T_{13}, T_1) &= \tfrac{1}{3}\tfrac{1}{4} + \tfrac{2}{3}\tfrac{4}{4} + \tfrac{1}{3}\tfrac{3}{4} + \tfrac{1}{3}\tfrac{1}{4} + \tfrac{1}{3}\tfrac{1}{4} + \tfrac{1}{3}\tfrac{1}{4} + \tfrac{1}{3}\tfrac{1}{4} = 1.33\\ <\ \mathrm{sim}(T_{13}, T_2) &= \tfrac{1}{3}\tfrac{0}{3} + \tfrac{2}{3}\tfrac{3}{3} + \tfrac{1}{3}\tfrac{3}{3} + \tfrac{1}{3}\tfrac{1}{3} + \tfrac{1}{3}\tfrac{1}{3} + \tfrac{1}{3}\tfrac{1}{3} + \tfrac{1}{3}\tfrac{1}{3} = 1.44.\end{aligned}$$

That is, T_{13} and T_2 with higher similarity score by combANZ will be erroneously returned, whereas the truth is that T_{13} and T_1 indeed denote the same real world entity. □

5. STRUCTURE-AWARE COMBINATION

In order to distinguish the entities with parent-child relationships, we need a coefficient (ρ_F) to tell the difference of a term w's importance in the parent and child entities. Intuitively, such a coefficient could be learned from entities sharing the same structure (the same parent-child relationships w.r.t. node types). If most entities with the same structure share contents between head nodes and their children (e.g., between T_2 and T_3, T_6 and T_7, T_{10} and T_{11}, in Figure 1), the shared terms in children could contribute positively in retrieving the parents. A novel score combination function is thus introduced following the intuition, in Section 5.1. We illustrate different scenarios where the parent or the child could be ranked higher in retrieval, in Section 5.2.

5.1 Hierarchy Smooth Combination

Referring to the intuition at the beginning of this section, we learn a coefficient (ρ_F), evaluating the term score difference between an entity and its children, from all entities having the same structure.

Let ρ_F denote the expected coefficient of scores between the head node and all the n children of an entity in F

$$\rho_F = \mathbf{E}\left(\frac{\sum_{i=1}^{n} \mathrm{s}(w, T_i)}{\mathrm{s}(w, t_0)}\right) \quad (7)$$
$$= \frac{1}{|W_F| \cdot |F|} \sum_{w \in W_F} \sum_{T \in F} \frac{\sum_{i=1}^{n} \mathrm{s}(w, T_i)}{\mathrm{s}(w, t_0)}$$

where W_F is the set of all terms appearing in the head nodes of entities in F, t_0 is the head node of entity T, and T_i are children of T.

As discussed in the introduction, if a type of nodes frequently share terms with their children, i.e., large coefficient ρ_F, terms are likely to be representative in the parent entities, i.e., assigning a high weight of the node in combination. On the other hand, if a node type barely shares contents with its children, i.e., small coefficient ρ_F, terms might not be representative in the parent entities, i.e., probably assigning a low weight of the node in combination. We first introduce the combination function following this intuition, then explain its detailed properties in Section 5.2.

Definition 4. *The score of term w in entity T in an entity family F is given by* combHIS

$$\mathrm{s}(w, T) = (1 + \lambda \cdot \rho_F)\, \mathrm{s}(w, t_0) + (1 - \lambda) \sum_{i=1}^{n} \mathrm{s}(w, T_i) \quad (8)$$

where $\mathrm{s}(w, t_0)$ is the score of the head node t_0 of T, $\mathrm{s}(w, T_i)$ is the score of the child entity T_i of T, ρ_F is the aforesaid coefficient on term sharing, and λ is a smooth factor in the range of $[0, 1]$ on weighting the parent and child entities (see explanations below).

The combHIS$(\cdot)$ score combination function is defined recursively. For a family F of entities without any child, we have $\rho_F = 0$. It follows $\mathrm{s}(w, T) = \mathrm{s}(w, t_0)$. The value of ρ_F is determined by the contents of head nodes and children entities in F, and shared among all the entities in F.

In order to tune individually the importance of the head node in an entity and distinguishing parent from its children, we introduce a parameter of smooth factor λ in the hierarchy combination. The larger the smooth factor λ is, the more the entity $\mathrm{s}(w, T)$ relies on the head node $\mathrm{s}(w, t_0)$.

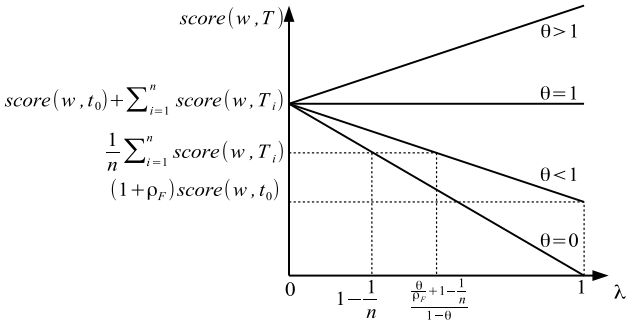

Figure 4: Hierarchy smooth combination

Example 4. Consider the entity family $F_2 = \{T_2, T_6, T_{10}\}$ sharing the same structure. There are 10 terms appearing in the head nodes of entities in F_2, each with node score 1. Four of them also appear in the child entities. For entities $T_3, T_4, T_7, \ldots$ without child, we have $\mathrm{s}(w, T) = \mathrm{s}(w, t_0)$.

Referring to Formula 7, the ratio is computed as $\rho_{F_2} = \frac{1}{10\times 3}(\frac{2}{1} + \frac{2}{1} + \frac{2}{1} + \frac{1}{1}) = \frac{7}{30}$. Suppose that the smooth factor λ is set to 0.9. According to Formula 8, the scores for entities in F_2 could be computed, e.g., $\mathrm{s}(\text{Hong}, T_2) = (1 + 0.9 \times \frac{7}{30})\,\mathrm{s}(\text{Hong}, I134) + (1 - 0.9)(\mathrm{s}(\text{Hong}, T_3) + \mathrm{s}(\text{Hong}, T_4)) = 1.21 + 0.1 * 2 = 1.41$.

For the parent entities, i.e., entity family F_1=$\{T_1, T_5, T_9\}$, we obtain $\rho_{F_1} = \frac{1.41}{54}$. Similarly, scores for entities in F_1 are computed, e.g., $\mathrm{s}(\text{Hong}, T_1) = 1.1645$. That is, the term Hong is found more representative in T_2 than T_1.

Finally, consider again the query entities T_{13} and T_{14}. For simplicity, we assume $\rho_{F_{13}} = \rho_{F_{14}} = 1$, where $F_{13} = \{T_{13}, \ldots\}$ and $F_{14} = \{T_{14}, \ldots\}$. The similarities of entities for retrieval are $\mathrm{sim}(T_{13}, T_1) = 4.43$, $\mathrm{sim}(T_{13}, T_2) = 3.47$, $\mathrm{sim}(T_{14}, T_1) = 2.73$ and $\mathrm{sim}(T_{14}, T_2) = 7.87$. That is, T_{13} successfully matches T_1 and T_{14} matches T_2, which cannot be identified in the previous Examples 2 and 3 using combSUM or combANZ. □

5.2 Properties of combHIS

To illustrate the properties of combHIS in (four) different cases below, we rewrite Formula 8 w.r.t. smooth factor λ,

$$\mathrm{s}(w, T) = \lambda \left(\rho_F \cdot \mathrm{s}(w, t_0) - \sum_{i=1}^{n} \mathrm{s}(w, T_i)\right) + \mathrm{s}(w, t_0) + \sum_{i=1}^{n} \mathrm{s}(w, T_i).$$

It can be interpreted as a linear function on the variable λ. In Figure 4, its y-intercept is $\mathrm{s}(w, t_0) + \sum_{i=1}^{n} \mathrm{s}(w, T_i)$, (which is equivalent to combSUM in Formula 4). Let

$$\theta = \frac{\rho_F \cdot \mathrm{s}(w, t_0)}{\sum_{i=1}^{n} \mathrm{s}(w, T_i)} \quad (9)$$

denote the difference between $\frac{\sum_{i=1}^{n} \mathrm{s}(w, T_i)}{\mathrm{s}(w, t_0)}$ and ρ_F. We consider the linear functions with various slopes $\rho_F \cdot \mathrm{s}(w, t_0) - \sum_{i=1}^{n} \mathrm{s}(w, T_i) = (\theta - 1) \sum_{i=1}^{n} \mathrm{s}(w, T_i)$, given different θ.

5.2.1 *Case 1: $\theta > 1$*

For a term w with $\theta > 1$, i.e., $\rho_F > \frac{\sum_{i=1}^{n} \mathrm{s}(w, T_i)}{\mathrm{s}(w, t_0)}$, the head node t_0 does not share as much as other same-structured

entities with children $\sum_{i=1}^{n} \mathrm{s}(w, T_i)$, compared to the expectation ρ_F. Term w is more representative in the parent.

Proposition 3. *If the coefficient of children* $\sum_{i=1}^{n} \mathrm{s}(w, T_i)$ *to head node* $\mathrm{s}(w, t_0)$ *is less than the coefficient* ρ_F*,*

$$\frac{\sum_{i=1}^{n} \mathrm{s}(w, T_i)}{\mathrm{s}(w, t_0)} < \rho_F,$$

i.e., $\theta > 1$*, it always has* $\mathrm{s}(w, T) > \mathrm{s}(w, T_i)$.

5.2.2 Case 2: $\theta = 1$

For a term w with $\theta = 1$, i.e., term w is as representative as most other terms in the entity family F, the sum combination applies.

Proposition 4. *If the coefficient of children* $\sum_{i=1}^{n} \mathrm{s}(w, T_i)$ *to head node* $\mathrm{s}(w, t_0)$ *is proportional to the coefficient* ρ_F*,*

$$\frac{\sum_{i=1}^{n} \mathrm{s}(w, T_i)}{\mathrm{s}(w, t_0)} = \rho_F,$$

i.e., θ=1*, the entity* T *has* $\mathrm{s}(w, T) = \mathrm{s}(w, t_0) + \sum_{i=1}^{n} \mathrm{s}(w, T_i)$.

In this case, the parent also has a higher score than the child, i.e., $\mathrm{s}(w, T) \geq \mathrm{s}(w, T_i)$.

5.2.3 Case 3: $\theta < 1$

For a term w with $\theta < 1$, i.e., the coefficient $\frac{\sum_{i=1}^{n} \mathrm{s}(w, T_i)}{\mathrm{s}(w, t_0)}$ is greater than the expected coefficient ρ_F, term w is not as representative as most other terms in the entity family F. To illustrate this, we rewrite the entity score in Formula 8 by using θ in Formula 9, $\mathrm{s}(w, T) = \left(\frac{\theta}{\rho_F} + \lambda \cdot \theta + 1 - \lambda\right) \sum_{i=1}^{n} \mathrm{s}(w, T_i)$.

Proposition 5. *If the coefficient of children* $\sum_{i=1}^{n} \mathrm{s}(w, T_i)$ *to head node* $\mathrm{s}(w, t_0)$ *is greater than* ρ_F*,*

$$\frac{\sum_{i=1}^{n} \mathrm{s}(w, T_i)}{\mathrm{s}(w, t_0)} > \rho_F,$$

i.e., $\theta < 1$*, for a smooth factor* λ *with* $\frac{\frac{\theta}{\rho_F} + 1 - \frac{1}{n}}{1 - \theta} < \lambda < 1$*, we have* $\mathrm{s}(w, T) < \frac{1}{n} \sum_{i=1}^{n} \mathrm{s}(w, T_i)$.

Some child entity T_i of T may have $\mathrm{s}(w, T) < \mathrm{s}(w, T_i)$.

Example 5 (Example 4 continued)**.** Consider term Hong in entity T_{13} in Figure 1. Suppose that $\mathrm{s}(\text{Hong, O42}) = \mathrm{s}(\text{Hong}, T_{14}) = 1$ and $\rho_{F_{13}} = 1$, where $F_{13} = \{T_{13,\ldots}\}$, i.e., the aforesaid Case 2 with $\theta = 1$. In Example 4, we have $\mathrm{s}(\text{Hong}, T_{13}) = 2 = \mathrm{s}(\text{Hong, O42}) + \mathrm{s}(\text{Hong}, T_{14})$, which verifies the conclusion in Proposition 4. It implies that the term Hong is more presentative in the parent entity T_{13} than the child T_{14} (similar to Case 1).

When computing $\mathrm{s}(\text{Hong}, T_1)$, we have $\theta = \frac{\frac{1.41}{54} \times 1}{1.41} = 0.02 < 1$, i.e., Case 3. Example 4 illustrates $\mathrm{s}(\text{Hong}, T_1) = 1.1645 < \mathrm{s}(\text{Hong}, T_2) = 1.41$. It verifies the results in Proposition 5. In other words, the parent entity score could be lower than the child. □

Summary

Thus far, we illustrate that the hierarchy smooth score combination is able to distinguish the representative ability of a term in entities with parent-child relationships. While combSUM$(\cdot)$ is indeed a special of combHIS$(\cdot)$, the experimental results in Section 7 also show that our combHIS$(\cdot)$ yields significant improvement compared to combANZ$(\cdot)$.

5.3 Analysis of Smooth Factor λ

While the coefficient ρ_F is learned and shared among all the entities in the entity family F, smooth factor λ tunes individually the importance of the head node in an entity and distinguishing parent from its children.

First, for a smooth factor $\lambda = 0$, the score function in Formula 8 has $\mathrm{s}(w, T) = \mathrm{s}(w, t_0) + \sum_{i=1}^{n} \mathrm{s}(w, T_i)$, which is exactly the combSUM combination in Formula 4.

Proposition 6. combSUM *is a special* combHIS *with* $\lambda = 0$.

When $\lambda = 0$, as illustrated in Figure 4, the parent $\mathrm{s}(w, T)$ is the same under different θ (denoting the difference between $\frac{\sum_{i=1}^{n} \mathrm{s}(w, T_i)}{\mathrm{s}(w, t_0)}$ and ρ_F). The score function cannot distinguish the different representative power of terms in parent and child entities. Parents will always be returned, as analyzed after Proposition 1 in Section 4.

With the increase of λ*, referring to Formula 8, the weight of the head node* $\mathrm{s}(w, t_0)$ *increases in the combination function.* Meanwhile, the impacts of children T_i decrease. An entity with large $\mathrm{s}(w, t_0)$ relative to $\sum_{i=1}^{n} \mathrm{s}(w, T_i)$, i.e., having $\frac{\sum_{i=1}^{n} \mathrm{s}(w, T_i)}{\mathrm{s}(w, t_0)}$ less than ρ_F, will be ranked significantly higher in retrieval. On the other hand, a small $\mathrm{s}(w, t_0)$ relative to $\sum_{i=1}^{n} \mathrm{s}(w, T_i)$, i.e., $\frac{\sum_{i=1}^{n} \mathrm{s}(w, T_i)}{\mathrm{s}(w, t_0)}$ greater than ρ_F, leads to much lower similarity in retrieval. That is, in Figure 4, with the increase of λ, the distinction of $\mathrm{s}(w, T)$ becomes more significant under various θ.

Finally, for a smooth factor $\lambda = 1$, referring to Formula 8, we have $\mathrm{s}(w, T) = (1 + \rho_F)\, \mathrm{s}(w, t_0)$ *which considers only the head node* $\mathrm{s}(w, t_0)$. When $\theta = 0$, as illustrated in Figure 4, we have $\mathrm{s}(w, T) = 0$. That is, the child entities will have higher scores, and thus the parent could never be retrieved. If λ is too large (greater than 1), the score of parent T could be less than 0 which is meaningless.

Summary

The parameter λ could take a value in the range of $[0, 1]$ to retrieve meaningful answers for particular entities. A small λ (close to 0) will rank parent entities higher in the results. The larger the λ is, the more the entity score relies on the head node. For $\lambda = 1$, the entity score considers only the head node and ignores all its children contributions. In practice, the smooth factor λ can be determined by observing the difference of similarity scores of returned entities (as illustrated in Figure 10 in the experiments).

6. EFFICIENT ENTITY RETRIEVAL

To efficiently retrieve entities with the highest $\mathrm{sim}(T_q, T)$, rather than comparing query T_q to all entities T, we explore the monotonicity property of the proposed score combination function (Proposition 7). It enables efficient pruning on entity families enlightened by the threshold algorithm [14].

6.1 The Beauty of Monotonicity in Ranking

The property of monotonicity is often employed in ranking [19]. It is easy to see that our hierarchy smooth combination in Formula 8 is also monotonic. With this property, entities in the same entity family could be efficiently ranked.

Proposition 7. *For any two entities* T, T' *in the same entity family* F*, if* $\mathrm{s}(w, t) \leq \mathrm{s}(w, t')$*, for each* w *appearing in* T_q*, and each node pair* t *and* t' *with the same* type *appearing in* T *and* T'*, then it always has* $\mathrm{sim}(T_q, T) \leq \mathrm{sim}(T_q, T')$.

6.1.1 Index

To support efficient ranking, we build an inverted index on terms over nodes. (Note that building index over entities by materializing their scores is not practical. The reason is that the entity score in Formula 8 is determined by the coefficient ρ_F which changes with the update of entities in F, and depends on the parameter of smooth factor λ.)

Let L_{li} denote the list of nodes t_i with type_i that contains term w_l, ranked by $\mathrm{s}(w_l, t_i)$. The inverted index is thus a collection of lists L_{li} on all terms w_l appearing in possible nodes t_i with type_i. We denote $L_{li}[d]$ the d-th node in the ranked list, with $\mathrm{s}(w, L_{li}[d])$. For instance, $L_{12}[2]$ in Figure 5 denotes the 2nd node I1 in the list, with $\mathrm{s}(w_1, \mathrm{I1}) = 0.85$.

6.1.2 Threshold for Ranking

To extend the threshold-based pruning for efficient ranking [14], we define below the threshold for entity scores in an entity family. Consider the d-th level of ranked lists for entity family F_i. Referring to the term score in Formula 8 in Definition 4, we recursively define the threshold.

Definition 5. *The threshold at level d of term w_l in an entity family F_i is given recursively by*

$$\mathrm{t}(w_l, F_i, d) = (1 + \lambda \cdot \rho_{F_i})\, \mathrm{s}(w_l, L_{li}[d]) + (1-\lambda) \sum_{j=1}^{n} \mathrm{t}(w_l, F_j, d), \tag{10}$$

where the head nodes of entities in F_i are with type_i, and the entities in F_j are child entities of those in F_i.

By aggregating the thresholds on all the terms referred by the given T_q, we obtain the threshold for similarity scores between the given T_q and the entities (after level d) in F_i,

$$\mathrm{t}(T_q, F_i, d) = \sum_{w \in W} \mathrm{s}(w, T_q) \cdot \mathrm{t}(w, F_i, d). \tag{11}$$

Referring to entity score in Formula 8 and similarity score in Formula 2, the threshold at level d has following property.

Proposition 8. *For a query entity T_q and any entity $T_i \in F_i$, if $\mathrm{s}(w_l, t_j) \le \mathrm{s}(w_l, L_{lj}[d])$ for each node t_j with type_j appearing in T_i and each term w_l appearing in T_q, it always has $\mathrm{sim}(T_q, T_i) \le \mathrm{t}(T_q, F_i, d)$.*

Example 6. Consider an entity T_q with $\mathrm{s}(w_1, T_q) = 1$ and $\mathrm{s}(w_2, T_q) = 1$. There are four entity families $F_1, \ldots, F_4$ with head node $\mathsf{type}_1, \ldots, \mathsf{type}_4$, respectively. Suppose $\rho_{F_i} = 1, i = 1, \ldots, 4$ and $\lambda = 0.5$. Let $d = 2$ be the second level in the inverted index shown in Figure 5. We first compute the score threshold according to Formula 10, $\mathrm{t}(w_1, F_4, 2) = (1+\frac{0.5}{1})*0.3 = 0.45$ and $\mathrm{t}(w_1, F_3, 2) = (1+\frac{0.5}{1})*0.78 = 1.17$. Referring to the structure in Figure 1 that entities in F_2 contain those of F_3 and F_4 as children, we have $\mathrm{t}(w_1, F_2, 2) = (1 + \frac{0.5}{1}) * 0.85 + (1 - 0.5) * (1.17 + 0.45) = 2.085$. It follows $\mathrm{t}(w_1, F_1, 2) = (1 + \frac{0.5}{1}) * 0.1 + (1 - 0.5) * 2.085 = 1.1925$. Similarly, we can get $\mathrm{t}(w_2, F_1, 2) = 0.652875$. According to Formula 11, the similarity threshold of entity family F_1 in the second level is computed by, $\mathrm{t}(T_q, F_1, 2) = 1 * 1.1925 + 1 * 0.652875 = 1.845375$.

Consider entity $T_{\mathrm{E2}} \in F_1$ with head node E2, we have $\mathrm{s}(w_l, t_j) \le \mathrm{s}(w_l, L_{lj}[2])$, for $l = 1, 2$ and $j = 1, \ldots, 4$. Referring to Proposition 8, T_{E2} and T_q has similarity $\mathrm{sim}(T_q, T) = 0.9375 < \mathrm{t}(T_q, F_1, 2)$. □

w_1				w_2				
L_{11}	L_{12}	L_{13}	L_{14}	L_{21}	L_{22}	L_{23}	L_{24}	
(E1,0.15)	(I4,0.9)	(R3,0.84)	(A734,0.5)	(E4,0.16)	(I1,0.8)	(R3,0.85)	(A9, 0.001)	1
(E287,0.1)	(I1,0.85)	(R4,0.78)	(A1,0.3)	(E3,0.12)	(I4,0.3)	(R1,0.66)	(A8, 0.001)	2
(E136,0.1)	(I3,0.4)	(R1,0.1)	(A6,0.1)	(E2,0.1)	(I3,0.2)	(R2,0.5)	(A5,0.001)	3
...	...	...	...	...	...	...	...	
(E2,0.05)	(I134,0.2)	(R5,0.02)	(A2,0.1)	(E7,0.1)	(I372,0.2)	(R4,0.15)		d
...	...	...	...	...	...	...		

Figure 5: Lists L of ranked nodes

Table 2: Statistics on datasets

Dataset	Source	#entities	#entity families
Article	DBLP-a	104106	3
Article	DBLP-b	104295	4
Article	DBLP-c	104338	5
Article	CiteSeer-a	116048	3
Article	CiteSeer-b	146975	5
Article	CiteSeer-c	152647	5
Article	**Total**	728409	25
Movie	IMDB-a	135162	4
Movie	IMDB-b	128738	4
Movie	IMDB-c	128262	4
Movie	Freebase-a	155705	4
Movie	Freebase-b	146412	4
Movie	Freebase-c	140541	4
Movie	**Total**	834820	24

6.2 Pruning Entity Families

To process multiple entity families, we can entirely ranking one entity family after another. For each entity family F_i, it ranks entirely on all the necessary levels d. Referring to Proposition 8, if the current top-k answers (denoted by $K[k]$) already have scores greater than the threshold, $\mathrm{sim}(T_q, K[k]) \ge \mathrm{t}(T_q, F_i, d)$, the remaining entities in F_i can be pruned. Otherwise, we need to rank all the entities in F_i with nodes appearing in the level d. Specifically, for each w_l appearing in T_q and each type_j appearing in F_i, We get the entity T_i in F_i that contains node $L_{lj}[d]$ and rank entity T_i in the top-k list K.

7. EXPERIMENTAL EVALUATION

In this section, we compare our proposal to existing methods. The determination of smooth factor λ is also presented.

7.1 Experimental Setup

Structured entities in two domains, Article and Movie, are considered. Article dataset consists of scientific papers collected from DBLP [2] and CiteSeer [1]. Movie dataset collects movie information from IMDB [4] and Freebase [3].

To simulate a real world data lake, we intentionally add both textual and structural noises to the datasets. As shown in Figures 6 and 7, the datasets are split into multiple parts, each of which uses a different structural representation with

Table 3: Methods in comparison, categorized referring to the discussion in Sections 1 and 2

Category	Method	Comparison on (dis)advantages
Our proposal	combHIS(BOW) combHIS(embedding)	tolerant to structural differences and overlapping contents of the entities
Graph similarity	Labeled-graph-similarity [9] Graph-edit-distance [31]	weak tolerance to structural inconsistency
Entity retrieval	WHIRL(BOW) [13] FSDM [44]	no consideration of overlapping contents
Entity matching/resolution	Entity-resolution [21, 17] Collective-ER [6]	considering only single node entities
Graph/tree vectorization	Keyword-search(combANZ) [19] CP/CV [22]	ineffective in parent-child score combination does not utilize the same structure of entities

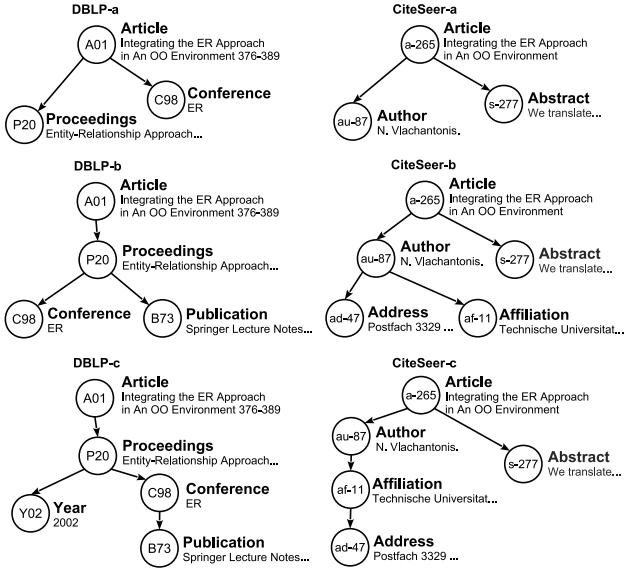

Figure 6: Example structured entities in Article

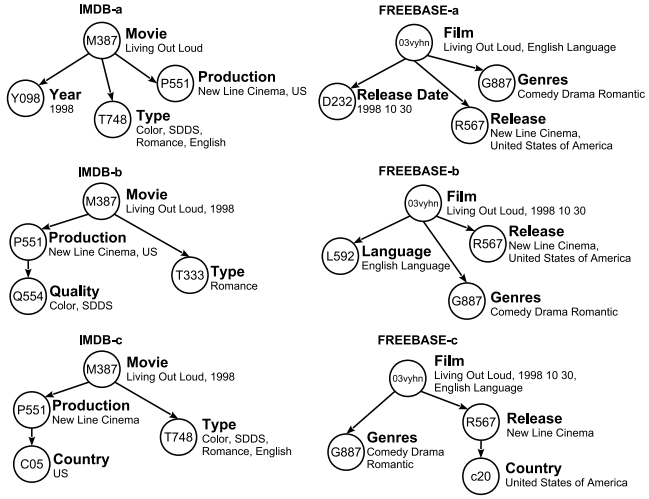

Figure 7: Example structured entities in Movie

significant diversity. For instance, in Figure 6, an article is represented by a two-level structured entity in DBLP-a, while the entities in DBLP-b and DBLP-c have three and four levels, respectively.

In order to simulate the scenarios of synonyms, which may not be frequently observed in articles or movies, we randomly replace words in the datasets by their synonyms. A total number of 732k words (20%) are substituted by the corresponding synonyms from WordNet [30].

Table 2 reports the statistics of datasets. For the Article dataset, there are 728k structured entities (including child entities) categorized into 25 entity families. Recall that an entity family is a set of entities sharing the same structure on types, e.g., the structure of DBLP-a in Figure 6, as defined in Definition 2. For the Movie dataset, there are nearly 835k structured entities categorized into 24 entity families.

For each source in the dataset (say DBLP-a in Figure 6), we randomly draw 200 entities as the query entities for retrieval, including both parent entities (e.g., with head node Article) and child entities (with head node Proceedings). Each query entity is searched in the other sources, and collects top-k answers, for instance, search an entity of DBLP-a in the other DBLP-b, CiteSeer-a and so on. We manually identify the true answers of queries in different sources.

7.2 Comparison to Existing Methods

In this experiment, we compare our proposed methods with existing approaches. As discussed in Section 3.3, the proposed score combination combHIS can work together with both the bag-of-words model, denoted by combHIS(BOW), and the word embedding model, i.e., combHIS(embedding). GloVe [33] is employed as word embeddings. The smooth factor is λ=0.8 in the Article dataset (relies more on the head node in representing and retrieving entities) and 0.6 in the Movie dataset (the determination of the parameter will be discussed in the following experiments).

7.2.1 Compared Approaches

The compared methods are generally the related studies discussed in Sections 1 and 2. Table 3 lists the methods in comparison. Each approach has been fine-tuned by iteratively choosing parameters with the best performance.

The graph/tree edit distance based approaches are discussed in Section 1. (1) Labeled-graph-similarity [9] finds a mapping that matches the nodes of two entities, and then

Table 4: Varying the depth of structures and the number of data sources on Article and Movie datasets

Method	Article dataset					Movie dataset			
	# data sources		depth of structures			# data sources		depth of structures	
	=1	=2	=2	=3	=4	=1	=2	=2	=3
combHIS(BOW)	**0.963**	**0.923**	**0.912**	**0.921**	**0.935**	**0.888**	**0.877**	**0.873**	**0.881**
combHIS(embedding)	0.912	0.903	0.888	0.903	0.918	0.844	0.841	0.842	0.84
Labeled-graph-similarity	0.514	0.512	0.568	0.441	0.526	0.673	0.582	0.523	0.642
Graph-edit-distance	0.525	0.501	0.538	0.476	0.488	0.621	0.55	0.544	0.556
WHIRL(BOW)	0.594	0.586	0.597	0.585	0.576	0.648	0.658	0.658	0.658
FSDM	0.737	0.676	0.697	0.659	0.674	0.852	0.798	0.771	0.825
Entity-resolution	0.625	0.628	0.641	0.609	0.635	0.806	0.672	0.635	0.708
Collective-ER	0.586	0.594	0.588	0.594	0.6	0.585	0.513	0.467	0.558
Keyword-search(combANZ)	0.682	0.414	0.429	0.385	0.426	0.602	0.605	0.496	0.715
CP/CV	0.38	0.401	0.429	0.388	0.385	0.379	0.37	0.375	0.365

combines the label similarity of each pair of matched nodes as the similarity of two entities. (2) Graph-edit-distance [31] considers entities as graphs and computes the edit distance of two graphs as the similarity of the corresponding entities.

The document retrieval approaches for entities with fields are also discussed in Section 1. (3) WHIRL(BOW) [13] combines the contents of a structured entity as an unstructured document and evaluates similarity between combined documents. (4) FSDM [44] considers distinct weights of different fields, and the relative importance of unigrams and bigrams.

The entity resolution methods are discussed in Section 2.1. (5) Entity-resolution first combines the tuples of a structured entity as a "big joined tuple", then finds matching of attributes between big joined tuples [21], and finally combines the value similarity on all matched attributes [17]. (6) Collective-ER [6] considers the entity resolution as a clustering problem, and utilizes the edges between two nodes to merge clusters. Although the method considers connections among entities, each entity is still a tuple without structure.

The keyword search method over structured data is discussed in Section 2.2. (7) Keyword-search(combANZ) [19] studies joining-trees of tuples (as entities) and evaluates their IR-style scores with size normalization. Similarly, we also employ (8) CP/CV [22] for vectorizing and propagating concepts in taxonomies as discussed in Section 2.3.

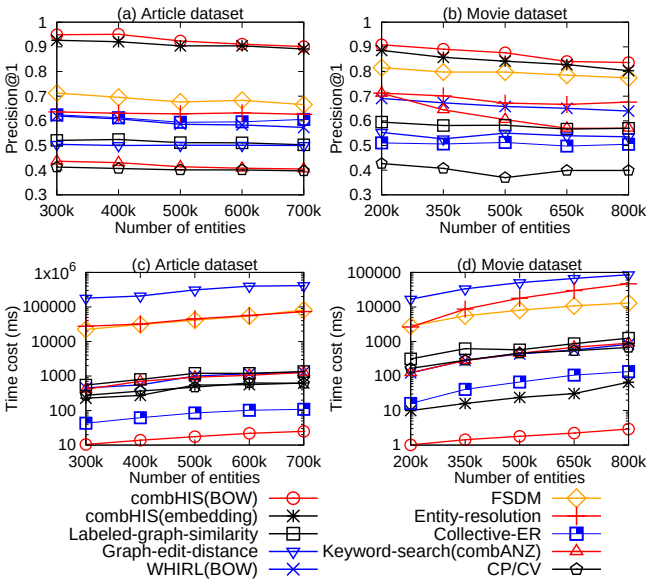

Figure 8: Scalability over various numbers of entities

7.2.2 *Comparison Results*

We evaluate the proposed combHIS cooperating with both the BOW and embedding models as introduced in Section 3.3. Figure 8 presents the scalability over various numbers of entities, and the corresponding time costs.

Our proposed method combHIS(BOW) with the bag-of-words model shows a surprisingly high accuracy in all the tests while keeping the time costs low. The other proposal combHIS(embedding) with word embeddings is comparable to (a bit lower than) combHIS(BOW). The reason is that although word embeddings could capture similar semantics, the contents (in addition to the structures) of entities from DBLP and CiteSeer denoting the same real-world object could be very different. The bag-of-words approach in Formula 2 only considers the terms that occur in both two entities. However, by aggregating the word embeddings as the entity embedding by Formula 3, the resulting embeddings of the entities from DBLP and CiteSeer would be very different as well.

Labeled-graph-similarity and Graph-edit-distance emphasize the structural similarity of two entities. However, as explained at the beginning of this paper, the same real world entities may have different structures. Therefore, the graph similarity based approaches show low accuracy. With tolerance to such structural variations, our combHIS method can achieve significantly higher retrieval accuracy.

The retrieval method WHIRL(BOW) by treating structured entities as unstructured documents, already achieves relatively high precision. It verifies the motivation of this study, i.e., to focus more on the contents of the entities and tolerant to structural differences. Nevertheless, being aware of both the contents and structural information, our combHIS(BOW) shows a clearly better retrieval precision.

FSDM [44] shows better results than WHIRL(BOW), but still not as high as our combHIS(BOW). The result is not

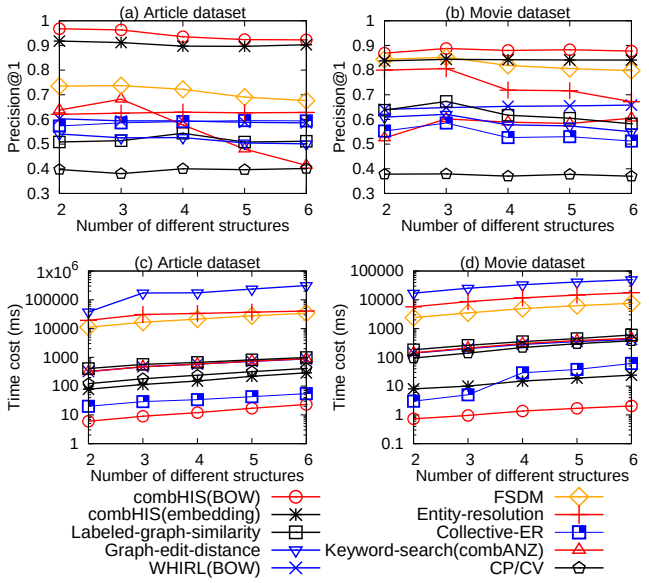

Figure 9: Performances over different structures

surprising, since the unique parent-child containment challenge is not considered.

The Entity-resolution approach by matching heterogenous attributes between entities could further improve the retrieval precision, compared to the WHIRL(BOW) model. Our proposed combHIS(BOW) considers the hierarchy structure of entities rather than a "big joined tuple", and shows the even higher accuracy.

The accuracy of Collective-ER is not high, since it considers only connections among entities where each entity is still a tuple without structure. The method utilizes mainly the contents in the head nodes of entities, while the children contribute only in link structures (instead of contents).

The Keyword-search(combANZ) method with entity size normalization unfortunately shows low precision. The reason is that the same real world entities may have various sizes (number of nodes in an entity). The normalization by entity size in combANZ makes the small entities rank higher and fails to retrieve large entities. The results verify again the analysis of existing approaches in Section 4.

The precision of CP/CV is not as high as our proposal. The reason is that CP/CV considers sum when propagating from children to parents (see Section 2.3). Moreover, the concepts of parents are also propagated to children, i.e., bidirectionally. For instance, all articles are propagated to the corresponding conferences. In this sense, given the highly similar keywords of all articles in SIGMOD and VLDB, it may not be able to distinguish between two conferences.

7.2.3 *Varying the Datasets*

To study other factors in the data that may affect the performance of the approaches. In addition to varying the number of entities in Figure 8 in Section 7.2.2, and the smoothing factor in Figures 10 and 11 in Section 7.3, we report the results on various number of data sources, various number of different structures, and various depth of structures.

Figure 9 reports the results by varying the number of different structures, from 2 data sources. For instance, in Figure 9(a) on Article, the first 3 different structures are from DBLP in Figure 6, and the last 3 different structures are from CiteSeer. The experiments range from 2 to 6 different structures. As shown, with the increase of the number of different structures from various data sources, the accuracy of returned results may drop (slightly). The result is not surprising, since handling the heterogeneity on entity structures is challenging as discussed in Section 1.1.

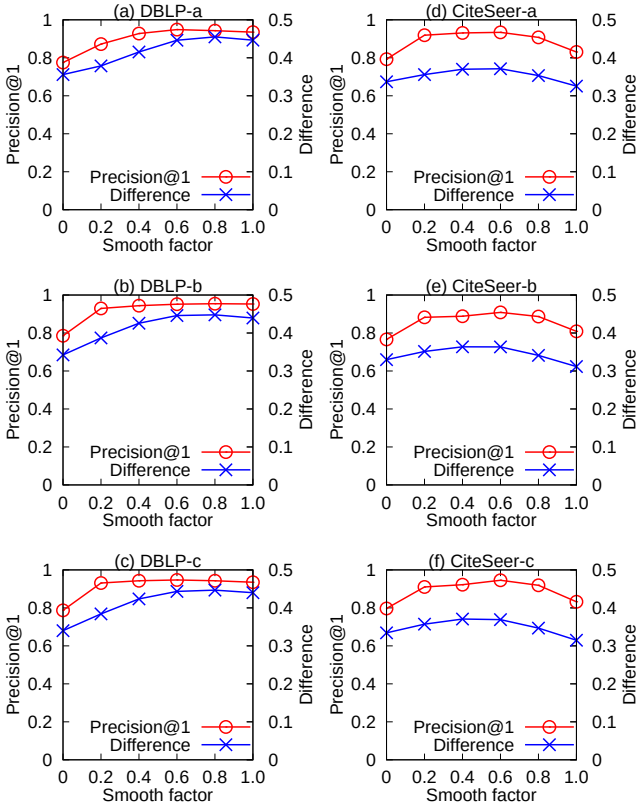

Figure 10: Determining smooth factor λ by the largest difference between top-1 and top-2 answers

Table 4 presents the results with various number of data sources and various depth of structures. For instance, by 2 data sources in Article, we consider all 6 structures in both DBLP and CiteSeer in Figure 6, while a single data source means only the 3 structures in DBLP. A small structure such as DBLP-a has only two levels in depth, while DBLP-c is a larger structure with four levels. Since the properties in Movie are limited, we can construct entities with only 2 or 3 levels in depth. Nevertheless, the results of various approaches are generally proportional to those with different structures in Figure 9. The results demonstrate again the performance and robustness of our proposal.

7.3 Evaluation on Smooth Factor

We analyze the effect of varying the smooth factor λ in the combHIS combination. As introduced in Section 5.3, the larger the λ is, the more the retrieval relies on the head nodes of entities. To practically determine whether a λ is sufficient to distinguish the importance of head nodes, we observe the normalized difference of the top-1 and top-2 answers, i.e., $\frac{\text{sim}_1 - \text{sim}_2}{\text{sim}_1}$, where sim_1 and sim_2 are the entity similarities

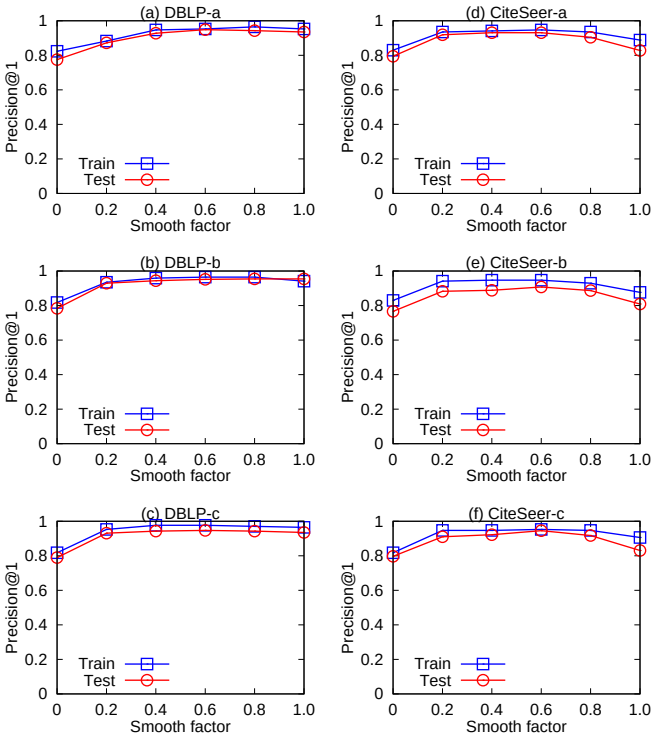

Figure 11: Determining smooth factor λ by a set of training examples leading to the highest precision

(e.g., computed by Formula 2) of the top-1 and top-2 results, respectively.

Figure 10 presents the precision with various λ settings. (1) For $\lambda = 0$, as analyzed in Proposition 6, combHIS is equivalent to combSUM. The precision of $\lambda = 0$ in Figure 10 is low in this case, since the parent entities will always have higher scores and the child entities could never be retrieved. In particular, the difference of top-1 and top-2 answers is low in such as a case, i.e., the λ is not sufficient to distinguish the entities. (2) With the increase of λ, head node becomes more important in an entity. Referring to Proposition 5, some child entities may have higher scores in this case, and could be retrieved. The retrieval precision increases in Figure 10. The difference of top-1 and top-2 answers increases as well, i.e., the λ becomes more distinguishing. (3) However, if λ is set too large, e.g., 1.0 in Figure 10(d), only the head node score will be counted, while the children's contributions are completely ignored. The precision may drop. Coincidently, the difference of top-1 and top-2 answers drops, i.e., the λ is not able to distinguish the entities again.

One may choose λ automatically by observing the difference of similarities in top-k answers. In a real data lake, the same entity might be represented in various ways. We can set an individual λ value for each data source with different structures. It is true that such an individual λ value drastically increases the amount of tuning. Fortunately, the tuning task is performed automatically without much manual effort, i.e., selecting a λ value which leads to the largest difference of top-1 and top-2 answers, such as $\lambda = 0.8$ in Figure 10(a). As shown, the corresponding precision is high, since such a λ is able to distinguish most the top-1 and top-2 answers. Indeed, a similar λ may be sufficient to apply to the structures from alike sources, i.e., $\lambda = 0.8$ for DBLP-a, b and c, and $\lambda = 0.4$ for CiteSeer-a, b and c, according to Figure 10. The rationale refers to the analysis in Section 5.3. Since article names are often different from the conference and proceeding information in DBLP, as illustrated in Figure 6, the head node may be sufficient to retrieval and thus a large λ is preferred. In CiteSeer, however, article names might be similar to the contents in the abstract, and thus head node is not sufficient. A small λ could be used to consider more children's contributions.

If a set of (query, positive example, negative example) tuples are available as training data, we can learn more easily a λ that maximized the positive examples and minimized the negative examples. For instance, in Figure 11(d), $\lambda = 0.4$ with the highest precision in the training data will be chosen and applied in the subsequent queries to the data source of CiteSeer-a, i.e., test phase. As shown, in most tests, a λ with higher precision in training leads to better results in testing. Indeed, the precisions of training and testing are very close. In this sense, the determination of λ by a set of (query, positive example, negative example) tuples would be more accurate than observing the difference of top-1 and top-2 answers in Figure 10.

8. CONCLUSIONS

In this paper, we study the problem of retrieving structured entities. As observed in our experiments, by directly treating structured entities as unstructured documents, the bag-of-words model already shows considerably high retrieval precision. We argue that by further integrating the structural information, the retrieval could be improved. On the other hand, however, existing graph similarity approaches emphasize too much on structural inconsistency and thus may not work well in entity retrieval (as also observed in our experiments). To adapt the successful bag-of-words/word embedding models with structure, the core is to combine the term scores of different nodes in a structured entity. Again, directly applying the sum combination of scores (with entity size normalization), widely considered in keyword search over structured data, is not effective as analyzed in Section 4 (Propositions 1 & 2) and observed in our experiments.

Recognizing the challenge that one entity may contain another and the opportunity that entities share the same structures, in this paper, we devise a novel score combination function, combHIS, by learning score aggregation coefficient ρ_F from entities sharing the same structure. To further tune individually the importance of the head node in an entity and distinguish parent from its children, we introduce a smooth factor λ in the proposed combination function. Proposition 6 shows that sum combination is indeed a special case of our combHIS with $\lambda = 0$. The parameter of smooth factor λ can be practically determined by observing the difference on similarity scores of the returned entity results (Figure 10). The proposed method may cooperate with both the bag-of-words (BOW) and word embedding models, successful in retrieving unstructured documents, for structured entities. Extensive experiments demonstrate both effectiveness and efficiency of our proposal.

Acknowledgement

This work is supported in part by the National Key Research and Development Plan (2019YFB1705301) and the National Natural Science Foundation of China (61572272, 71690231).

9. REFERENCES

[1] http://citeseerx.ist.psu.edu.

[2] https://dblp.uni-trier.de/xml/.

[3] http://www.freebase.com/.

[4] http://www.imdb.com/.

[5] A. Balmin, V. Hristidis, and Y. Papakonstantinou. Objectrank: Authority-based keyword search in databases. In *VLDB*, pages 564–575, 2004.

[6] I. Bhattacharya and L. Getoor. Collective entity resolution in relational data. *TKDD*, 1(1), 2007.

[7] C. Böhm, G. de Melo, F. Naumann, and G. Weikum. LINDA: distributed web-of-data-scale entity matching. In *21st ACM International Conference on Information and Knowledge Management, CIKM'12, Maui, HI, USA, October 29 - November 02, 2012*, pages 2104–2108, 2012.

[8] M. Cataldi, K. S. Candan, and M. L. Sapino. Narrative-based taxonomy distillation for effective indexing of text collections. *Data Knowl. Eng.*, 72:103–125, 2012.

[9] P. Champin and C. Solnon. Measuring the similarity of labeled graphs. In *ICCBR*, pages 80–95, 2003.

[10] L. J. Chen and Y. Papakonstantinou. Supporting top-k keyword search in XML databases. In *ICDE*, pages 689–700, 2010.

[11] Y. Chen, W. Wang, Z. Liu, and X. Lin. Keyword search on structured and semi-structured data. In *SIGMOD*, pages 1005–1010, 2009.

[12] V. S. Cherukuri and K. S. Candan. Propagation-vectors for trees (PVT): concise yet effective summaries for hierarchical data and trees. In *Proceeding of the 2008 ACM Workshop on Large-Scale Distributed Systems for Information Retrieval, LSDS-IR 2008, Napa Valley, California, USA, October 30, 2008*, pages 3–10, 2008.

[13] W. W. Cohen. Integration of heterogeneous databases without common domains using queries based on textual similarity. In *SIGMOD Conference*, pages 201–212, 1998.

[14] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. In *PODS*, 2001.

[15] M. Farah and D. Vanderpooten. An outranking approach for rank aggregation in information retrieval. In *SIGIR*, pages 591–598, 2007.

[16] S. Guha, H. V. Jagadish, N. Koudas, D. Srivastava, and T. Yu. Integrating XML data sources using approximate joins. *ACM Trans. Database Syst.*, 31(1):161–207, 2006.

[17] S. Guha, N. Koudas, A. Marathe, and D. Srivastava. Merging the results of approximate match operations. In *VLDB*, pages 636–647, 2004.

[18] N. Hamilton. The mechanics of a deep net metasearch engine. In *Proceedings of the Twelfth International World Wide Web Conference - Posters, WWW 2003, Budapest, Hungary, May 20-24, 2003*, 2003.

[19] V. Hristidis, L. Gravano, and Y. Papakonstantinou. Efficient IR-style keyword search over relational databases. In *VLDB*, pages 850–861, 2003.

[20] A. R. Jaiswal, D. J. Miller, and P. Mitra. Schema matching and embedded value mapping for databases with opaque column names and mixed continuous and discrete-valued data fields. *ACM Trans. Database Syst.*, 38(1):2:1–2:34, 2013.

[21] J. Kang and J. F. Naughton. Schema matching using interattribute dependencies. *IEEE Trans. Knowl. Data Eng.*, 20(10):1393–1407, 2008.

[22] J. W. Kim and K. S. Candan. CP/CV: concept similarity mining without frequency information from domain describing taxonomies. In *Proceedings of the 2006 ACM CIKM International Conference on Information and Knowledge Management, Arlington, Virginia, USA, November 6-11, 2006*, pages 483–492, 2006.

[23] S. Lacoste-Julien, K. Palla, A. Davies, G. Kasneci, T. Graepel, and Z. Ghahramani. Sigma: simple greedy matching for aligning large knowledge bases. In *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013*, pages 572–580, 2013.

[24] W. Le, F. Li, A. Kementsietsidis, and S. Duan. Scalable keyword search on large RDF data. *IEEE Trans. Knowl. Data Eng.*, 26(11):2774–2788, 2014.

[25] J. Lilleberg, Y. Zhu, and Y. Zhang. Support vector machines and word2vec for text classification with semantic features. In *14th IEEE International Conference on Cognitive Informatics & Cognitive Computing, ICCI*CC 2015, Beijing, China, July 6-8, 2015*, pages 136–140, 2015.

[26] F. Liu, C. T. Yu, W. Meng, and A. Chowdhury. Effective keyword search in relational databases. In *SIGMOD Conference*, pages 563–574, 2006.

[27] Y. Luo, X. Lin, W. Wang, and X. Zhou. SPARK: Top-k keyword query in relational databases. In *SIGMOD Conference*, 2007.

[28] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[29] D. Milano, M. Scannapieco, and T. Catarci. Structure-aware XML object identification. *IEEE Data Eng. Bull.*, 29(2):67–74, 2006.

[30] G. A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, 1995.

[31] M. Neuhaus and H. Bunke. A convolution edit kernel for error-tolerant graph matching. In *ICPR*, pages 220–223, 2006.

[32] A. Nierman and H. V. Jagadish. Evaluating structural similarity in XML documents. In *WebDB*, pages 61–66, 2002.

[33] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543, 2014.

[34] G. Salton and M. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill Book Company, 1984.

[35] M. Sayyadian, H. LeKhac, A. Doan, and L. Gravano. Efficient keyword search across heterogeneous relational databases. In *ICDE*, pages 346–355, 2007.

[36] J. Seo and W. B. Croft. Geometric representations for multiple documents. In *SIGIR*, pages 251–258, 2010.

[37] J. A. Shaw and E. A. Fox. Combination of multiple searches. In *Proceedings of The Third Text REtrieval Conference, TREC 1994, Gaithersburg, Maryland, USA, November 2-4, 1994*, pages 105–108, 1994.

[38] R. Shaw, A. Datta, D. E. VanderMeer, and K. Dutta. Building a scalable database-driven reverse dictionary. *IEEE Trans. Knowl. Data Eng.*, 25(3):528–540, 2013.

[39] A. Singhal. Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.*, 24(4):35–43, 2001.

[40] K. Sparck Jones. Index term weighting. *Information Storage and Retrieval*, 9(11):619–633, 1973.

[41] M. Weis and F. Naumann. Dogmatix tracks down duplicates in XML. In *SIGMOD Conference*, pages 431–442, 2005.

[42] J. X. Yu, L. Qin, and L. Chang. *Keyword Search in Databases*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2010.

[43] W. Zheng, L. Zou, X. Lian, J. X. Yu, S. Song, and D. Zhao. How to build templates for RDF question/answering: An uncertain graph similarity join approach. In *SIGMOD Conference*, pages 1809–1824, 2015.

[44] N. Zhiltsov, A. Kotov, and F. Nikolaev. Fielded sequential dependence model for ad-hoc entity retrieval in the web of data. In *SIGIR*, pages 253–262, 2015.

[45] Y. Zhou, H. Cheng, and J. X. Yu. Graph clustering based on structural/attribute similarities. *PVLDB*, 2(1):718–729, 2009.

[46] X. Zhu, S. Song, X. Lian, J. Wang, and L. Zou. Matching heterogeneous event data. In *SIGMOD*, pages 1211–1222, 2014.

[47] L. Zou and M. T. Özsu. Graph-based RDF data management. *Data Science and Engineering*, 2(1):56–70, 2017.