Building Taxonomies with Triplet Queries

Donatella Firmani[1],*, Sainyam Galhotra[2], Barna Saha[3] and Divesh Srivastava[4]

[1] *Sapienza University*
[2] *Cornell University*
[3] *University of California San Diego*
[4] *AT&T Chief Data Office*

Abstract

The organization of records referring to different entities into a taxonomy is crucial for capturing their relationships. Nevertheless, the automatic identification of such relationships often faces inaccuracies due to noise and heterogeneity of records across various sources. Simultaneously, manual maintenance of these relationships proves impractical and lacks scalability. This study addresses these challenges by adopting a weak supervision strategy, in the form of an oracle, to solve a novel *Hierarchical* Entity Resolution task. Within our framework, records are organized into a tree-like structure that encompasses records at the bottom level and encapsulates entities and categories at the higher levels. To make the most effective use of supervision, we employ a *triplet comparison* oracle, which takes three records as input and output the most similar pair(s). Finally, we introduce `HierER`, a querying strategy utilizing record pair similarities to minimize the number of oracle queries while simultaneously maximizing the identification of the hierarchical structure. Theoretical and empirical analyses demonstrate the effectiveness and efficiency of `HierER` with noisy datasets with millions of records.

1. Introduction

In many applications, records are represented in diverse formats like images, unstructured and structured text and these records need to be organized to capture complex relationships. Assigning records to taxonomies is useful for diverse applications like recommendation [1], categorization [2], and search [3]. For example, e-commerce websites like Amazon organize products in the form of a taxonomy to enable better search and recommendations. Animal and plant species along with their textual descriptions can be arranged in hierarchies of varying depth, average degree and shape, called pylogenetic trees. In Figure 1a we show an example hierarchy for arranging a collection of records describing birds. Example records are r_a and r_b, describing respectively the little bunting (Emberiza pusilla) and the yellow-breasted bunting (Emberiza aureola). Example categories in the hierarchy is u_1. Note that categories are not required to have a descriptive name: in this example the category u_1 is simply defined as the set comprising the little bunting (r_a) and the yellow-breasted bunting (r_b).

Two main types of relationships arise naturally in a hierarchy:

- *is-A* (in orange) connecting record-nodes such as r_a to category-nodes such as u_1;
- *category-supercategory* (in black) connecting pairs of category-nodes such as u_1 to u_2;

SEBD 2024: 32nd Symposium on Advanced Database Systems, June 23-26, 2024, Villasimius, Sardinia, Italy

*Corresponding author.

donatella.firmani@uniroma1.it (D. Firmani); sg@cs.cornell.edu (S. Galhotra); bsaha@ucsd.edu (B. Saha); divesh@research.att.com (D. Srivastava)

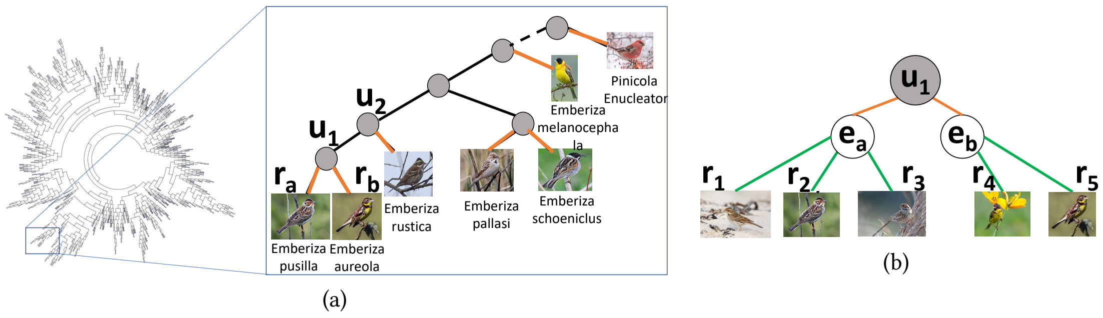


Figure 1: (a) Example hierarchy for phylogenetic data. The center of the diagram corresponds to the root and the subsequent levels are placed in concentric circles. (b) Entity-nodes and duplicate records.

In order to account for duplicate records, that are, different records referring to the same entity in the real world, we introduce a third type of relationship, that we call *co-reference*. In Figure 1b, the co-reference relationship is shown in green. To represent this relationship we introduce intermediate entity-nodes, such as e_a and e_b, between record-nodes and category-nodes. In the figure, records r_1, r_2 and r_3 are different pictures of little bunting, which is now represented by e_a, while records r_4, r_5 both represent a yellow-breasted bunting, that we now call e_b. Both e_a and e_b belong to the u_1 category, which can be equivalently defined as a set of entities.

Starting from records $r_1, \ldots, r_n$, our task is to return the complete tree structure, identifying duplicate records and enriching them with is-A and category-supercategory relationships. We call this problem as the *Hierarchical Entity Resolution* (HER) problem. The popular Entity Resolution (ER) problem corresponds to identification of record-entity relationships only (see green edges in Figure 1b) and thus our problem represents a natural extension.

Before our work there are two main roads to the solution of HER.

- **Fully automatic construction.** To automatically construct categories and identify their relationships, prior techniques have proposed to leverage co-occurence patterns of hypernyms [4, 5]. However, certain categories may not always be explicitly mentioned in the records [6] nor respect the same terminology across different sources.
- **Fully manual Construction.** A domain expert can easily generate a hierarchy that contains all well-known categories. However, it might be impossible to capture all the fine-grain relationships in the data (such as, very specialized category-nodes like u_1 and u_2 in Example 1) without processing all the records manually, which is not scalable.

Even though the benefits of constructing a hierarchy are widely acknowledged, due to limitations of fully automatic construction, the majority of category-supercategory relationships are maintained manually by domain experts. Manual maintenance of such relationships is labor-intensive as the hierarchy evolves whenever new records are introduced/discontinued.

Our intuition. We aim at using a hybrid *weakly supervised* approach; that is, an automatic approach that is guided by domain experts providing answers to targeted queries. While manually processing all the records might be infeasible, if only *three* records are considered in isolation, say r_1, r_2 and r_4, then a user – or even a trained classifier – can easily distinguish that r_1 and r_2 are closer to each other than either of them is to r_3. Even though considering record triplets in isolation can help uncover the hierarchical structure, it is still unfeasible to compare all possible triples in million scale datasets. Our framework is capable of automatically prioritizing records to optimize the number of triplet comparisons and to minimize the query workload. We refer to the domain expert (or the classifier) using the abstraction of a black-box *oracle*. Oracle-based algorithms have been widely popular to study fairness [7], correlation clustering [8] and classification [9, 10], identify maximum elements [11, 12], top-k elements [13, 14, 15], information retrieval [16], skyline computation [17], and so on. Our oracle can answer to:

- triplet queries: "which pair of records among u, v and w are most similar?"
- co-reference queries: "do records u and v refer to the same entity?";

These queries can reveal the local hierarchical structure and can be answered without requiring (i) the context of all the identified categories in the constructed hierarchy, nor (ii) the knowledge of other records in the dataset. Recent advancements of deep-learning based classifiers for ER [18, 19, 20, 21] are also alternative implementation of our oracle.

Related works and oracle-based methods. The closest task to our problem is ER, featuring a variety of oracle-based methods [22, 23, 24]. However ER techniques typically ignore category information.Another related task is Hierarchical Clustering. This has been studied in a variety of application domains including the construction of phylogenetic trees [25, 26] and taxonomies [27, 28]. The work in [26] describes an oracle-based method. In Hierarchical Clustering, records refer to different entities, and thus clustering methods ignore entity resolution. Moreover, these techniques build almost binary hierarchies, where every node has approximately two children. Thus, neither ER nor Hierarchical Clustering techniques can by itself solve our HER problem effectively. Even pipelining the two processes turns out to achieve a sub-optimal query workload. Let n be the number of records. Running a hierarchical clustering technique first and then post-processing the bottom level in order to detect entities can require $O(n^2)$ oracle queries for non-binary hierarchies in the worst case [26], before even identifying the entities. Running an oracle-based ER technique first and post-processing entities after that to detect categories can be efficient in case of large entities but can require $O(n^2)$ queries to identify small entity clusters [23], before even starting to process categories.

Contributions and Outline. We show that previous methods can be significantly outperformed by our approach, thus alleviating the manual workload required for construction and maintenance of taxonomies over very large databases. Section 2 contains a high-level overview of our approach, and in particular of the `HierER` algorithm. Detailed description of the algorithm and theoretical analysis can be found in [29]. Main experimental results of [29] are reported in Section 3. Finally, Section 4 contain our concluding remarks.

2. Overview

Let $V = \{v_1, v_2, \ldots, v_n\}$ be a collection of n records. We use the notion of *laminar* family of sets to define the ground truth hierarchy H^* more formally. A family of sets $\mathcal{C}^*$ is laminar iff $\forall X_1, X_2 \in \mathcal{C}^*$, either $X_1 \cap X_2 = \phi$ or $X_1 \subseteq X_2$ or $X_2 \subseteq X_1$. The hierarchy H^* corresponds to a laminar family of labelled sets $\mathcal{C}^*$ such that each set in $\mathcal{C}^*$ is labelled with one of the three labels: `record (r)`, `entity (e)` or `category (t)`. Each labelled set is denoted as $\langle \texttt{label} : X \rangle$ where $X \subseteq V$. According to this notation, $\langle \texttt{r} : \{v\} \rangle \in \mathcal{C}^*, \forall v \in V$ and $\langle \texttt{t} : V \rangle \in \mathcal{C}^*$. This hierarchy has an additional constraint that a set labelled '`entity`' cannot have a proper superset of label '`entity`' or '`record`' and a set labelled '`category`' cannot have a superset labelled 'entity' or 'record'. Following this definition, there exists a one-to-one mapping between internal nodes of the hierarchy and the laminar family of sets $\mathcal{C}^*$ where an internal node of the hierarchy (say u) is equivalent to a set $C \in \mathcal{C}^*$ containing all the leaf-level descendants of u and vice versa. In this formulation, a category node that has a single category node as a child is redundant and can be ignored. However, an entity can have a single child (record).

A hierarchy has an interesting property that for any three records $v_1, v_2, v_3 \in V$, the lca's of two pairs of these records are the same and the lca of the third pair is either the same or a descendant of the other two lca's [26]. Without loss of generality, one of the following hold.

$$\texttt{lca}(v_2, v_3) \geq \texttt{lca}(v_1, v_2) = \texttt{lca}(v_1, v_3)$$

where $\texttt{lca}(v_1, v_2) > \texttt{lca}(x, y)$ denotes that $\texttt{lca}(v_1, v_2)$ is a descendant of $\texttt{lca}(x, y)$. In case all three lca's are the same, then v_1, v_2 and v_3 belong to three different descendant-branches of the internal node corresponding to $\texttt{lca}(v_1, v_2) = \texttt{lca}(v_1, v_3) = \texttt{lca}(v_2, v_3)$. A *triplet oracle* is a function $q_t : V \times V \times V \rightarrow V \cup \{\phi\}$ that takes three records as input and outputs the farthest record (if any). For an input (v_1, v_2, v_3), the oracle outputs $q_t(v_1, v_2, v_3) = v_1$ if $\texttt{lca}(v_2, v_3) > \texttt{lca}(v_1, v_2) = \texttt{lca}(v_1, v_3)$ and $q_t(v_1, v_2, v_3) = \phi$ if $\texttt{lca}(v_1, v_2) = \texttt{lca}(v_1, v_3) = \texttt{lca}(v_2, v_3)$.

Prioritizing triplet oracle queries to maximize the accuracy in the H^* estimation task after every query, is the goal of the high-level workflow presented in Figure 2. Note that evaluating accuracy is non-trivial. A naive way is to compare the fraction of the total $\Theta(n^3)$ relationships (i.e., $\binom{n}{3}$ triplets and $\binom{n}{2}$ equality relationships) that are correctly identified by a given method. However, such an approach can be infeasible even in medium-sized datasets. Therefore, we extend the popular metric of comparing *F-score* of different ER techniques to our hierarchical setting. In ER, F-score is computed over two types of pairwise relationships: intra-cluster and inter-cluster. Following these ideas, we consider co-reference relationships as intra-cluster and enumerate the different types of inter-entity relationships between record pairs. We define the notion of t-ancestor relationship to capture the distance between record pairs and then use it to compute the F-score of the output hierarchy H. A pair of records (u, v) satisfies a t-ancestor relationship if their lca is at most t edges away from both u and v's entity nodes. 0-ancestor relationship is equivalent to an equality (i.e., co-reference) relationship. We map t-ancestor relationships identified from the output hierarchy H to the ground truth hierarchy H^* and then use those to compute precision as the weighted fraction of correctly identified pairwise relationships among the identified relationships and recall as the weighted fraction of total relationships that were identified. Weighting mechanisms are discussed in detail in [29]. F-score is finally computed as the harmonic mean of precision and recall.

Auxiliary modules. In the following, we first describe the auxiliary modules in Figure 2, then we provide more intuition on the core **Oracle strategy** module.

- The **Blocking** module is used to reduce the number of pairs considered for similarity computation. Blocking is a widely used operation in ER literature [30] to efficiently generate a small set of candidate pairs so that similarity values are computed only for this small set of candidates. Standard blocking (also known as token-based blocking) is one of the most popular mechanisms that generates a block for each token in the input set of records [30]. Similarity values may not be directly interpretable as probability distributions over the possible oracle responses. For practical purposes, calibration approach in [23, 31] can be used to map values $s(v_1, v_2)$ to probability distributions $p(v_1, v_2)$ and $p(v_1, v_2, v_3)$.

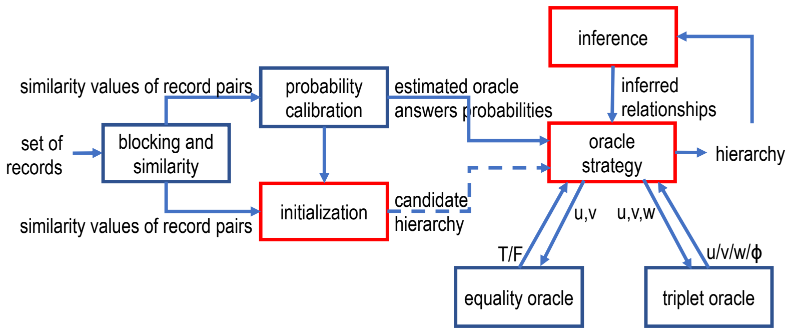


Figure 2: Our hierarchical ER workflow.

- The **Initialization** module in our workflow constructs a candidate hierarchy $\bar{H}$ that can be used downstream to guide the querying strategies. Construction of $\bar{H}$ is based solely on the similarity scores $s : V \times V \to [0, 1]$ and requires no oracle queries. We prove in [29] that $\bar{H}$ has high F-score under low noise of similarity values.
- The **Inference** module provides tools to *infer* relationships from previously asked triplet and equality queries, without asking new oracle queries leading to unnecessary cost. Such module allows to infer, for instance, which is the farthest record among v, w and x by looking at the result of previous queries about other triples like (u, v, w) and (u, w, x).

We observe that inferring category-subcategory relationships is the major challenge, whereas co-reference can be easily inferred via transitive closure.[1] Consider three records r_1, r_2, r_6 and a query $q_1 \equiv q_t(r_1, r_2, r_6)$ that returns r_6. To interpret q_1 mathematically, we define three variables corresponding to the lca's of involved record pairs (r_1, r_2), (r_1, r_6) and (r_2, r_6). Using these variables, q_1 can be represented as $\texttt{lca}(r_1, r_2) > \texttt{lca}(r_1, r_6) = \texttt{lca}(r_2, r_6)$. This inequality characterizes a relation between the lca of record pairs (r_1, r_2), (r_1, r_6) and (r_2, r_6). Each query can be written in the form of such inequality constraints over at most $\binom{n}{2}$ lca variables. Consider another query $q_2 \equiv q_t(r_1, r_6, r_11) = r_11$, meaning $\texttt{lca}(r_1, r_6) > lca(r_1, r_11) = \texttt{lca}(r_6, r_11)$. Using the inequalities of q_1 and q_2, we can infer that $\texttt{lca}(r_1, r_2) > \texttt{lca}(r_1, r_6) > \texttt{lca}(r_1, r_11)$.

Oracle strategy. This module prioritizes oracle queries by leveraging (i) the candidate hierarchy from the initialization step to give higher priority to queries that yield higher F-score increment and (ii) the inference engine to identify inferable relationships for free. The algorithm `HierER` described in [29] can be thought of as an oracle strategy that leverage the initialization and inference methods and focus on the following principles to maximize progressive F-score.

- *Internal node discovery.* Prioritize queries that enable the discovery of new internal nodes. Indeed, identifying the tree structure, specifically the internal nodes between the root node and the leaf nodes corresponding to the processed records is important to provide optimal progressive behavior.
- *Large entities.* Give high priority to queries enabling the discovery of new children of high-degree entity nodes. This principle has also been used in ER literature [23, 22] since asking queries in non-increasing order of entity sizes provides the maximum gain in

[1] E.g., if we know that v_1 refers to the same entity as v_2, and v_2 refers to the same entity as v_3, then we can infer that v_1 refers to the same entity as v_3 without asking the corresponding oracle query.

Dataset	n	Depth	Average degree	Max degree	Largest Entity
Phylogenetic	1039	21	2.01	4	1
DMOZ	100K	5	274.3	17K	1
Cars	16.5K	2	330	1800	1800
Camera	30K	3	5	91	91
Amazon	30K	7	8.46	760	1
Geography	3M	5	8K	35K	1

Figure 3: Dataset description.

progressive recall.

- *Connectivity.* Prioritize queries that grow the hierarchy in a connected fashion. Indeed, ensuring that the processed records at any given time form a single connected hierarchy (rather than growing multiple disjoint hierarchies in parallel) allows for the inference of more relationships with the same number of queries.

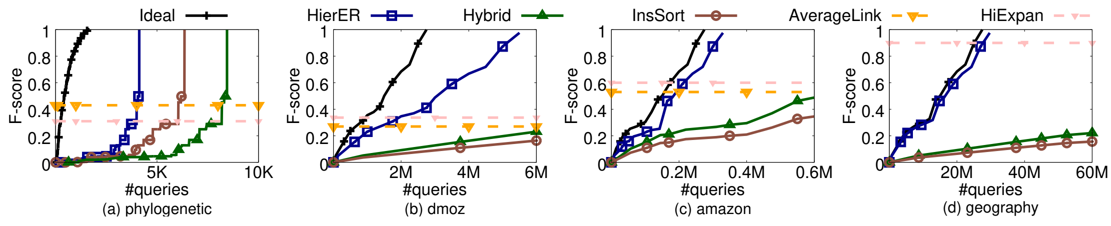
3. Experiments

We now compare HierER and baselines on real-world datasets and answer the following questions. **Q1:** What is the end-to-end quality vs query complexity trade off for HierER? **Q2:** Is HierER sensitive to noise in the dataset? **Q3:** Is HierER scalable to large-scale datasets?

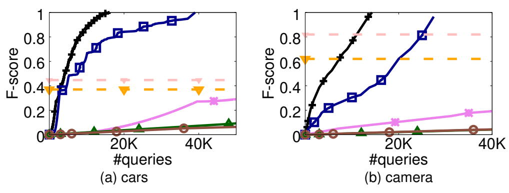
Set-up. Figure 3 reports the six real-world datasets in our experiments, consisting of hierarchies of varying depth, average degree and shape, and comprising either textual (records) or visual description (images) of entities. Phylogenetic contains scientific names of bird and insect species along with textual descriptions collected from Wikipedia. The hierarchy corresponds to the phylogenetic tree obtained from [32]. Figure 1a in Section 1 shows the shape of such hierarchy. DMOZ [33] is an open-content directory of web pages along with a hierarchy that organizes these webpages according to their categories like art, science, mathematics, etc. Cars comprises images of different models of cars. For Cars we generate textual descriptions using Google's vision API [34] and hierarchy is constructed based on their make and model. Camera [35] is a collection of specifications of cameras collected from over 25 retail companies and the hierarchy corresponds to the brand-model categorization. Amazon [36] contains descriptions of products and the amazon catalog ontology. Geography [37] contains names of cities across the world with categories corresponding to their state, country and continent.

Due to low-training data requirements for random forests as compared to deep-learning based techniques [20], we used a random forest classifier trained with active learning for cars, geography and camera datasets as an oracle. Such model achieved more than 0.95 F-score for all three datasets. For other datasets we consider a simulated oracle model (i.e., using ground truth hierarchy to generate responses) to control the noise level. For more implementation details such as similarity calculation and triplet probability we refer the reader to [29].

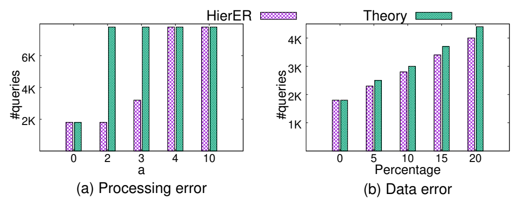
We consider the following pipelined baseline strategies that perform hierarchical clustering and entity resolution separately as a two-step procedure. (i) Average Linkage (denoted by AverageLink) is an Agglomerative clustering technique. We used the sklearn package [38] to construct the hierarchy and then run triplet queries bottom up to merge neighboring internal nodes. (ii) HiExpan denotes the automated taxonomy construction technique from [28] that uses the initial seed hierarchy to generate other internal nodes assuming access to all internal nodes



(a)



(b)



(c)

Figure 4: F-score vs #queries for datasets where (Top) all records refer to distinct entities and (bottom-left) datasets have many co-references. (Bottom-right) Query complexity for varying similarity noise.

in the hierachy. We added the internal nodes to run this algorithm. (iii) `InsSort` considers the extension of the insertion sort algorithm [26] for non-binary hierarchies to generate a hierarchy, followed by `Hybrid` [23] to identify equality relationships. (iv) The pipeline that performs ER followed by hierarchical clustering has almost zero F-score at the end of ER phase when the datasets contain singleton entities and therefore suffer from poor progressiveness. Instead, we consider `Hybrid` as an adaptation of the state-of-the-art entity resolution strategy [23].

Q1: Result Quality. In order to answer to our first question, we compare the quality of techniques by measuring the progressive F-score. The F-score value after every query is computed by mapping the relationships in the constructed hierarchy to those in the the ground truth hierarchy. Figures 4a and Figures 4b compare the F-score of `HierER` with other baselines on multiple datasets. Across all datasets, `HierER` achieves the highest progressive F-score and is closest to the `ideal` curve. `InsSort` and `Hybrid` achieve poor progressive F-score. These techniques require more than $5\times$ the queries required by `HierER` to achieve comparable F-score across all large scale datasets. `AverageLink` generates a hierarchical clustering over the records without any oracle queries. This hierarchy achieves non-zero F-score but the mistakes in the constructed hierarchy can not be corrected. `HiExpan` performs better than `AverageLink` for most datasets but does not achieve high F-score. It is sensitive to the initial structure provided as input and does not generalize if it does not contain all levels of the hierarchy. Due to the presence of noise in datasets, such automated techniques do not achieve high F-score.

We observe different behaviors across datasets. In `phylogenetic` dataset, the ideal requires $< 2n$ queries as majority of the internal nodes have two children and require less than 2 queries to be inserted. For each leaf level record, `HierER` requires $O(\log n)$ queries to identify its location in the processed hierarchy [29], performing better than `InsSort` but requiring much more queries than the `ideal` strategy. This is due to high noise in similarity values, with more than 85% of the records suffering from data error in this dataset. Among other datasets, including those with internal nodes having much higher average degree, `HierER` achieves

Dataset	HierER	InsSort	Hybrid	AverageLink
Phylogenetic	**1min 40sec**	1min 50sec	1min 53sec	2min 10sec
DMOZ	**1hr 23min**	4hr 47min	6hr 20min	3hr 17min
Cars	**45min**	3hr 25min	5hr 30min	3hr 5min
Camera	**52min**	2hr 47min	3hr 35min	3hr 10min
Amazon	**1hr 5min**	3hr 27min	3hr 55min	3hr 40min
Geography	**11hr 35min**	30hr 35min	34hr 35min	Did Not finish

Figure 5: Running Time Comparison

near-optimal progressive F-score and performs much better than all other baselines.

The experiments in Figure 4 assumes that the oracle answers all queries correctly. For experiments with independent triplet error we refer the reader to [29]. To develop *robust* methods, we leverage the random graph toolkit [24] for each oracle query.

Q2: Noise sensitivity. To validate the effect of noise, Figure 4c considers synthetic similarities in the Phylogenetic dataset and compares the query complexity with the theoretically proven bounds in [29], where we establish that HierER requires $O(n \log n)$ triplet comparisons, assuming reasonable data error. Figure 4c (left) simulates *processing error*, where each pairwise similarity ($s(u, v)$) is sampled independently according to a normal distribution with mean $\mu_{(u,v)}$ and variance $\sigma^2 = a\mu^2$ (μ denotes the difference in expected similarity for pairs connected at different depths). The query complexity of HierER is the same as that of ideal (roughly $2n$) for $a < 3$. For higher noise, the query complexity increases but it plateaus at $O(n \log n)$. Figure 4c (right) simulates *data error*, where a random sample of the nodes is erroneous such that all pairwise similarities containing these records are erroneous. In this case, the query complexity of the technique is directly proportional to the error.

Q3: Scalability. Figure 5 compares the running time of HierER to reach 0.90 F-score with respect to other baselines. HierER has significantly lower running time for all the datasets due to the linear dependence of running time on the number of queries, finishing in less than 2 minutes on Phylogenetic and in less than 12 hrs on the million-scale Geography dataset.

4. Conclusions

In this paper, we have formalized the Hierarchical Entity Resolution problem, introducing an oracle-based approach. Our algorithm, HierER, presents a novel query ordering strategy that capitalizes on pairwise record similarities, prioritizing triplet and equality oracle queries. Theoretical and empirical analysis underscores that HierER is capable of constructing an accurate hierarchy with a limited query workload across diverse real-world datasets.

Acknowledgments

Donatella Firmani has been partially supported by the following research grants: SEED PNR Project “FLOWER” “Frontiers in Linking records: knOWledge graphs, Explainability and tempoRal data”, Sapienza Research Project B83C22007180001 “Trustworthy Technologies for Augmenting Knowledge Graphs” and HORIZON Research and Innovation Action 101135576 INTEND “Intent-based data operation in the computing continuum”.

References

[1] J. Huang, Z. Ren, W. X. Zhao, G. He, J.-R. Wen, D. Dong, Taxonomy-aware multi-hop reasoning networks for sequential recommendation, in: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, 2019, pp. 573–581.

[2] Y. Mao, J. Tian, J. Han, X. Ren, Hierarchical text classification with reinforced label assignment, arXiv preprint arXiv:1908.10419 (2019).

[3] J. Wang, C. Kang, Y. Chang, J. Han, A hierarchical dirichlet model for taxonomy expansion for search engines, in: Proceedings of the 23rd international conference on World wide web, 2014, pp. 961–970.

[4] M. A. Hearst, Automatic acquisition of hyponyms from large text corpora, in: Coling 1992 volume 2: The 15th international conference on computational linguistics, 1992.

[5] A. Panchenko, S. Faralli, E. Ruppert, S. Remus, H. Naets, C. Fairon, S. P. Ponzetto, C. Biemann, Taxi at semeval-2016 task 13: a taxonomy induction method based on lexico-syntactic patterns, substrings and focused crawling, in: Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), 2016, pp. 1320–1327.

[6] Y. Mao, T. Zhao, A. Kan, C. Zhang, X. L. Dong, C. Faloutsos, J. Han, Octet: Online catalog taxonomy enrichment with self-supervision, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 2247–2257.

[7] C. Ilvento, Metric learning for individual fairness, arXiv preprint arXiv:1906.00250 (2019).

[8] A. Ukkonen, Crowdsourced correlation clustering with relative distance comparisons, in: 2017 IEEE International Conference on Data Mining (ICDM), IEEE, 2017, pp. 1117–1122.

[9] O. Tamuz, C. Liu, S. Belongie, O. Shamir, A. T. Kalai, Adaptively learning the crowd kernel, in: Proceedings of the 28th International Conference on International Conference on Machine Learning, 2011, pp. 673–680.

[10] M. Hopkins, D. Kane, S. Lovett, G. Mahajan, Noise-tolerant, reliable active classification with comparison queries, arXiv preprint arXiv:2001.05497 (2020).

[11] S. Guo, A. Parameswaran, H. Garcia-Molina, So who won? dynamic max discovery with the crowd, in: Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, 2012, pp. 385–396.

[12] P. Venetis, H. Garcia-Molina, K. Huang, N. Polyzotis, Max algorithms in crowdsourcing environments, in: Proceedings of the 21st international conference on World Wide Web, 2012, pp. 989–998.

[13] S. Davidson, S. Khanna, T. Milo, S. Roy, Top-k and clustering with noisy comparisons, ACM Trans. Database Syst. 39 (2015). URL: https://doi.org/10.1145/2684066. doi:10.1145/2684066.

[14] N. M. Kou, Y. Li, H. Wang, L. H. U, Z. Gong, Crowdsourced top-k queries by confidence-aware pairwise judgments, in: Proceedings of the 2017 ACM SIGMOD International Conference on Management of Data, 2017, pp. 1415–1430.

[15] E. Dushkin, T. Milo, Top-k sorting under partial order information, in: Proceedings of the 2018 ACM SIGMOD International Conference on Management of Data, 2018, pp. 1007–1019.

[16] E. Kazemi, L. Chen, S. Dasgupta, A. Karbasi, Comparison based learning from weak oracles, arXiv preprint arXiv:1802.06942 (2018).

[17] V. Verdugo, Skyline computation with noisy comparisons, in: Combinatorial Algorithms: 31st International Workshop, IWOCA 2020, 2020, p. 289.

[18] Y. Li, J. Li, Y. Suhara, A. Doan, W.-C. Tan, Deep entity matching with pre-trained language models, arXiv preprint arXiv:2004.00584 (2020).

[19] M. Ebraheem, S. Thirumuruganathan, S. Joty, M. Ouzzani, N. Tang, Distributed representations of tuples for entity resolution, PVLDB 11 (2018) 1454–1467.

[20] S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute, V. Raghavendra, Deep learning for entity matching: A design space exploration, in: Proceedings of the 2018 ACM SIGMOD International Conference on Management of Data, 2018, pp. 19–34.

[21] R. Cappuzzo, P. Papotti, S. Thirumuruganathan, Creating embeddings of heterogeneous relational datasets for data integration tasks, Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (2020). URL: http://dx.doi.org/10.1145/3318464.3389742. doi:10.1145/3318464.3389742.

[22] N. Vesdapunt, K. Bellare, N. Dalvi, Crowdsourcing algorithms for entity resolution, PVLDB 7 (2014) 1071–1082.

[23] D. Firmani, B. Saha, D. Srivastava, Online entity resolution using an oracle, PVLDB 9 (2016) 384–395.

[24] S. Galhotra, D. Firmani, B. Saha, D. Srivastava, Robust entity resolution using random graphs, in: Proceedings of the 2018 ACM SIGMOD International Conference on Management of Data, 2018, pp. 3–18.

[25] D. G. Brown, J. Truszkowski, Fast error-tolerant quartet phylogeny algorithms, in: Annual Symposium on Combinatorial Pattern Matching, Springer, 2011, pp. 147–161.

[26] E. Emamjomeh-Zadeh, D. Kempe, Adaptive hierarchical clustering using ordinal queries, in: Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, 2018, pp. 415–429.

[27] C. Zhang, F. Tao, X. Chen, J. Shen, M. Jiang, B. Sadler, M. Vanni, J. Han, Taxogen: Unsupervised topic taxonomy construction by adaptive term embedding and clustering, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 2701–2709.

[28] J. Shen, Z. Wu, D. Lei, C. Zhang, X. Ren, M. T. Vanni, B. M. Sadler, J. Han, Hiexpan: Task-guided taxonomy construction by hierarchical tree expansion, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 2180–2189.

[29] S. Galhotra, D. Firmani, B. Saha, D. Srivastava, Hierarchical entity resolution using an oracle, in: Z. G. Ives, A. Bonifati, A. E. Abbadi (Eds.), SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022, ACM, 2022, pp. 414–428. URL: https://doi.org/10.1145/3514221.3526147. doi:10.1145/3514221.3526147.

[30] G. Papadakis, J. Svirsky, A. Gal, T. Palpanas, Comparative analysis of approximate blocking techniques for entity resolution, PVLDB 9 (2016) 684–695.

[31] S. E. Whang, P. Lofgren, H. Garcia-Molina, Question selection for crowd entity resolution, PVLDB 6 (2013) 349–360.

[32] C. Roquet, S. Lavergne, W. Thuiller, One tree to link them all: a phylogenetic dataset for the european tetrapoda, PLoS currents 6 (2014).

[33] Dmoz https://dmoz-odp.org/, 2018.
[34] Google vision api https://cloud.google.com/vision, 2024.
[35] V. Crescenzi, A. D. Angelis, D. Firmani, M. Mazzei, P. Merialdo, F. Piai, D. Srivastava, Alaska: A flexible benchmark for data integration tasks, 2021. arXiv:2101.11259.
[36] R. He, J. McAuley, Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering, in: proceedings of the 25th international conference on world wide web, 2016, pp. 507–517.
[37] World cities database https://simplemaps.com/data/world-cities, 2024.
[38] Scikit-learn https://scikit-learn.org/stable/modules/clustering.html, 2024.