DAG-Structured Clustering by Nearest Neighbors

Nicholas Monath UMass Amherst
Manzil Zaheer Google
Avinava Dubey Google
Amr Ahmed Google
Andrew McCallum UMass Amherst

Abstract

Hierarchical clusterings compactly encode multiple granularities of clusters within a tree structure. Hierarchies, by definition, fail to capture different flat partitions that are not subsumed in one another. In this paper, we advocate for an alternative structure for representing multiple clusterings, a directed acyclic graph (DAG). By allowing nodes to have multiple parents, DAG structures are not only more flexible than trees, but also allow for points to be members of multiple clusters. We describe a scalable algorithm, Llama, which simply merges nearest neighbor substructures to form a DAG structure. Llama discovers structures that are more accurate than state-of-the-art tree-based techniques while remaining scalable to large-scale clustering benchmarks. Additionally, we support the proposed algorithm with theoretical guarantees on separated data, including types of data that cannot be correctly clustered by tree-based algorithms.

1 Introduction

Hierarchical clusterings are tree-structured nested partitions of a dataset. Each internal node of the tree corresponds to the candidate cluster of its descendant leaf data points. The tree structure encodes multiple alternative granularities of clusterings of the dataset (Heller and Ghahramani, 2005b). This representation of uncertainty has been beneficial to modeling user feedback in entity resolution (Kobren et al., 2019) and selecting flat partitions from trees (Vitale et al., 2019).

The tree structure of these nested clusterings places constraints on what clusters (and thereby clusterings) can be encoded in the tree. In particular, the clusters for which a leaf data point is a member must all exhibit sub/super cluster relationships. This requires that leaves and internal nodes have a single parent in the structure. This disallows leaf data points to simultaneously sit in multiple non-nested clusters.

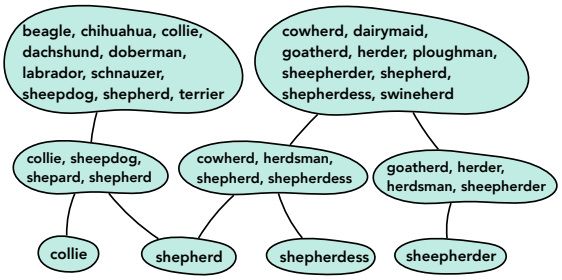

Figure 1: **DAG-Structured Clustering.** A substructure of the clustering produced by our proposed algorithm on a dataset of word vectors. Observe how the word shepherd appears in both the cluster of dog breeds as well as the cluster of farm professions.

The constraint of representing only nested clusters has its limitations (Gama et al., 2017; Jeantet et al., 2020). Representing overlapping (but non-subsuming) clusters may be desirable when the cluster membership of data points may be difficult to determine or when the underlying structure is better described by a *cover* rather than a partition of the data.

Removing the tree-based constraints, we can instead aim to discover *DAG-structured* clusterings (Diday, 1987; Liu et al., 2006; Carlsson et al., 2014; Jeantet et al., 2020, inter alia). A DAG structure is able to express both alternative sub/superset clusters for a point and can better explain data for which points should belong to overlapping clusters. For example, Figure 1 shows a subset of the DAG structure built by our method when clustering word embeddings. The structure can simultaneously represent two senses of the word shepherd (the type of dog and the profession). Such a structure cannot be represented by a tree.

To the best of our knowledge, there does not exist effective algorithms for inferring DAG-structured clusterings at scale. Inferring DAG structures is computationally very challenging. Discovering meaning-

Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego, California, USA. PMLR: Volume 130.

ful DAG-structured clusterings increases the computational burden over the already massive combinatorial search space of hierarchical clusterings.

Previous work on building DAG-structured clusterings (Diday, 1987; Liu et al., 2006, 2007; Jeantet et al., 2020) use sequential bottom-up approaches limited to datasets with a few thousand points, much smaller than datasets on which hierarchical clustering has been applied. Flat structures where points are assigned to multiple clusters is common in approaches such as latent feature models (Griffiths and Ghahramani, 2011) and dictionary learning (Mairal et al., 2009). These approaches are not typically nested, however, and bear different semantics than the hierarchical clustering. Other work has studied the representational capacity of DAG-structured clusterings theoretically (Bertrand and Janowitz, 2002; Gama et al., 2017; Culbertson et al., 2018). For particular families of probabilistic clustering, Greenberg et al. (2018, 2021) use DAG structured clusterings to compactly represent distributions over (and MAP inference for) flat/hierarchical clusterings efficiently via dynamic programming.

Contributions: We present a scalable algorithm for discovering meaningful DAG-structured clusterings on large datasets. The proposed method, Llama, scales to large datasets and is inspired by the classic reciprocal nearest neighbors algorithm (Murtagh, 1983), Llama is a simple round-based algorithm which works by merging together pairs of nearest neighbor nodes. The asymmetry of nearest neighbor relationships leads to the departure from the tree-structure. The proposed method supports parallelism and builds structures that are not exponential in the size of the dataset. We advocate Llama as an effective alternative to hierarchical clustering with both a theoretical and empirical analysis. We provide a theoretical analysis of Llama, demonstrating that it can not only recover the same classes of separated data that scalable hierarchical clustering methods can recover (Monath et al., 2019a), but also recover a less restrictive class of separated data that is not recoverable by these tree-based methods. We provide a comprehensive empirical evaluation that demonstrates that Llama produces higher quality clusterings than state-of-the-art tree and DAG structured methods. We further motivate a series of metrics for evaluating DAG-structured clusterings.

2 Clustering Structures

Before describing our proposed approach, we define the DAG-structured clusterings that we seek to discover in this paper. Let X refer to a dataset of N points $X = \{x_1, \ldots, x_N\}$. A *cover* of X refers to a collection of subsets of X such that their union is equal to X. A clustering or *partition* of X is a set of disjoint subsets that are a cover of X. A hierarchical clustering is:

Definition 1. (Hierarchical Clustering (Krishnamurthy et al., 2012)) *A hierarchical clustering, $\mathcal{T}$, of a dataset X, is a set of clusters where $X \in \mathcal{T}$, and $\forall x \in X, \{x\} \in \mathcal{T}$, and for each $C_i, C_j \in \mathcal{T}$ either $C_i \subset C_j$, $C_j \subset C_i$ or $C_i \cap C_j = \emptyset$. For any cluster $C \in \mathcal{T}$, if $\exists C'$ with $C' \subset C$, then there exists a set $\{C_i\}_{i=1}^k$ of disjoint clusters such that $\bigcup_{i=1}^k C_i = C$.*

The definition of hierarchical clustering is in terms of subsets of X rather than as a discrete data structure with nodes and edges. There is, however, a direct mapping between the discrete data structure and the set-based definition. In the data structure, the nodes correspond to sets in $\mathcal{T}$. A parent-to-child edge exists between $C_\mathtt{p}$ and $C_\mathtt{c}$ if $C_\mathtt{c} \subsetneq C_\mathtt{p}$ and $\nexists C'$ such that $C_\mathtt{c} \subset C' \subset C_\mathtt{p}$. A hierarchical clustering encodes a number of different *tree consistent partitions* (Heller and Ghahramani, 2005b), which are sets of (sub)-tree roots in $\mathcal{T}$ that form flat clusterings.

We are interested in discovering rooted DAGs:

Definition 2. (DAG-Structured Clustering) *A DAG-Structured clustering, $\mathcal{D}$, of a dataset X, is a subset of the powerset of X, $\mathcal{D} \subset \mathcal{P}(X)$ containing at least a root node of the entire dataset $X \in \mathcal{D}$ and the shattered partition $\{\{x\} \mid x \in X\} \subset \mathcal{D}$.*

As in the case of tree structures, we can provide a mapping between a DAG data structure and the nested collection of sets. Like the tree structure case, a parent-to-child edge exists between $C_\mathtt{p}$ and $C_\mathtt{c}$ if $C_\mathtt{c} \subsetneq C_\mathtt{p}$ and $\nexists C'$ such that $C_\mathtt{c} \subset C' \subset C_\mathtt{p}$. In this case, however, a node may have multiple parent nodes, hence calling this a DAG-structured clustering. The set of clusters in a DAG-structured clustering form a partially ordered set, where the containment relationship defines the ordering. The connections to partially ordered sets/Hasse Diagrams have been extensively studied theoretically (Liu et al., 2007; Gama et al., 2017; Carlsson et al., 2014; Culbertson et al., 2018, inter alia).

3 DAG-Structured Clustering

In this section, we describe an agglomerative algorithm for building DAG-structured clusterings. The approach is simple; in each round, nearest neighbor nodes are merged to build additional structure. The asymmetry of nearest neighbor relationships enables the points to sit in multiple clusters simultaneously. The algorithm builds structures that are polynomial in the size of the dataset in the worst case, which is advantageous given that DAG-structured clusterings may have exponentially many nodes.

3.1 Warmup: Reciprocal Nearest Neighbors

Given a dataset X, agglomerative methods work in a sequential round-based fashion, merging together clus-

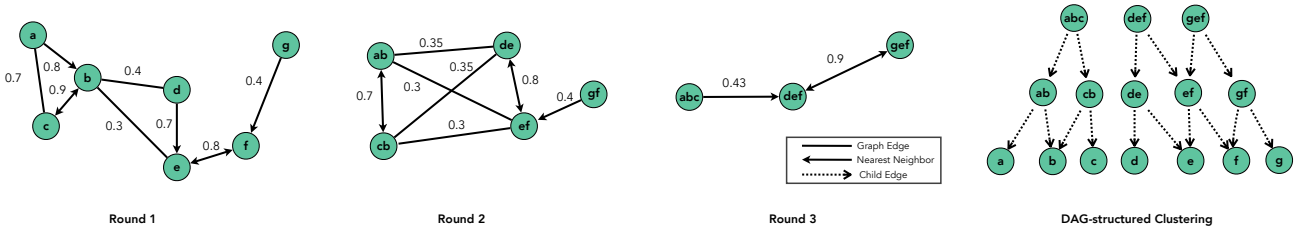

Figure 2: **Llama Algorithm.** An example of the algorithm applied to a toy example graph. The nearest neighbor relationships of each round are shown. The resulting structure is shown on the far right.

ters from the previous round. The initial round of the algorithm begins with each point in a separate cluster $\{\{x\} \mid x \in X\}$. Each round is a flat clustering; $\mathcal{H}^{(i)} = \{C_1, \cdots, C_K\}$ is used to refer to round i.

A *linkage* function, $f : \mathcal{P}(X) \times \mathcal{P}(X) \to \mathbb{R}$, describes the inter-cluster similarities. In the classic hierarchical agglomerative clustering algorithm, each round builds a single new cluster that is the union of the most similar pair of clusters in round $\mathcal{H}^{(i)}$, i.e.,

$$C, C' = \underset{B, B' \in \mathcal{H}^{(i-1)} \times \mathcal{H}^{(i-1)}}{\operatorname{argmax}} f(B, B') \tag{1}$$

$$\mathcal{H}^{(i)} = \left(\mathcal{H}^{(i-1)} \cup \{C \cup C'\}\right) \setminus \{C, C'\}. \tag{2}$$

For convenience, we assume that the self-similarity of a set is the minimum possible value, i.e. $f(C, C) = -\infty$.

The reciprocal nearest neighbor algorithm (Murtagh, 1983) can be more efficient than HAC and for reducible linkage functions produce the same structure. Reciprocal nearest neighbors are defined as two C, C' in the clustering, $\mathcal{H}$, which are more similar to one another than either is any other object in the collection. The algorithm builds the clustering of round $\mathcal{H}^{(i)}$ by merging all pairs of reciprocal nearest neighbors in $\mathcal{H}^{(i-1)}$:

$$\mathcal{R}^{(i)} = \{(C, C') \mid C' = \underset{B \in \mathcal{H}^{(i-1)}}{\operatorname{argmax}} f(C, B) \wedge C = \underset{B \in \mathcal{H}^{(i-1)}}{\operatorname{argmax}} f(C', B)\} \tag{3}$$

$$\mathcal{H}^{(i)} = \left(\mathcal{H}^{(i-1)} \cup \{C \cup C' | (C, C') \in \mathcal{R}^{(i)}\}\right) \setminus \{B | B = C \vee B = C', (C, C') \in \mathcal{R}^{(i)}\}. \tag{4}$$

3.2 Building DAG-Structures

Akin to the reciprocal nearest neighbor algorithm (Murtagh, 1983), we present a simple algorithm for building DAG-structured clusterings. First, we will depart from the hierarchical clustering tradition by removing the assumption that each round of the algorithm will refer to a flat clustering. Instead, each round will represent a *cover* of the dataset. Points may be assigned to multiple clusters[1] in the given round. We overload notation and use $\mathcal{H}^{(i)}$ to refer to the cover produced in round i.

The algorithm begins with each data point in its own cluster. In round i, each cluster C finds its nearest neighbor C' among the members of the cover $\mathcal{H}^{(i-1)}$. These two are merged to form a new node $C \cup C'$ in the following round. This is done for all pairs of nearest neighbors:

$$\mathcal{N}^{(i)} = \{(C, C') \mid C' = \underset{B \in \mathcal{H}^{(i-1)}}{\operatorname{argmax}} f(C, B)\} \tag{5}$$

$$\mathcal{H}^{(i)} = \{C \cup C' | (C, C') \in \mathcal{N}^{(i)}\}. \tag{6}$$

Observe that non-reciprocal nearest neighbor relationships are the source of a cluster having more than one parent in the given round. The cluster of points, C, may be the nearest neighbor of many other clusters C'. Any cluster $C \in \mathcal{H}^{(i-1)}$ may have multiple supersets in $\mathcal{H}^{(i)}$, and that every nearest neighbor cluster C' of $C \in \mathcal{H}^{(i-1)}$ will lead to a unique superset of $C \in \mathcal{H}^{(i)}$. From the data structure point of view, the node corresponding to cluster C will have a parent corresponding to the cluster of $C \cup C'$ for each unique nearest neighbor cluster C'.

We refer to this algorithm as **L**attices by **L**everaging **A**gglomerations and **M**ultiple **A**ncestors (Llama) because of its ability to build clustering structures where points have multiple ancestries. We take the number of rounds used in the algorithm as an optional hyperparamater. Pseudocode is given in Algorithm 1.

The algorithm can be implemented to utilize parallelism. The computation of $\mathcal{N}^{(i)}$ is trivially parallelizable. The computation of $\mathcal{H}^{(i)}$ can also be parallelized using $\mathcal{H}^{(i-1)}$ and $\mathcal{N}^{(i)}$. In Appendix A.1, we discuss how the choice of linkage function impacts scalability.

3.3 Limiting the Size of the DAG-structures

The size of the DAG-structures discovered by Llama will depend on the number of parents of each node.

[1] We use cluster to refer to member sets of a cover.

Algorithm 1 LLAMA

Input: X : dataset, f: set similarity function, L: number of rounds (optional, default ∞)
Output: $\mathcal{D}$: a DAG-structured clustering
$\mathcal{H}_0 \leftarrow \{\{x\} \mid x \in X\}$
$\mathcal{D} \leftarrow \mathcal{H}_0$
for i **from** 1 **to** L **do**
  $\mathcal{N}^{(i)} \leftarrow \{(C, C') | C' = \text{argmax}_{C'' \in \mathcal{H}^{(i-1)}} f(C, C'')\}$
  $\mathcal{H}^{(i)} \leftarrow \{C \cup C' \mid (C, C') \in \mathcal{N}^{(i)}\}$.
  $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{H}^{(i)}$
  if $|\mathcal{H}^{(i)}| = 1$; **return** $\mathcal{D}$
return $\mathcal{D}$

Most simply, the number of parents can be be bounded by an additional hyperparameter in the algorithm, p. We can then adapt the LLAMA algorithm to select at most p parents for each node. We propose to do this by finding the top p entries in $\mathcal{N}^{(i)}$ for C, according to $f(\cdot, \cdot)$, we refer to this set as P_C. We will then restrict the entries in $\mathcal{N}^{(i)}$ to be those tuples (C, C') such that $(C, C') \in \mathsf{P}_C$ and $(C, C') \in \mathsf{P}_{C'}$ (in both of the clusters' top p lists). Let $\mathcal{N}_C^{(i)}$ be the entries in $\mathcal{N}^{(i)}$ that contain C, then in our set-based notation, this is:

$$\mathsf{P}_C = \underset{(B,B') \in \mathcal{N}_C^{(i)}}{\text{argtopk}} \; f(B, B') \tag{7}$$

$$\mathcal{N}^{(i)'} \leftarrow \{(C, C') \mid (C, C') \in \mathsf{P}_C \cap \mathsf{P}_{C'}\} \tag{8}$$

We then update $\mathcal{H}^{(i)}$ to include both the new sets from $\mathcal{N}^{(i)'}$ as well as singleton clusters that were assigned no parent:

$$\mathcal{H}^{(i)'} = \{C \cup C' \mid (C, C') \in \mathcal{N}^{(i)'}\} \cup \{C | \mathsf{P}_C \cap \mathcal{N}^{(i)'} = \emptyset\} \tag{9}$$

4 Theoretical Analysis

We would like to understand what kinds of data LLAMA will be effective at clustering. We prove that, under certain separation assumptions, LLAMA is able to recover the ground truth clustering as is often done in the clustering literature (Balcan et al., 2014; Kushagra et al., 2016; Bachem et al., 2015; Kobren et al., 2017, inter alia). While data in practice may not satisfy these separation assumptions, we believe that this analysis supports our empirical evaluation by: (1) proving our algorithm will work as expected on data for which clusters are clearly defined, (2) providing insights into the representational capacity of the model, which can help understand its behavior.

In this section, we show that LLAMA is able to accurately recover the target partition when data follows a particular generalization of strictly separated data, *model-based separated data* (Monath et al., 2019a). We then describe a related class of separated data, less restrictive than model-based separated. We demonstrate that LLAMA is able to recover the target partition while tree-based methods such as HAC and Grinch (Monath et al., 2019a) cannot.

4.1 Model-based Separation

The *model-based separation* condition defines the behavior of a linkage function f with respect to dataset X and a ground truth partition of X, denoted $\mathcal{H}^\star$. Model-based separation defines f's behavior in terms of a latent graph structure. This latent graph is unobserved at the time of clustering (i.e., the input to the clustering problem is X, not this graph structure). This graph $G = (X, E)$ has one vertex per datapoint in X and edges are defined such that the connected components of G are exactly the clusters of $\mathcal{H}^\star$.

Intuitively, model-based separation says that the linkage function similarity between two sets of points C_0 and C_1 that are connected in G is higher than C_0 or C_1's similarity with any other set C_2 which it is not connected to. Formally, the condition is as follows:

Definition 3. (Model-based Separation (Monath et al., 2019a)) *Let $G = (X, E)$ be a graph. Let $f : \mathcal{P}(X) \times \mathcal{P}(X) \rightarrow \mathbb{R}^+$ be a symmetric linkage function that computes the similarity of two groups of vertices and let $g : \mathcal{P}(X) \times \mathcal{P}(X) \rightarrow \{0, 1\}$ be a function that returns 1 if the union of its arguments is a connected subgraph of G. Then f separates G if $\forall C_0, \; C_1, \; C_2 \subseteq X, \; g(C_0, C_1) > g(C_0, C_2) \implies f(C_0, C_1) > f(C_0, C_2)$. The target partition, $\mathcal{H}^\star$, which is model-based separated, corresponds to connected components in G.*

Observe that strictly separated data (i.e., data for which all within cluster similarities are more than across cluster similarities) (Balcan et al., 2008) is a special case of model-based separation. Strictly separated data corresponds to the case where each cluster in the underlying latent graph is a clique. Similarly, tree or chain structured clusters can be described if each cluster's connected component in the underlying latent graph is a tree or chain.

We are interested in understanding whether a target partition $\mathcal{H}^\star$ is contained in the hierarchical / DAG-structured clusterings produced by an algorithm. Hierarchical agglomerative clustering and GRINCH (Monath et al., 2019a) both produce trees that contain the target partition as a tree consistent partition. We now show that LLAMA is also able to recover the target partition as a DAG-consistent partition.

To show that LLAMA can recover DAG structures that contain the target model-based separated partition $\mathcal{H}^\star$, we make the following observation about the pairs of nearest neighbors that are merged in each round:

Lemma 1. *Given a dataset X and a symmetric link-*

age function f such that X is model-based separated with respect to f, let $\mathcal{H}^\star$ be the target partition corresponding to the separated data. In each round of Llama*, each pair of nearest neighbors $(C, C') \in \mathcal{N}^{(i)}$, will satisfy:*

$$\exists C^* \in \mathcal{H}^* \text{ s.t. } C \cup C' \subseteq C^*, C^* \subseteq C, \text{ or } C^* \subseteq C'.$$

Please see Appendix §B.1 for the proof.

Theorem 1. *Given a dataset X and a symmetric linkage function f such that X is model-based separated with respect to f, let $\mathcal{H}^\star$ be the target partition corresponding to the separated data. Let $\mathcal{D}$ be the DAG-structured clustering produced by* Llama *(Alg. 1), then $\mathcal{H}^\star$ is a $\mathcal{D}$ consistent partition, $\mathcal{H}^\star \subset \mathcal{D}$.*

Proof Sketch. Lemma 1 indicates that points from the same ground truth cluster in the target partition will be merged together until there are no more such points to be merged. We only extend to cross-cluster merging (the latter two conditions of Lemma 1) when we have contained a full ground-truth cluster in the DAG. After some number of rounds, we show that this must happen for each cluster in the target partition. A complete proof is in the Appendix §B.1.

We have now seen that Llama is at least as expressive as tree-based methods for clustering model-based separated data. Now we turn to a data separation assumption that is not recovered by tree-based methods.

4.2 Noisy Model-based Separation

While model-based separation allows some additional flexibility compared to strict separation, it is overly rigid in its assumption that *every* point in a cluster has some point in their cluster that is closer than every point outside of their cluster. In this section, we propose a loosening of this restriction to allow *some* points to have nearest neighbors outside their clusters, which we refer to as *noisy model-based separation*.

Intuitively, noisy model-based separation says that a point x in a ground truth cluster $C^\star$ may have a nearest neighbor x' such that x' is not in $C^\star$, provided that there exists another point $x'' \in C^\star$ whose nearest neighbor is x. Formally, we need to make an additional restriction on the linkages that separate this data:

Definition 4. (Noisy Model-based Separation) *Let $G = (X, E)$ be a graph with connected components $\mathcal{H}^\star$. Let $f : \mathcal{P}(X) \times \mathcal{P}(X) \to \mathbb{R}^+$ be a symmetric linkage function that computes the similarity of two groups of vertices. Let $g : \mathcal{P}(X) \times \mathcal{P}(X) \to \{0, 1\}$ be a function that returns 1 if the union of its arguments is a connected subgraph of G. The function f separates G if $\forall C_0,\ C_1,\ C_2 \subseteq X$ either:*

- $g(C_0, C_1) > g(C_0, C_2) \implies f(C_0, C_1) > f(C_0, C_2)$; *or*
- $|C_0| = |C_1| = |C_2| = 1$, $g(C_0, C_1) > g(C_0, C_2)$, $f(C_0, C_2) > f(C_0, C_1)$, $\exists x \in X$ *s.t.* $g(C_0, \{x\}) = 1$ *and* $C_0 = \text{argmax}_{x' \in X} f(\{x\}, \{x'\})$

We now prove that Llama recovers a DAG-structure with the noisy model-based separated partition and that tree-based methods cannot recover the target partition for some such datasets.

Proposition 1. *Given a dataset X and a symmetric linkage function f such that X is noisy model-based separated with respect to f, let $\mathcal{H}^\star$ be the target partition corresponding to the noisy model-based separated data. Let $\mathcal{D}$ be the DAG-structured clustering produced by* Llama *(Alg. 1), then $\mathcal{H}^\star$ is a $\mathcal{D}$ consistent partition, $\mathcal{H}^\star \subset \mathcal{D}$.*

Proposition 2. *There exists a datasets X and symmetric linkage function f such that X is noisy model-based separated wrt f, let $\mathcal{H}^\star$ be the target partition corresponding to the noisy model-based separated data. HAC and* Grinch *produces a structure $\mathcal{T}$ such that $\mathcal{H}^\star$ is not a tree consistent partition, $\mathcal{H}^\star \not\subset \mathcal{T}$.*

Please see Appendix §B.2 for more detail.

Finally, we note that less restrictive separation models have been considered (Balcan et al., 2014). This work shows that a *particular* linkage function can be used to discover tree structures that contain the target partition. Model-based separation analysis, on the other hand, presumes a linkage function with the particular properties is given. Future work might consider if *particular* linkage functions that can be used with DAG-structured clustering algorithms for broader, less-rigid, data separation assumptions.

4.3 Complexity

We analyze the space and time complexity of Llama. See Appendix §B.3 for proofs for each statement.

Proposition 3. (Space Complexity). *Given a dataset of N points,* Llama *produces DAG-structured clusterings with at most $O(N^2)$ nodes.*

Proposition 4. (Time Complexity). *Given a dataset of N points, R rounds of* Llama *requires at most $O(R * N^2)$ linkage function computations.*

Proposition 5. (Number of Rounds). *Let $\mathcal{H}^\star$ be the target partition of a dataset that is (noisy) model-based separated, let K be the size of the largest cluster in $\mathcal{H}^\star$. $K = \max_{C \in \mathcal{H}^\star} |C|$. After K rounds,* Llama *produces a structure that contains $\mathcal{H}^\star$.*

5 Evaluation Measures of DAG-structured Clusterings

Hierarchical clusterings are often evaluated on datasets for which there exists a ground truth *flat* cluster-

ing. Evaluation measures such as dendrogram purity (Heller and Ghahramani, 2005b) are often used. Dendrogram purity, which is the average purity of the least common ancestor of pairs of same cluster points, does not translate well to DAG-structured clusterings. In particular, there exist trivial DAGs that produce perfect dendrogram purity scores (see Appendix §C.1).

To the best of our knowledge, there are not well established metrics to measure the quality of DAG-structured clusterings. We describe and motivate metrics here that can be used for both tree and DAG-structured clustering. Further, we note that these metrics can be used in the case that there exists a ground truth *partition* of the data as well as in the case where there is a ground truth *cover* of the data.

5.1 Recall-Focused Metrics

First, we design a metric that mimics recall metrics in information retrieval. We measure whether each ground truth cluster is faithfully represented in a tree or DAG-structured clustering using Jaccard similarity. The Jaccard similarity is defined between a ground truth cluster $C^\star$ and predicted cluster $\hat{C}$: $\mathsf{Jacc}(C^\star, \hat{C}) = \frac{|C^\star \cap \hat{C}|}{|C^\star \cup \hat{C}|}$.

Mean Jaccard Per Label We measure the mean over ground truth cluster (or cover) labels of the maximum Jaccard similarity of the ground truth clusters with the predicted structure. This metric was also proposed by (Liu et al., 2007).

$$\mathsf{Jacc/lbl}(\mathcal{D}, \mathcal{H}^\star) = \frac{1}{|\mathcal{H}^\star|} \sum_{C^\star \in \mathcal{H}^\star} \max_{\hat{C} \in \mathcal{D}} \mathsf{Jacc}(C^\star, \hat{C}) \quad (10)$$

Mean Jaccard Per Point We also compute the mean of the Jaccard similarity over the ground truth points for the highest scoring predicted clustering in the DAG:

$$\mathsf{Jacc/pt}(X, \mathcal{D}, \mathcal{H}^\star) = \frac{1}{Z} \sum_{x \in X} \sum_{C^\star \in \mathcal{H}^\star_x} \max_{\hat{C} \in \mathcal{D}} \mathsf{Jacc}(C^\star, \hat{C}) \quad (11)$$

where $Z = \sum_{x \in X} |\mathcal{H}^\star_x|$ and where $\mathcal{H}^\star_x$ are the ground truth cluster assignments of x. Observe that each metric obtains a value of 1 if and only if each of the ground truth clusters are represented predicted structure.

5.2 Precision-Focused Metrics

The recall-focused metrics are not enough to measure the quality of a DAG or tree structure. A DAG structure that contains the full powerset $\mathcal{P}(X)$ of a dataset would be unmanagably large and contain a multitude of potentially irrelevant substructure. Yet this would achieve a perfect score in terms of the recall-focused metrics. And so, we introduce a metric that is focused on the quality of each node in the predicted structures. We encourage structures to be as precise as possible.

Mean Jaccard Per Node We measure mean of the Jaccard similarity of each node with its best aligned ground truth cluster.

$$\mathsf{Jacc/node}(\mathcal{D}, \mathcal{H}^\star) = \frac{1}{|\mathcal{D}|} \sum_{\hat{C} \in \mathcal{D}} \max_{C^\star \in \mathcal{H}^\star} \mathsf{Jacc}(C^\star, \hat{C}) \quad (12)$$

6 Experiments

We compare the performance of Llama to state-of-the-art methods for hierarchical and DAG-structured clusterings. We evaluate the effectiveness of each at recovering ground-truth labeled data. We further attempt to automatically reconstruct the DAG-structure WordNet (Miller, 1995) from vector representations of words using Llama.

6.1 Clustering Benchmarks

First, we consider datasets where each point is assigned one ground truth cluster. In this experiment, we hope to understand if the clusters for each point that are discovered by Llama are better aligned with the underlying data than those of competing methods.

Following previous work (Kobren et al., 2017), we run experiments on publicly available large scale hierarchical clustering benchmark datasets. We evaluate on the following datasets: **Speaker**, feature vectors representing audio signals of spoken voices from different speakers (each speaker is a ground truth cluster) (Greenberg et al., 2014); **ALOI** (Amsterdam Library of Object Images), histogram features of toy objects (Geusebroek et al., 2005); **ILSVRC (Sm.) (50K subset)** and **ILSVRC (Lg.) (1.2M Images)** Inception embeddings from the ImageNet ILSVRC 2012 dataset (Russakovsky et al., 2015); **Imagenet** a sample of 100k images from all 17K classes present in ImageNet. See Appendix §C.2 for additional details.

We evaluate against the following tree-based methods: **Affinity clustering (Aff.)** (Bateni et al., 2017), a round-based bottom-up hierarchical clustering method that connects each point to its nearest neighbor in a single round and builds nodes in a tree based on connected components in this 1-nearest neighbor graph; **Grinch** (Monath et al., 2019a), an online hierarchical clustering method that performs tree re-arrangements after each point is inserted; **Reciprocal Nearest Neighbors (RcNN)** (Murtagh, 1983), the classic agglomerative algorithm described in Section 3.1.

Each dataset uses cosine similarity. Llama, RcNN, and Affinity all make use of linkage functions that use k-nearest neighbor graph sparsification. This technique precomputes a k-NN graph over the dataset so as

		Llama		RcNN		Affinity		Grinch
		Sing.	Avg.	Sing.	Avg.	Sing.	Avg.	
Jacc/node	**ALOI**	0.067	**0.117**	0.068	0.037	0.027	0.027	0.052
	ILSVRC (Sm.)	0.154	**0.284**	0.146	0.076	0.044	0.047	0.171
	Speaker	0.271	**0.329**	0.231	0.227	0.175	0.177	0.257
	ImageNet	0.154	0.154	**0.178**	0.173	0.171	0.169	0.372
	ILSVRC (Lg.)	0.023	**0.023**	-	0.005	0.002	0.003	-
Jacc/pt	**ALOI**	**0.700**	0.560	0.594	0.593	0.648	0.518	0.509
	ILSVRC (Sm.)	0.559	**0.655**	0.393	0.626	0.537	0.555	0.575
	Speaker	0.485	**0.582**	0.467	0.563	0.430	0.447	0.564
	ImageNet	0.219	**0.219**	0.201	0.218	0.199	0.199	0.208
	ILSVRC (Lg.)	0.540	0.604	-	**0.621**	0.546	0.530	-
Jacc/lbl	**ALOI**	**0.759**	0.647	0.704	0.669	0.713	0.605	0.615
	ILSVRC (Sm.)	0.638	**0.728**	0.528	0.707	0.617	0.638	0.661
	Speaker	0.665	**0.726**	0.659	0.713	0.607	0.615	0.711
	ImageNet	0.390	**0.399**	0.366	0.384	0.360	0.360	0.372
	ILSVRC (Lg.)	0.623	0.677	-	**0.702**	0.625	0.615	-

Table 1: **Clustering Benchmarks**. *Precision metric* is Jacc/node and *Recall metrics* are Jacc/pt and Jacc/lbl.

		Llama	Llama	RcNN	OHC
		Avg. link.	Approx.	Exact.	
Jacc/node	**ALOI**	**0.152**	0.100	0.044	0.091
	ILSVRC	**0.346**	0.2647	0.090	0.197
	Speaker	0.405	**0.408**	0.271	0.349
	ImageNet	0.275	**0.324**	0.280	0.268
Jacc/pt	**ALOI**	0.984	0.979	0.950	**0.990**
	ILSVRC	**0.975**	0.973	0.976	0.924
	Speaker	0.804	**0.832**	0.813	0.827
	ImageNet	0.593	**0.604**	0.583	0.567
Jacc/lbl	**ALOI**	0.897	0.892	0.880	**0.908**
	ILSVRC	**0.936**	0.935	**0.936**	0.904
	Speaker	0.844	**0.860**	0.853	0.843
	ImageNet	0.688	**0.698**	0.690	0.690

Table 2: **Comparison to OHC**. We sample datasets of 1000 points and report results with average linkage. Aff. and Grinch are outperformed by other methods. We also compare the two variants of average linkage with Llama (§A.1).

to make the argmax operations in the algorithm more efficient (§A.1). For RcNN, and Aff. we report results with both single and average linkage. For Llama, we use single and a approximation of average linkage that supports a more efficient implementation (§A.1). For Grinch, which does not use k-NN graph sparsification, we use its most efficient (and best performing) implementation that uses a centroid-based linkage.

Table 1 shows the results for this experiment. We observe that Llama outperforms the other methods in all but three of the dataset/metric combinations. We hypothesize that the improvements observed by Llama are due to the DAG structure's flexibility in representing alternative clusterings. Importantly,

		Llama	RcNN	Aff.	Grinch
Jacc/node	**EURLex-4k**	**0.182**	0.156	0.142	0.111
	Bibtex	**0.081**	0.025	0.017	0.024
	Wiki10-31K	0.298	0.404	**0.411**	–
	Delicious	**0.068**	0.027	0.020	0.026
	MediaMill	**0.009**	0.004	0.003	–
Jacc/pt	**EURLex-4k**	0.172	**0.180**	0.143	0.061
	Bibtex	0.178	**0.198**	0.166	0.067
	Wiki10-31K	**0.184**	0.104	0.168	–
	Delicious	0.108	**0.129**	0.124	0.109
	MediaMill	0.335	**0.342**	0.339	–
Jacc/lbl	**EURLex-4k**	**0.466**	0.455	0.424	0.332
	Bibtex	0.172	**0.179**	0.138	0.089
	Wiki10-31K	**0.415**	0.392	0.361	–
	Delicious	0.090	**0.091**	0.076	0.056
	MediaMill	**0.099**	0.096	0.089	–

Table 3: **Covering Benchmarks**. The datasets for which Grinch did not finish are marked with dashes. All methods use average linkage.

Llama performs better on both the precision-based (node) and recall-based (point/label) metrics. This indicates that the structures discovered by the method include, on average, nodes that are better aligned with the ground truth clustering (recall) and fewer spurious nodes that do not have significance with respect to the underlying data (precision). The dashed cells indicate the algorithm exceeded our 10hr, 150GB RAM limit.

To compare with bottom-up DAG-structured clustering algorithms that operate in a sequential fashion, we compare to **Overlapping Hierarchical Clustering (OHC)** (Jeantet et al., 2020). OHC is a DAG-structured clustering method that considers agglomerations, like HAC, one edge at a time and uses a distance threshold to determine whether a node should participate in multiple agglomerations. We could not

Leaf Node	Ancestors Discovered by Llama
blossoming	{blossom, blossoming}, {budding, blossoming}, {budding, emergent, emerging, fledgling, incipient, nascent, blossoming}, {abloom, blooming, flowered, flowering, bloom, bloomer, bloomers, blossom, blossoming}, {abloom, autumn-flowering, blooming, early-blooming, early-flowering, fall-blooming, flowered, flowering, half-hardy, late-blooming, late-flowering, planted, seeded, sown, spring-blooming, sprouted, summer-bloom}
disloyal	{disloyal, allegiance, disloyalty, loyalty}, {anti-american, disloyal, pro-american, seditious, traitorous, treasonable, treasonous, un-american, unpatriotic, collaborationist, disloyalty, incitement, quisling, sedition, traitor, treason, treasonist, turncoat }, {adulterous, disloyal, faithless, unfaithful, adulterer, adultery, allegiance, commitment, dedication, devotedness, devotion, disloyalty, faithfulness, faithlessness, fealty, fidelity, fornication, infidelity, loyalty, unfaithful}

Table 4: **WordNet Clusters** Sample nodes from the DAG-structured clustering. We observe that the algorithm discovers interesting overlapping components clusters with different lineages of words revealing multiple senses.

get results for OHC on the above datasets in the 10 hours/dataset we allot to each method as these are much larger than the ones used by in the original paper. To provide a comparison to OHC, we compare the methods on a random subset of 1000 points and evaluate the methods on these subsets. We run a hyperparameter sweep over the parameters of OHC (merging criterion and batch size) and report the best performing OHC result for each dataset in Table 2.

In the experiments, we use 50 rounds for Llama and restrict the number of parents to be 5. RcNN needs around 100 rounds for convergence on all except ILSVRC (Lg.) needing 200 rounds. We analyze two hyperparameters of Llama in Fig. 3. We plot the accuracy performance as a function of the number of neighbors in the k-NN sparsification and the number of rounds used. We perform additional analysis of the hyperparameters in §C.3 of the Appendix.

6.2 Covering Benchmarks

Next, we take extreme multi-label classification benchmark datasets for which the ground truth is a *cover* rather than a partition (See §C.2): **MediaMill**, **Delicious**, **BiBTeX**, **EURLex-4k**, **Wiki10-31K**.

We compare to the same set of algorithms as used in Section 6.1. We use the same Jaccard-based metrics as before since these metrics can be applied to both partition and cover-based labelings of data. We use the same experimental settings that are used for the partition-based benchmarks. Table 3 provides the results for this experiment. We observe that our proposed method either outperforms or is competitive with tree-based metrics on all datasets/metrics.

	Llama -50	Llama -5	RcNN	Aff.	Grinch
Jacc/node	0.307	0.474	0.571	**0.667**	0.532
Jacc/pt	**0.714**	**0.714**	0.695	0.645	0.664
Jacc/lbl	**0.869**	**0.869**	0.857	0.823	0.839

Table 5: **WordNet** Reconstruction evaluation metrics.

6.3 WordNet Reconstruction

We perform analysis on the task of automatically building lexical resources. WordNet (Miller, 1995) is a manually curated resource that records, among other information, *synsets*, sets of English words that are synonymous. Words may be polysemous (have multiple senses) and so the same word spelling may exist in two synsets. We attempt to recover these synsets from the vector data using Llama and other approaches.

We select the subset of WordNet for which the word type has a representation in the fasttext model (leaving 64K words) (Mikolov et al., 2018). We again use average linkage with cosine similarities as in the prior experiments. Each word has a single embedding for its spelling. This means that both senses of the word *crane* are represented by the same point. We take the synset labels of these words as the ground truth labels.

Table 5 provides the quantitative results. We show results for two variants of Llama, one with 50 rounds and another with 5 rounds. We hypothesized that the structure of the synsets is relatively fine grained and so introducing additional rounds of the algorithm that adds larger nodes is the reason for the decrease in the precision-based mean Jaccard per node metric.

Despite each word having a single point representation, we are able to discover alternative senses of various words using Llama (Figure 1 and Table 4).

7 Related Work

DAGs. The discovery of DAG-structured clusterings has been considered by previous work as PoClustering (Liu et al., 2006, 2007) (poset clustering), overlapping hierarchical clustering (OHC) (Jeantet et al., 2020), and others (Jardine and Sibson, 1971; Diday, 1987; Bertrand and Janowitz, 2002; Kramer et al., 2014; Carlsson et al., 2014; Bertrand and Diatta, 2017; Gama et al., 2017; Culbertson et al., 2018; Mémoli and Okutan, 2020, inter alia). In our empirical comparison, we compare to OHC which shares a similar structure to methods such as PoCluster (Liu et al., 2006, 2007) and CLIXO (Kramer et al., 2014), in their sequential

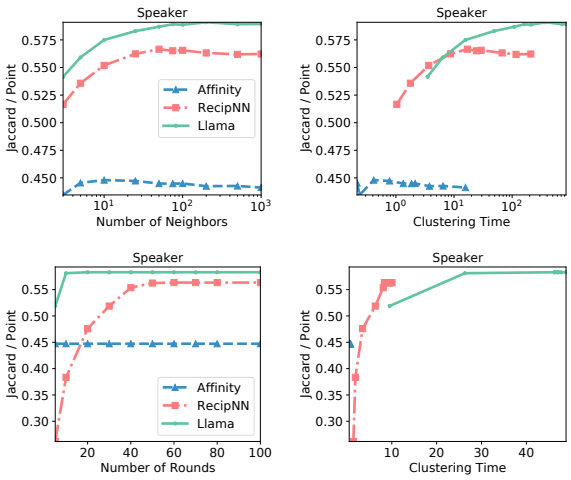

Figure 3: **k-NN Sparsification & Num Rounds**. We plot the Jacc/pt for each method as a function of the k-NN and the number of rounds hyperparameters.

consideration of ordered pairwise similarities. Pyramidal clustering (Diday, 1987; Bertrand and Janowitz, 2002) represent a special case of DAG-structured clustering where nodes have at at most two parents. In the same way the relationship between ultrametrics and tree structures has been explored (Ailon and Charikar, 2005; Carlsson and Mémoli, 2010; Cohen-Addad et al., 2020), theoretical work has considered the relationship between different kinds of metrics and DAG-structured clusterings as well as more general representational capacity considerations (Bertrand and Janowitz, 2002; Carlsson et al., 2014; Gama et al., 2017; Culbertson et al., 2018; Mémoli and Okutan, 2020).

Feature Models & Grouped Data. Mixed membership models such as sparse dictionary learning (Mairal et al., 2009, inter alia) and latent feature models (Griffiths and Ghahramani, 2011, inter alia) produce an assignment of points to overlapping clusters. These approaches typically attempt to reconstruct a data matrix and use the overlapping clusters to capture different components of each data point. Our work, on the other hand, represents alternative clusters, where each point is an equal member of its clusters. Our work differs from topic models and related models that build tree and DAG structures (Paisley et al., 2014; Zhang and Paisley, 2015, inter alia) in that we do not operate on grouped data.

Graph-based methods. Ego-splitting methods (Epasto et al., 2017) operate on graph-based data, and create duplicate copies of certain nodes in the graph, allowing data to be simultaneously attributed to multiple clusters, however these approaches are limited to flat structures. Other work has considered discovering clusters in asymmetric graphs (Carlsson et al., 2014; Vasiliauskaite and Evans, 2020).

Multiple Alternative Clusterings. Other work has attempted to discover for a given datasets, several distinct, but meaningful partitions. (Wu et al., 2018; Qi and Davidson, 2009; Jain et al., 2008; Niu et al., 2010).

Gradient-based Methods. Recent work has explored discovering DAG structures via gradient descent (Zheng et al., 2018, 2020). Other work has considered continuous representations of hierarchical clusterings with objectives optimized by gradient descent (Monath et al., 2019b; Chami et al., 2020).

Geometric embeddings. Cones (Vendrov et al., 2015; Lai and Hockenmaier, 2017; Ganea et al., 2018), hyperbolic (Nickel and Kiela, 2017, 2018; Law et al., 2019), discs (Suzuki et al., 2019), and box (Vilnis et al., 2018; Dasgupta et al., 2020) embeddings been shown to be used to be effective at representing partially ordered sets, DAGs, and trees. Directly representing every member of the powerset of a dataset using these methods is computationally infeasible and so would not be a reasonable alternative. Instead, we might consider using union/intersection operations to represent clusters while only parameterizing the base elements.

Hierarchies. Hierarchical clustering is widely studied theoretically (Balcan et al., 2014; Chaudhuri et al., 2014; Dasgupta, 2016, inter alia), empirically (Zhang et al., 1997; Rao et al., 2010; Kobren et al., 2017, inter alia) and from scalability aspects (Olson, 1995; Garg et al., 2006; Jin et al., 2013; Dubey et al., 2014; Hu et al., 2015; Jin et al., 2015; Bateni et al., 2017; Yaroslavtsev and Vadapalli, 2018; Moseley et al., 2019; Santos et al., 2019; Dubey et al., 2020). These related works define cost functions, data models, and approximation algorithms for discovering meaningful structures. Apart from the methods mentioned in this paper, we refer readers to hashing-based approaches (Aboud et al., 2019) and randomized approaches (Heller and Ghahramani, 2005a). Our goal in this paper was to show the applicability of DAG-based methods to settings where tree-based method are typically used.

8 Conclusion

In this paper, we present an algorithm for building DAG-structured clusterings, Llama, as an alternative to hierarchical clustering. We show that Llama can discover higher quality structures than state-of-the-art tree and DAG-based alternatives. We evaluate on clustering benchmarks (of around 1M points) demonstrating Llama can be run at the same scale as tree-based methods. We additionally provide a theoretical analysis that shows there exist classes of separable data for which Llama can recover the ground truth clustering while tree-based alternatives cannot. We hope that this paper can lead to future work to considering DAG-structured clustering work as an alternative to tree-based methods.

Acknowledgements

We thank Michael Boratko & Javier Burroni, for their detailed and thoughtful suggestions. We also thank Rajarshi Das, Ari Kobren, Craig Greenberg, Sebastian Macaluso, Rico Angell, and Nishant Yadav for their feedback and relevant discussions of clustering and DAG structures. We thank the anonymous reviewers for their helpful comments and suggestions. Initial work began when Nicholas Monath was an intern at Google. Andrew McCallum and Nicholas Monath are supported in part by the Center for Data Science and the Center for Intelligent Information Retrieval, and in part by the National Science Foundation under Grant No. NSF-1763618. The work reported here was performed in part by the Center for Data Science and the Center for Intelligent Information Retrieval, and in part using high performance computing equipment obtained under a grant from the Collaborative R&D Fund managed by the Massachusetts Technology Collaborative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

References

A. Abboud, V. Cohen-Addad, and H. Houdrougé. Subquadratic high-dimensional hierarchical clustering. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

N. Ailon and M. Charikar. Fitting tree metrics: Hierarchical clustering and phylogeny. *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, 2005.

O. Bachem, M. Lucic, and A. Krause. Coresets for nonparametric estimation-the case of dp-means. *International Conference on Machine Learning (ICML)*, 2015.

M.-F. Balcan, A. Blum, and S. Vempala. A discriminative framework for clustering via similarity functions. *Symposium on Theory of Computing (STOC)*, 2008.

M.-F. Balcan, Y. Liang, and P. Gupta. Robust hierarchical clustering. *The Journal of Machine Learning Research (JMLR)*, 2014.

M. Bateni, S. Behnezhad, M. Derakhshan, M. Hajiaghayi, R. Kiveris, S. Lattanzi, and V. Mirrokni. Affinity clustering: Hierarchical clustering at scale. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

P. Bertrand and J. Diatta. Multilevel clustering models and interval convexities. *Discrete Applied Mathematics*, 2017.

P. Bertrand and M. F. Janowitz. Pyramids and weak hierarchies in the ordinal model for clustering. *Discrete Applied Mathematics*, 2002.

G. Carlsson, F. Mémoli, A. Ribeiro, and S. Segarra. Hierarchical Quasi-Clustering methods for asymmetric networks. *International Conference on Machine Learning (ICML)*, 2014.

G. E. Carlsson and F. Mémoli. Characterization, stability and convergence of hierarchical clustering methods. *Journal of Machine Learning Research (JMLR)*, 2010.

I. Chami, A. Gu, V. Chatziafratis, and C. Ré. From trees to continuous embeddings and back: Hyperbolic hierarchical clustering. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

K. Chaudhuri, S. Dasgupta, S. Kpotufe, and U. von Luxburg. Consistent procedures for cluster tree estimation and pruning. *IEEE Transactions on Information Theory*, 2014.

V. Cohen-Addad, C. Karthik, and G. Lagarde. On efficient low distortion ultrametric embedding. *International Conference on Machine Learning (ICML)*, 2020.

J. Culbertson, D. P. Guralnik, and P. F. Stiller. Functorial hierarchical clustering with overlaps. *Discrete Applied Mathematics*, 236:108–123, 2018.

S. Dasgupta. A cost function for similarity-based hierarchical clustering. *Symposium on Theory of Computing (STOC)*, 2016.

S. S. Dasgupta, M. Boratko, D. Zhang, L. Vilnis, X. L. Li, and A. McCallum. Improving local identifiability in probabilistic box embeddings. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

E. Diday. *Orders and overlapping clusters by pyramids.* PhD thesis, INRIA, 1987.

A. Dubey, Q. Ho, S. Williamson, and E. P. Xing. Dependent nonparametric trees for dynamic hierarchical clustering. *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

A. Dubey, M. M. Zhang, E. P. Xing, and S. A. Williamson. Distributed, partially collapsed mcmc for bayesian nonparametrics. *Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.

A. Epasto, S. Lattanzi, and R. Paes Leme. Ego-splitting framework: From non-overlapping to overlapping clusters. *Conference on Knowledge Discovery and Data Mining (KDD)*, 2017.

F. Gama, S. Segarra, and A. Ribeiro. Hierarchical overlapping clustering of network data using cut metrics. *IEEE Transactions on Signal and Information Processing over Networks*, 2017.

O.-E. Ganea, G. Bécigneul, and T. Hofmann. Hyperbolic entailment cones for learning hierarchical embeddings. *International Conference on Machine Learning (ICML)*, 2018.

A. Garg, A. Mangla, N. Gupta, and V. Bhatnagar. Pbirch: A scalable parallel clustering algorithm for incremental data. *International Database Engineering and Applications Symposium (IDEAS)*, 2006.

J.-M. Geusebroek, G. J. Burghouts, and A. W. M. Smeulders. The amsterdam library of object images. *International Journal of Computer Vision (IJCV)*, 2005.

C. Greenberg, N. Monath, A. Kobren, P. Flaherty, A. McGregor, and A. McCallum. Compact representation of uncertainty in clustering. 2018.

C. S. Greenberg, D. Bansé, G. R. Doddington, D. Garcia-Romero, J. J. Godfrey, T. Kinnunen, A. F. Martin, A. McCree, M. Przybocki, and D. A. Reynolds. The nist 2014 speaker recognition i-vector machine learning challenge. *Odyssey: The Speaker and Language Recognition Workshop*, 2014.

C. S. Greenberg, S. Macaluso, N. Monath, J.-A. Lee, P. Flaherty, K. Cranmer, A. McGregor, and A. McCallum. Cluster trellis: Data structures & algorithms for exact inference in hierarchical clustering. *Conference on Artificial Intelligence and Statistics (AISTATS)*, 2021.

T. L. Griffiths and Z. Ghahramani. The indian buffet process: An introduction and review. *Journal of Machine Learning Research (JMLR)*, 2011.

R. Guo, P. Sun, E. Lindgren, Q. Geng, D. Simcha, F. Chern, and S. Kumar. Accelerating large-scale inference with anisotropic vector quantization. In *International Conference on Machine Learning (ICML)*, 2020.

K. Heller and Z. Ghahramani. Randomized algorithms for fast bayesian hierarchical clustering. 2005a.

K. A. Heller and Z. Ghahramani. Bayesian hierarchical clustering. *International Conference on Machine Learning (ICML)*, 2005b.

Z. Hu, H. Qirong, A. Dubey, and E. Xing. Large-scale distributed dependent nonparametric trees. *International Conference on Machine Learning (ICML)*, 2015.

P. Jain, R. Meka, and I. S. Dhillon. Simultaneous unsupervised learning of disparate clusterings. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 2008.

N. Jardine and R. Sibson. Mathematical taxonomy. Technical report, 1971.

I. Jeantet, Z. Miklós, and D. Gross-Amblard. Overlapping hierarchical clustering (OHC). *Advances in Intelligent Data Analysis*, 2020.

C. Jin, M. M. A. Patwary, A. Agrawal, W. Hendrix, W.-k. Liao, and A. Choudhary. Disc: A distributed single-linkage hierarchical clustering algorithm using mapreduce. 2013.

C. Jin, R. Liu, Z. Chen, W. Hendrix, A. Agrawal, and A. Choudhary. A scalable hierarchical clustering algorithm using spark. *International Conference on Big Data Computing Service and Applications*, 2015.

A. Kobren, N. Monath, A. Krishnamurthy, and A. McCallum. A hierarchical algorithm for extreme clustering. *Knowledge Discovery and Data Mining (KDD)*, 2017.

A. Kobren, N. Monath, and A. McCallum. Integrating user feedback under identity uncertainty in knowledge base construction. *Automated Knowledge Base Construction (AKBC)*, 2019.

M. Kramer, J. Dutkowski, M. Yu, V. Bafna, and T. Ideker. Inferring gene ontologies from pairwise similarity data. *Bioinformatics*, 2014.

A. Krishnamurthy, S. Balakrishnan, M. Xu, and A. Singh. Efficient active algorithms for hierarchical clustering. *International Conference on Machine Learning (ICML)*, 2012.

S. Kushagra, S. Samadi, and S. Ben-David. Finding meaningful cluster structure amidst background noise. *International Conference on Algorithmic Learning Theory (ALT)*, 2016.

A. Lai and J. Hockenmaier. Learning to predict denotational probabilities for modeling entailment. *EACL*, 2017.

M. Law, R. Liao, J. Snell, and R. Zemel. Lorentzian distance learning for hyperbolic representations. *International Conference on Machine Learning (ICML)*, pages 3672–3681, 2019.

J. Liu, Q. Zhang, W. Wang, L. McMillan, and J. Prins. Clustering pair-wise dissimilarity data into partially ordered sets. *Conference on Knowledge discovery and data mining (KDD)*, 2006.

J. Liu, Q. Zhang, W. Wang, L. McMillan, and J. Prins. Poclustering: Lossless clustering of dissimilarity data. *International Conference on Data Mining*, 2007.

J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. *International Conference on Machine Learning (ICML)*, 2009.

F. Mémoli and O. B. Okutan. Reeb posets and tree approximations. *Discrete Mathematics*, 2020.

T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin. Advances in pre-training distributed word representations. *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, 2018.

G. A. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 1995.

N. Monath, A. Kobren, A. Krishnamurthy, M. R. Glass, and A. McCallum. Scalable hierarchical clustering with tree grafting. *Knowledge Discovery & Data Mining (KDD)*, 2019a.

N. Monath, M. Zaheer, D. Silva, A. McCallum, and A. Ahmed. Gradient-based hierarchical clustering using continuous representations of trees in hyperbolic space. *Conference on Knowledge Discovery & Data Mining (KDD)*, 2019b.

B. Moseley, K. Lu, S. Lattanzi, and T. Lavastida. A framework for parallelizing hierarchical clustering methods. *ECML PKDD*, 2019.

D. Müllner. Modern hierarchical, agglomerative clustering algorithms. *arXiv preprint arXiv:1109.2378*, 2011.

F. Murtagh. A survey of recent advances in hierarchical clustering algorithms. *Comput. J.*, 1983.

M. Nickel and D. Kiela. Poincaré embeddings for learning hierarchical representations. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

M. Nickel and D. Kiela. Learning continuous hierarchies in the lorentz model of hyperbolic geometry. *International Conference on Machine Learning (ICML)*, 2018.

D. Niu, J. G. Dy, and M. I. Jordan. Multiple non-redundant spectral clustering views. *ICML*, 2010.

C. F. Olson. Parallel algorithms for hierarchical clustering. *Parallel computing*, 1995.

J. Paisley, C. Wang, D. M. Blei, and M. I. Jordan. Nested hierarchical dirichlet processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014.

Z. Qi and I. Davidson. A principled and flexible framework for finding alternative clusterings. *Conference on Knowledge discovery and data mining (KDD)*, 2009.

D. Rao, P. McNamee, and M. Dredze. Streaming cross document entity coreference resolution. *Coling*, 2010.

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 2015.

J. Santos, T. Syed, M. C. Naldi, R. J. Campello, and J. Sander. Hierarchical density-based clustering using mapreduce. *IEEE Transactions on Big Data*, 2019.

R. Suzuki, R. Takahama, and S. Onoda. Hyperbolic disk embeddings for directed acyclic graphs. *International Conference on Machine Learning (ICML)*, 2019.

V. Vasiliauskaite and T. S. Evans. Making communities show respect for order. *Applied Network Science*, 2020.

I. Vendrov, R. Kiros, S. Fidler, and R. Urtasun. Order-embeddings of images and language. *arXiv preprint arXiv:1511.06361*, 2015.

L. Vilnis, X. Li, S. Murty, and A. McCallum. Probabilistic embedding of knowledge graphs with box lattice measures. *ACL*, 2018.

F. Vitale, A. Rajagopalan, and C. Gentile. Flattening a hierarchical clustering through active learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

C. Wu, S. Ioannidis, M. Sznaier, X. Li, D. Kaeli, and J. Dy. Iterative spectral method for alternative clustering. *International Conference on Artificial Intelligence and Statistics*, 2018.

G. Yaroslavtsev and A. Vadapalli. Massively parallel algorithms and hardness for single-linkage clustering under ℓ_p distances. *International Conference on Machine Learning (ICML)*, 2018.

A. Zhang and J. Paisley. Markov mixed membership models. *International Conference on Machine Learning (ICML)*, 2015.

T. Zhang, R. Ramakrishnan, and M. Livny. Birch: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery*, 1997.

X. Zheng, B. Aragam, P. K. Ravikumar, and E. P. Xing. Dags with no tears: Continuous optimization for structure learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

X. Zheng, C. Dan, B. Aragam, P. Ravikumar, and E. Xing. Learning sparse nonparametric dags. *International Conference on Artificial Intelligence and Statistics*, 2020.