Objective-Based Hierarchical Clustering of Deep Embedding Vectors

Stanislav Naumov[1], **Grigory Yaroslavtsev**[2], **Dmitrii Avdiukhin**[2]
[1]ITMO University
[2]Indiana University*
josdas@mail.ru, grigory.yaroslavtsev@gmail.com, davdyukh@iu.edu

Abstract

We initiate a comprehensive experimental study of objective-based hierarchical clustering methods on massive datasets consisting of deep embedding vectors from computer vision and NLP applications. This includes a large variety of image embedding (ImageNet, ImageNetV2, NaBirds), word embedding (Twitter, Wikipedia), and sentence embedding (SST-2) vectors from several popular recent models (e.g. ResNet, ResNext, Inception V3, SBERT). Our study includes datasets with up to 4.5 million entries with embedding dimensions up to 2048.

In order to address the challenge of scaling up hierarchical clustering to such large datasets we propose a new practical hierarchical clustering algorithm B++&C. It gives a $5\%/20\%$ improvement on average for the popular Moseley-Wang (MW) / Cohen-Addad et al. (CKMM) objectives (normalized) compared to a wide range of classic methods and recent heuristics. We also introduce a theoretical algorithm B2SAT&C which achieves a 0.74-approximation for the CKMM objective in polynomial time. This is the first substantial improvement over the trivial $2/3$-approximation achieved by a random binary tree. Prior to this work, the best poly-time approximation of $\approx 2/3 + 0.0004$ was due to Charikar et al. (SODA'19).

1 Introduction

Vector embeddings, in particular those obtained via deep neural nets, are an extremely popular technique for representing unstructured data (e.g. images, text, videos, etc.) as vectors in a d-dimensional feature space. While resulting vectors are most frequently used for classification they can also serve as representations for other downstream machine learning tasks, including clustering, deduplication, recommendation systems, etc. Flat clustering of vector embeddings has been studied extensively (e.g. (Min et al. 2018; Guérin et al. 2017)). In this paper we focus on *hierarchical clustering*, which has a large number of applications, including anomaly detection (Deb and Dey 2017; Parwez, Rawat, and Garuba 2017; Fu, Hu, and Tan 2005), personalized recommendations (Zhang et al. 2014) and construction of flat clusterings (Sander et al. 2003). There are classical and recent approaches which allow one to learn a hierarchy on objects in either supervised (Wu, Tygert, and LeCun 2019; Nickel and Kiela 2017) or unsupervised fashion (Yang, Parikh, and Batra 2016; Shin, Song, and Moon 2019; Mathieu et al. 2019). However, such approaches are substantially more expensive than hierarchical clustering of embedding vectors. Hence hierarchical clustering of deep vector embeddings has emerged as a computationally efficient alternative (e.g. for applications to face recognition (Lin, Chen, and Chellappa 2017)).

In this paper we focus on scalable algorithms for *objective-based hierarchical clustering*, i.e. clustering which optimizes a certain well-defined objective function. Designing an objective function for hierarchical clustering which can be approximated efficiently is challenging, and only recently substantial progress has been made following the work by Dasgupta (2015). In this paper we focus on two popular objectives inspired by it: a similarity-based objective introduced in Moseley and Wang (2017) (MW) and a distance-based objective introduced in Cohen-addad et al. (2019) (CKMM).

Intuitively, these objectives measure the quality of the resulting hierarchical clustering on a random triple of objects from the dataset. They incentivize solutions where the more similar pair in the triple is closer in the resulting hierarchy (see Sec 1.1 for formal definitions). Worst-case approximation algorithms for these objectives are known (Moseley and Wang 2017; Charikar, Chatziafratis, and Niazadeh 2019; Chatziafratis et al. 2020; Cohen-addad et al. 2019; Alon, Azar, and Vainstein 2020). Beyond worst-case analysis has been given for the hierarchical stochastic block model (Cohen-Addad, Kanade, and Mallmann-Trenn 2017) and for vector data (Charikar et al. 2019).

We study performance of objective-based hierarchical clustering methods on large beyond worst-case datasets consisting of deep vector embeddings. We perform experiments on massive datasets (number of objects n is in range $[5 \cdot 10^4, 4.5 \cdot 10^6]$ and embedding dimension d is in range $[100, 2048]$) of word, sentence, and image embedding vectors from the last hidden layer of various popular neural architectures. We study three types of algorithms: 1) algorithms with rigorous guarantees for the MW/CKMM objectives, 2) classic hierarchical agglomerative methods, 3) some other popular algorithms without guarantees scalable to large data.

While the best worst-case approximations for MW and CKMM objectives are 0.585 (Alon, Azar, and Vainstein 2020) and $\approx 2/3 + 0.0004$ (Charikar, Chatziafratis, and Niazadeh 2019) respectively, we show that in practice, for deep

*Supported by NSF CCF-1657477 and Facebook Faculty Award.

vector embeddings in computer vision and natural language processing, many algorithms achieve a much better approximation. We conduct our experiments for cosine similarity (for MW) and squared Euclidean distance (for CKMM) due to their widespread use as similarity/distance measures in deep learning applications, but we believe that our findings are likely to hold for various other measures as well.

Given the popularity of various heuristics for hierarchical clustering, we don't aim to provide a full list of all possible approaches and objectives (see classic and recent surveys for an overview (Murtagh and Contreras 2012, 2017; Jain, Murty, and Flynn 1999; Manning, Raghavan, and Schütze 2008)). A non-exhaustive list of other methods and objectives, which we omit in this study due to lack of rigorous guarantees for MW/CKMM or scalability issues, includes various spectral methods (Wu et al. 2018; Huang et al. 2019), LSH-based average-linkage (Cochez and Mou 2015a; Abboud, Cohen-Addad, and Houdrougé 2019), various k-means algorithms (Zhong 2005; Wang, Gittens, and Mahoney 2019; Chen, Zhou, and Ma 2020) and a recent objective for bisecting k-means (Wang and Moseley 2020).

1.1 Preliminaries

Distances and similarities. In machine learning applications, some of the most popular similarity and dissimilarity measures for feature vectors are cosine similarity $cos\text{-}sim(\mathbf{x},\mathbf{y}) = \frac{\langle \mathbf{x},\mathbf{y}\rangle}{2\|\mathbf{x}\|_2\|\mathbf{y}\|_2} + \frac{1}{2}$ (e.g. for deep representation learning (Reimers and Gurevych 2019)) and squared Euclidean distance $L_2^2(\mathbf{x},\mathbf{y}) = \|\mathbf{x}-\mathbf{y}\|_2^2$ (e.g. used in k-means). Another frequently used class of similarity functions is radial basis functions $RBF_\gamma(\mathbf{x},\mathbf{y}) = e^{-\gamma\|\mathbf{x}-\mathbf{y}\|_2^2}$. These measures are examples of asymmetric kernel functions, i.e. there exist kernel-defining functions ϕ and ψ which allow to compute these measures either exactly ($cos\text{-}sim$, L_2^2) or approximately (RBF_γ) as dot products $\langle \phi(\mathbf{x}), \psi(\mathbf{y})\rangle$ in some inner product spaces (Rahimi and Recht 2007). Such representation allows us to use an important optimization which we call an *inverse kernel trick* (see Section 2).

Hierarchical clustering. Given a set of n objects, the goal of *hierarchical clustering* (HC) is to find a tree $\mathcal{T}$ (also referred to as a *dendrogram*) which contains them as leaves. The internal nodes of $\mathcal{T}$ then correspond to clusters of objects at various levels of granularity. We hence refer to the internal nodes of $\mathcal{T}$ as *clusters*, while also treating them as sets of all objects in their subtrees. For a pair of leaves (e_1, e_2) let $\mathrm{LCA}_{\mathcal{T}}(e_1, e_2)$ be the cluster $C \in \mathcal{T}$ of the smallest cardinality such that $e_1, e_2 \in C$.

Objectives for HC. Measuring the quality of a HC is more challenging than evaluating the performance of basic flat clustering and label prediction tasks. Two major obstacles which have inhibited progress on optimization algorithms for HC are: 1) difficulty of collecting accurate ground truth tree information (instead, triples (Chatziafratis, Niazadeh, and Charikar 2018), quadruples (Ghoshdastidar, Perrot, and von Luxburg 2019) and flat classes (Kobren et al. 2017) are often used) and 2) diversity of methods using which such ground truth can be compared with the algorithm's output $\mathcal{T}$: tree edit distance (Cochez and Mou 2015b; Bille 2005), flat clustering score (Abboud, Cohen-Addad, and Houdrougé 2019; Bateni et al. 2017; Kobren et al. 2017). In a typical scenario when a ground truth partition $\{ C_i \}_{i=1}^K$ into flat clusters is known, a popular quality measure is *dendrogram purity* (DP) (Heller and Ghahramani 2005), defined as maximizing:

$$\mathrm{DP}(\mathcal{T}) = \frac{1}{\sum_{i=1}^K |C_i|^2} \sum_{i=1}^K \sum_{e_1,e_2 \in C_i} \frac{|C_i \cap \mathrm{LCA}_{\mathcal{T}}(e_1,e_2)|}{|\mathrm{LCA}_{\mathcal{T}}(e_1,e_2)|} \tag{1}$$

However, DP says little about the quality of $\mathcal{T}$ overall – perfect DP can be achieved when each ground truth cluster corresponds to a subtree, regardless of hierarchy on top or inside of the subtrees.

Addressing the above challenges, a recent line of work by Dasgupta (2015); Moseley and Wang (2017); Cohen-addad et al. (2019) has proposed a family of related optimization objectives for HC. Instead of relying on ground truth information, these methods only use either distances (d_{ij}) or similarities (w_{ij}) between the data points.

Definition 1.1 *Let $w\colon V \times V \to \mathbb{R}_{\geq 0}$ be a similarity function. Then Dasgupta's objective minimizes*

$$Q_D(\mathcal{T}) := \sum_{i<j} w_{ij} |\mathrm{LCA}_{\mathcal{T}}(e_i, e_j)| \to \min \tag{2}$$

A complementary Moseley-Wang's (MW) objective maximizes

$$Q_M(\mathcal{T}) := \sum_{i<j} w_{ij} (n - |\mathrm{LCA}_{\mathcal{T}}(e_i, e_j)|) \to \max \tag{3}$$

Definition 1.2 *Let $d\colon V \times V \to \mathbb{R}_{\geq 0}$ be a distance function. Then Cohen-Addad et al. (CKMM) objective maximizes*

$$Q_C(\mathcal{T}) := \sum_{i<j} d_{ij} |\mathrm{LCA}_{\mathcal{T}}(e_i, e_j)| \to \max \tag{4}$$

over binary *trees.*

Note that $Q_D(\mathcal{T}) + Q_M(\mathcal{T}) = \sum_{i<j} w_{ij} n = const$, and hence minimizing Q_D is equivalent to minimizing Q_M. Q_D and Q_C have similar expressions; however, since one uses similarities and another one uses distance, Q_D is minimized, while Q_C is maximized. In the worst-case, optimizing these three objectives exactly is NP-hard (Dasgupta 2015; Cohen-addad et al. 2019)

Approximations and normalized objectives for HC. One of our goals is to measure approximations achieved by various algorithms. Since computing the optimum is NP-hard, we use an upper bound $Q^{ub} \geq OPT$ on it instead. *Approximation factor* is then defined as $\alpha(\mathcal{T}) = {}^{Q(\mathcal{T})}/_{Q^{ub}}$.

One of the effects of considering similarity/dissimilarity graphs induced by high-dimensional vector representations is the concentration of measure. The distributions of weights have small standard deviations and large means[1]. This is why

[1]E.g. for embeddings of ImageNetV2 using ResNet34 mean cosine similarity $cos\text{-}sim$ between vectors in the same class is ≈ 0.88 and between different classes is ≈ 0.75.

the approach which returns a random binary tree (denoted as RANDOM) gives a good approximation for the objectives above. To highlight the differences in quality between different algorithms, we propose normalized versions of the objectives which measures advantage over RANDOM.

Definition 1.3 *Let $\mathcal{T}_R$ be a random binary tree clustering and Q be a maximization objective. Then we define* normalized *approximation factors for Q as*

$$\alpha^*(\mathcal{T}) = \frac{Q(\mathcal{T}) - \mathbb{E}[Q(\mathcal{T}_R)]}{Q^{ub} - \mathbb{E}[Q(\mathcal{T}_R)]} \tag{5}$$

With a slight abuse of notation, for a triple (i, j, k) let $(\hat{i}, \hat{j}, \hat{k})$ be a permutation of indices such that $\hat{i}$ and $\hat{j}$ have the smallest cardinality $\mathrm{LCA}_{\mathcal{T}}$ in the triple. Charikar, Chatziafratis, and Niazadeh (2019, Section 4.3) show that $Q_M(\mathcal{T}) = \sum_{i<j<k} w_{\hat{i}\hat{j}}$ and hence for the MW objective we can use a standard upper bound

$$Q_M^{ub} = \sum_{i<j<k} \max(w_{ij}, w_{ik}, w_{jk})$$

For Dasgupta's objective, Wang and Wang (2018, Claim 1) show that $Q_D(\mathcal{T}) = \sum_{i<j<k}(w_{\hat{i}\hat{k}} + w_{\hat{j}\hat{k}}) + 2\sum_{i<j} w_{ij}$. Since the expressions for Q_D and Q_C are similar, we have

$$Q_C^{ub} = \sum_{i<j<k} \max(d_{ij}+d_{ik}, d_{ik}+d_{jk}, d_{ij}+d_{jk}) + 2\sum_{i<j} d_{ij}$$

1.2 Our Contributions

The main contributions of our paper are the following:

Experimental study of objective-based HC. We provide the first comprehensive experimental study of objective-based hierarchical clustering methods on massive datasets consisting of deep embedding vectors. For such vectors, we compare various HC clustering approaches, including classical hierarchical agglomerative clustering, top-down approaches and various other recent HC algorithms.

New normalized objectives. Due to the special structure of vector embedding data[2], even the simplest approaches produce solutions with very high MW and CKMM scores (even a trivial random tree achieves 85-97% approximation). To address this issue, in Section 1.1 we introduce *normalized* MW/CKMM objectives which are better suited for deep vector embedding data. They capture the advantage over a trivial random tree and allow to better separate performance of different algorithms.

New algorithm B++&C. In Section 2, we propose an algorithm B++&C inspired by Chatziafratis et al. (2020). The main idea is to consider a graph whose vertices are objects and edges are similarities (dissimilarities). We perform a top-level split by partitioning this graph, so that the cut between the resulting parts is minimized (maximized). Our approach differs from Chatziafratis et al. (2020) in that we perform multiple levels of partitioning while also allowing imbalance at each level. We show that on deep vector embeddings this algorithm outperforms a wide range of alternatives (see Figure 1, Tables 1 and 2).

[2]Similarities/distances between vectors from the same classes and from different classes in such data can be very close.

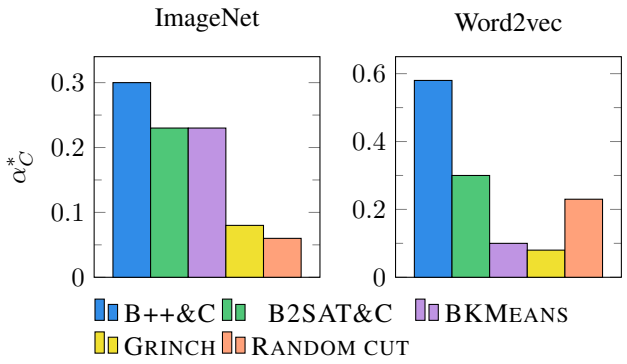

Figure 1: Normalized distance-based CKMM objectives α_C^* under squared Euclidean distance for embeddings of ImageNet using ResNet34 and word embeddings of Wikipedia using Word2vec. B++&C outperforms other approaches by 7% for ImageNet and by 35% for Word2vec.

Scaling up B++&C. One of the main advantages of B++&C is its efficiency. The algorithm is a gradient descent approach inspired by Avdiukhin, Pupyrev, and Yaroslavtsev (2019) applied to a quadratic function.Using a technique which we refer to as *inverse kernel trick* (see Section 2), for several widely used similarities and distance measures, we can represent A as a product of low-rank matrices, which allows us to compute the gradient efficiently.

B2SAT&C and improved approximation for CKMM. In Section 3, we introduce a theoretical hierarchical clustering algorithm B2SAT&C which achieves a 0.74-approximation for the CKMM objective in polynomial time, significantly improving existing $\approx 2/3 + 0.0004$ approximation (Charikar, Chatziafratis, and Niazadeh 2019). The main idea is to reduce the problem of performing a top-level split to the BALANCED MAX-2-SAT problem.

2 BISECT++ AND CONQUER

Our algorithm BISECT++ AND CONQUER (Algorithm 2) is an extension of the BISECT AND CONQUER technique from Chatziafratis et al. (2020) with several crucial modifications which allow one to achieve high practical performance and solution quality. If the set of items V is small (less than a specified parameter θ, typically $\theta \in [100, 5000]$), we solve HC using average-linkage. Otherwise, we reduce our problem to graph partitioning: we introduce a complete graph where vertices are objects and edge weights are similarities (distances) between them, and our goal is to partition the vertices into two sets of a fixed size so that the total weight of edges between the parts is minimized (maximized for distance-based objectives).

Let $\mathbf{x} \in \{-1, 1\}^n$ be a vector such that $\mathbf{x}_i = 1$ if element i belongs to the first part and $\mathbf{x}_i = -1$ otherwise. The graph partitioning problem can be reduced to optimizing a quadratic function $f(\mathbf{x}) = \mathbf{x}^\top W \mathbf{x}$, where W is the similarity matrix, under constraints $\mathbf{x}_i \in \{-1, 1\}$ (each vertex belongs to some part) and $\sum_i x_i \approx 2\delta n$ (balance constraint). Note that in $f(\mathbf{x})$, W_{uv} is taken with a positive sign if u and v are in

Algorithm 1: GRADIENTDESCENTPARTITIONING: δ-imbalanced Graph 2-Partitioning via Randomized Projected Gradient Descent

parameters: noise variance r, learning rates $\{\eta_t\}$, the number of iterations I, kernel-defining functions $\phi, \psi : \mathbb{R}^d \to \mathbb{R}^k$
input : Feature vectors $V = \{v_1, \dots, v_n\} \subseteq \mathbb{R}^d$, imbalance $\delta \in [0, 0.5]$
output : imbalanced partition of V into (V_1, V_2)

$\mathcal{B} = [-1, 1]^n \cap \{\mathbf{x} \in \mathbb{R}^n \mid \sum_i x_i = 2\delta n\}$
for $i = 1$ **to** n **do** `// Compute` $\Phi, \Psi \in \mathbb{R}^{n \times k}$
  $\Phi_i, \Psi_i \leftarrow \phi(v_i), \psi(v_i)$
$\mathbf{x}^{(0)} \leftarrow \operatorname{argmin}_{\mathbf{x} \in \mathcal{B}} \|\mathcal{N}_n(\mathbf{0}, r) - \mathbf{x}\|$
`// Projected gradient descent`
for $t = 0$ **to** $I - 1$ **do**
  $\mathbf{y}^{(t+1)} \leftarrow \mathbf{x}^{(t)} - \eta_t \Phi \Psi^\top \mathbf{x}^{(t)}$
  $\mathbf{x}^{(t+1)} \leftarrow \operatorname{argmin}_{\mathbf{x} \in \mathcal{B}} \|\mathbf{y}^{(t+1)} - \mathbf{x}\|$
$V_1 \leftarrow V_2 \leftarrow \emptyset$;
for *each* $i \in V$ **do** `// Randomized rounding`
  `With probability` $\frac{x_i^{(t)}+1}{2}$`,`
  `let` $V_1 \leftarrow V_1 \cup \{i\}$
  `otherwise,` $V_2 \leftarrow V_2 \cup \{i\}$
return (V_1, V_2)

Algorithm 2: BISECT++ AND CONQUER: Hierarchical clustering via imbalanced graph partitioning

parameters: required imbalance δ, the maximum number of elements to run average linkage θ
input : Feature vectors $V = \{v_1, \dots, v_n\} \subseteq \mathbb{R}^d$
output : Clustering tree on V

if $n < \theta$ **then**
  return AVERAGELINKAGE(V)
$V_1, V_2 \leftarrow$ GRADIENTDESCENTPARTITIONING(V, δ)
return $\{V_1, V_2\} \cup$ B++&C$(V_1) \cup$ B++&C(V_2)

the same part ($\mathbf{x}_u = \mathbf{x}_v$), and with a negative sign otherwise. δ is an imbalance parameter controlling sizes of the parts, which are approximately $(1/2 - \delta)|V|$ and $(1/2 + \delta)|V|$. If the parts should be equal, $\delta = 0$; otherwise, we can tune δ to improve partition quality on imbalanced datasets.

Our algorithm is based on the approach described in Avdiukhin, Pupyrev, and Yaroslavtsev (2019): we optimize a continuous relaxation ($\mathbf{x}_i \in [-1, 1]$) of the function above. The algorithm is a *projected gradient descent approach* which optimizes $f(\mathbf{x}) = \mathbf{x}^\top W \mathbf{x}$ under constraints $\mathbf{x}_i \in [-1, 1]$ and $\sum_i x_i = 2\delta n$. In the end, i-th element goes to the first part with probability $(\mathbf{x}_i+1)/2$. The key idea of our approach is the "Inverse kernel trick", which helps us to avoid building an explicit graph, as described below.

Inverse kernel trick. Note that computing the gradient $\nabla f(\mathbf{x}) = 2W\mathbf{x}$ naïvely requires either $O(n^2 d)$ time or $O(n^2)$ space/time per iteration. To scale B++&C, we use a technique which we call an *inverse kernel trick*. This technique is applicable for *kernelizable* similarities and distance measures which can be represented as $w_{ij} = \langle \phi(v_i), \psi(v_j) \rangle$ for functions $\phi, \psi : \mathbb{R}^d \to \mathbb{R}^k$, which we call *kernel-defining functions* (examples can be found in the full version). These functions can be defined using matrices $\Phi, \Psi \in \mathbb{R}^{n \times k}$, whose i-th rows are $\Phi_i = \phi(v_i)$ and $\Psi_i = \psi(v_i)$. Then the gradient can be computed as $W\mathbf{x} = \Phi(\Psi^\top \mathbf{x})$ in $O(nk)$ time. Some kernels (e.g. RBF) do not have finite-dimensional kernelization. In such cases we use an unbiased finite-dimensional estimation of the kernel (see Table 3 in the full version)

We now outline the algorithm. We first precompute matrices Φ and Ψ as described above. Then we repeatedly perform a gradient descent stepand project the current point onto the feasible space. In the end we perform randomized rounding. Time complexity of B++&C is $O(Ink \log n + \theta nk)$, where I is the number of iterations (since when we use average linkage, we have n/θ sets of size θ, and therefore the total complexity of average linkage is $O(\theta nk)$). Space complexity is $O(nk)$. B++&C is highly parallelizable since after each iteration, the tree is divided into two independent subtrees. Additionally, each iteration is highly parallelizable since the most expensive operations are matrix-vector multiplications.

3 B2SAT&C: Improved Approximation for the CKMM Objective

In this section, we introduce our main algorithm (Algorithm 4) which achieves 0.74-approximation for the CKMM objective, significantly improving the previous $\approx 2/3 + 0.0004$ approximation. The main idea behind our algorithm is to use BALANCED MAX-2-SAT PARTITIONING as a subroutine. In BALANCED MAX-2-SAT PARTITIONING the objective is to partition a set V into two sets S and T of approximately the same size, such that the total weight of edges with at least one endpoint in S, i.e. $\sum_{u,v \in V:\ u \in S \text{ or } v \in S} d_{uv}$, is maximized.

This objective can be expressed as an instance of BALANCED MAX-2-SAT: given a 2-SAT formula whose clauses have specified weights, the goal is to find an assignment for which exactly half of the variables are true and the total weight of satisfied clauses is maximized. Our Algorithm 3 constructs this instance of BALANCED MAX-2-SAT as a collection of n^2 disjunctions: for each $u, v \in V$ we introduce a clause $(x_u \vee x_v)$ with weight d_{uv} (Line 3). Given a solution for the instance let $S = \{u \colon x_u = 1\}$ and $T = \{u \colon x_u = 0\}$, we select the best of the following three solutions $\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3$ (see Appendix in the full version for an illustration):

- ($\mathcal{O}_1$) We define RANDOM(S) and RANDOM(T) as random permutations of elements of S and T. We then concatenate these permutations and define the solution as a path tree, where elements from S are at the top and elements from T are at the bottom.
- ($\mathcal{O}_2$) Union of recursive bisections of S and T.
- ($\mathcal{O}_3$) Recursive bisection of V.

The motivation for BALANCED MAX-2-SAT PARTITIONING is that, in the path solution, edges inside S and edges between S and T will have a greater LCA compared to edges inside T. Thus they make a more significant contribution to the CKMM objective. As a result, we can show the following:

Algorithm 3: BALANCED MAX-2-SAT PARTITIONING

input : Distance function $d\colon V \times V \to \mathbb{R}_{\geq 0}$.
output : Partitioning of S

Set $C \leftarrow \emptyset$
for $(u, v) \in V \times V$ **do**
  Add clause $(x_u \vee x_v)$ to C with weight d_{uv}
return BALANCED MAX-2-SAT(C)

Algorithm 4: Hierarchical Clustering via MAX-2-SAT (B2SAT&C)

input : Feature vectors $V = \{ v_1, \ldots, v_n \} \subseteq \mathbb{R}^d$, distance function $d\colon V \times V \to \mathbb{R}_{\geq 0}$
output : Clustering tree on V

Let $S \subseteq V$ be the set of vertices corresponding to positive variables in the assignment given by BALANCED MAX-2-SAT PARTITIONING(d);
$\mathcal{O}_1 \leftarrow$ PATH(CONCAT(RANDOM(S), RANDOM$(V \setminus S)$))
$\mathcal{O}_2 \leftarrow$ (BALANCED-BISECTION-HC(S, d), BALANCED-BISECTION-HC$(V \setminus S, d)$)
$\mathcal{O}_3 \leftarrow$ BALANCED-BISECTION-HC(V, d)
return *Best of* $\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3$

Theorem 3.1 *[Proof in the full version] Algorithm 4 constructs a tree which gives a multiplicative γ-approximation of the optimum value of the CKMM objective, where $\gamma \geq 0.74$.*

4 Experiments

We give a comprehensive experimental evaluation of a wide range of hierarchical clustering algorithms, including:

- Grafting and rotation-based incremental hierarchical clustering (**GRINCH**) (Monath et al. 2019a),
- Local rotation-based incremental hierarchical clustering (**PERCH**) (Kobren et al. 2017),
- Bisecting k-means (**BKMEANS**),
- Robust single-linkage implementation from the HDBSCAN library (McInnes and Healy 2017; McInnes, Healy, and Astels 2017) (**ROBUSTSL**) (Chaudhuri and Dasgupta 2010; Chaudhuri et al. 2014),
- Minimum spanning tree based affinity clustering (**AFFINITYC**) (Bateni et al. 2017),
- Ward's method (**WARD'SM**) (Ward 1963)
- Average linkage (**AVERAGEL**) (Sokal and Michener 1958),
- Single linkage (**SINGLEL**) (Gower and Ross 1969)
- Complete linkage (**COMPLETEL**) (Sørensen et al. 1948),
- 1D projection based **RANDOM CUT** (Charikar et al. 2019); we project all vectors onto random line and partition them based on the order they appear on the line,
- Our own new algorithms **B2SAT&C**[3] and **B++&C**.
- A special case of B++&C with no imbalance ($\delta = 0$), which we denote as **B++&C**($\delta = 0$). This algorithm is similar to a balanced bisection algorithm from Chatziafratis et al. (2020), with the main difference being is that we perform bisection on multiple levels.

In the full version, we additionally perform comparison with the following HC algorithms which produce non-binary trees:

- Gradient-based optimization of representations of trees in hyperbolic space (**GHHC**) (Monath et al. 2019b),
- Top-down incremental hierarchical clustering (**BIRCH**) (Zhang, Ramakrishnan, and Livny 1996),

4.1 Datasets

We use a large number of vector embeddings including basic supervised and unsupervised constructions, embeddings with pre-trained networks on fresh data from the same distribution, embeddings with pre-trained networks on a different distribution and metric/similarity learning with triplet loss. We focus on large high-dimensional datasets (n up to $4.5 \cdot 10^6$, d up to 2048) consisting of embedding vectors arising from applications to computer vision (CV) and natural language processing (NLP). We study five different types of datasets: three types of image data ("features", "generalization", "shift"), and two types of text data ("word", "sentence"). In order to facilitate comparison with the previous work, we also provide results on smaller datasets for other machine learning applications (Glass[4], Spambase[4], CovType[4], ALOI (Geusebroek, Burghouts, and Smeulders 2005)) and on larger datasets (ImageNet Inception (Monath et al. 2019b)) in the full version.

CV: supervised embeddings ("features"). This is the most vanilla setting, in which embedding vectors are constructed via supervised learning. Vectors are taken from the last hidden layer of a pre-trained neural net. We use image embeddings of ImageNet ILSVRC 2012 (Deng et al. 2009) via ResNet34 (He et al. 2015).

CV: generalization ("generalization"). The "generalization" setting is similar to "features" except that we perform evaluation on a fresh set of samples from a *similar distribution*. We use ResNet34 pre-trained on ImageNet ILSVRC 2012 to compute embedding vectors of images from ImageNetV2 (Recht et al. 2019). As shown in Recht et al. (2019), Top-1 accuracy drops by 12.1% on this dataset.

CV: distribution shift ("shift"). The "shift" setting is similar to "features" and "generalization" except that we perform evaluation on a *different distribution*. We use ResNet34 pre-trained on ImageNet ILSVRC 2012 to compute embedding vectors of NaBirds (Van Horn et al. 2015). These two datasets have very little intersection on classes: ImageNet ILSVRC 2012 is very general and contains 1000 classes, of which only 55 are birds, while NaBirds is more domain-specific and contains 555 classes of birds.

NLP: word embeddings ("word"). In the "word" setting we use unsupervised word embedding vectors trained on Twitter [5] and Wikipedia (Yamada et al. 2020) using two classic methods Glove (Pennington, Socher, and Manning 2014) and Word2vec (Yamada et al. 2016; Mikolov et al. 2013). We emphasize that this setting corresponds to a *fully*

[3] Implementation of our theoretical algorithm using kernelized gradient descent based balanced 2SAT solver.

[4] http://archive.ics.uci.edu/ml

[5] https://nlp.stanford.edu/projects/glove

unsupervised pipeline since both datasets and HC algorithms we use are unsupervised.

NLP: sentence embeddings ("sentence"). In the "sentence" setting we use a pre-trained Sentence-BERT (Reimers and Gurevych 2019) to construct embeddings from the sentiment analysis dataset of movie reviews SST-2 (Socher et al. 2013). We use a RoBERTa-based (Liu et al. 2019) model roberta-base-nli-stsb-mean-tokens [6] which has been trained to produce meaningful sentence representations. Comparing to "shift" this setting shows more advanced techniques of similarity and representation learning including siamese architecture and triplet loss. Cosine similarity $cos\text{-}sim$ between sentence representations corresponds to semantic similarity.

4.2 Results

We report our key experimental results in Table 1 and Table 2. Experiments were performed on 8 CPUs 2.0GHz Intel Xeon Scalable Processor (Skylake), 90Gb RAM. Missing entries are due to timeouts (5 hours) or memory limits. The scalability of different methods is discussed in the full version. In order to highlight the quality of the resulting HC, the algorithms are sorted by their average rank. Table 1 and Table 2 contains only mean value of 5 repetitions, full results and standard deviations are provided in the full version. MW and CKMM objectives are not suitable for non-binary trees. In order to compare with algorithms which produce non-binary trees (GHHC, BIRCH), in the full version we introduce appropriate extensions of these objectives to non-binary trees. See the full version for a complete set of experimental results.

Naïve random baseline. For the MW objective, a random binary tree (RANDOM) achieves a $1/3$-approximation in expectation, while for CKMM objective it achieves a $2/3$-approximation. Worst-case analysis predicts that beating these baselines is challenging: current best poly-time approximations are 0.42 for MW (B++&C($\delta = 0$) (Chatziafratis et al. 2020)) and 0.74 for CKMM (B2SAT&C, our work, Theorem 3.1). Furthermore, for many practical algorithms worst-case analysis predicts that they either can't go above the naïve baselines (AVERAGEL (Charikar, Chatziafratis, and Niazadeh 2019)) or even fail to meet it (BKMEANS (Moseley and Wang 2017)). Such worst-case instances are folklore for COMPLETEL, AFFINITYC and SINGLEL.

Approximation and normalization of objectives. On deep-learned vector embeddings, almost all algorithms dramatically outperform the naïve approximation baselines. Due to concentration of measure (as discussed in the end of Section 1.1) even RANDOM gets at least 85-94% / 91-97% of the optimum, which noticeably outperform the worst case $1/3$ and $2/3$ approximations. Furthermore, classic HAC algorithms (AVERAGEL, COMPLETEL, WARD'SM) also work much better than predicted by the worst-case analysis. Our results in Table 2 show that approximations achieved by various algorithms are very close, with many of them being within 1% difference. Therefore, to highlight performance variations between the algorithms, we measure advantage over RANDOM and focus on normalized objectives α^*_M/α^*_C.

Performance on different types of data. Our experiments show that the quality of HC produced by different algorithms can vary substantially across different types of data. For example, optimizing CKMM on unsupervised word embedding vectors (Twitter, Wikipedia) turns out to be rather challenging (see Table 1). Performance of most approaches drops drastically on these vectors, sometimes even below a simple RANDOM CUT. Nevertheless, despite this variablity, B++&C shows consistently best performance across the different types of vector embeddings.

Performance of various types of algorithms. While HAC approaches (AVERAGEL, COMPLETEL, WARD'SM) often show good performance, their running time scales superlinearly making them prohibitively slow on large datasets. Among scalable algorithms, our results show the advantage of top-down methods (B++&C, B2SAT&C, BKMEANS) over nearest-neighbor based approaches (SINGLEL, AFFINITYC, PERCH, and GRINCH). This is due to the fact that MW/CKMM objectives encourage solutions with good global structure which top-down methods tend to recover better than approaches focused on exploiting local structure.

Performance of B++&C. Our proposed combination of top-down unbalanced bisection (with kernelization and gradient descent optimization) and average-linkage B++&C appears to robustly give the highest quality performance across a diverse collection of datasets. On normalized MW/CKMM objectives (α^*_M/α^*_C), B++&C outperforms other approaches on all datasets by 2-17%/4-59% respectively.

Generalization properties of HC objectives. Since we use pretrained neural nets to generate vector embeddings, we can compute HC on fresh samples from a similar distribution very quickly[7]. Applying this approach to ImageNetV2 (a fresh sample from the ImageNet distribution) our results show that most algorithms show similar MW/CKMM scores compared to ImageNet. This is rather different from the consistent $\approx 10\%$ prediction accuracy drop reported in Recht et al. (2019). We believe that explanation of this phenomenon might be a promising direction for future work.

5 Conclusion and Future Work

In this paper, we initiate a comprehensive experimental study of HC algorithms on deep embedding vectors. Our results indicate that CKMM is particularly well-suited for such data as it captures the quality of the overall hierarchy and only relies on distances between vectors. We introduce a new scalable algorithm B++&C which outperforms existing approaches on all considered datasets. Moreover, we present a polynomial-time algorithm B2SAT&C that significantly improves the existing approximation for the CKMM objective to 0.74.

A possible future direction is to show approximation guarantees for imbalanced bisection. It might also be interesting to understand why there is no drop in HC quality (contrary to the drop in the classification accuracy shown in Recht et al. (2019)) when generalizing ImageNet embeddings for ImageNetV2 dataset.

[6] https://github.com/UKPLab/sentence-transformers

[7] At the cost of a single forward pass + running a HC algorithm.

Dataset	ImageNet	ImageNetV2	NaBirds	Twitter	Wikipedia	SST-2
Method	ResNet34	ResNet34	ResNet34	Glove	Word2vec	SBERT
Domain	CV	CV	CV	NLP	NLP	NLP
Setting	features	generalization	shift	word	word	sentence
Size	large	small	medium	large	large	medium
B++&C	**.30/.95**	**.83/.99**	**.71/.97**	**.60/.97**	**.58/.94**	**.49/.97**
AVERAGEL	–	.59/.97	–	–	–	–
B2SAT&C	.23/.95	.26/.95	.57/.96	.12/.93	.30/.90	.46/.96
ROBUSTSL	–	.70/.98	.45/.95	–	–	.15/.94
B++&C($\delta = 0$)	.23/.95	.26/.95	.57/.96	.10/.93	.17/.88	.46/.96
COMPLETEL	–	.48/.97	–	–	–	–
BKMEANS	.23/.95	.26/.95	.61/.96	.10/.93	.10/.87	.45/.96
GRINCH	.08/.94	.06/.94	.47/.95	.01/.92	.08/.86	.16/.94
PERCH	.05/.94	.08/.94	.36/.94	.00/.92	–	.15/.94
RANDOM CUT	.06/.94	.09/.94	.12/.92	.11/.93	.23/.89	.07/.94
RANDOM	.00/.94	.00/.94	.00/.91	.00/.92	.00/.85	.00/.93
$n \approx$	$1.2 \cdot 10^6$	10^4	$5 \cdot 10^4$	$1.3 \cdot 10^6$	$4.5 \cdot 10^6$	$7 \cdot 10^4$
d	512	512	512	200	100	768
#classes	10^3	10^3	555	–	–	2

Table 1: Normalized/unnormalized (α_C^*/α_C) distance-based CKMM objectives under squared Euclidean distance. On α_C all algorithms (including RANDOM) give at least 85-94% approximation. On α_C^* B++&C outperforms other approaches on all datasets by 4-59%. Among other scalable algorithms, BKMEANS shows good average performance. Among non-scalable algorithms, basic HAC methods (AVERAGEL, COMPLETEL) and robust single linkage (ROBUSTSL) show competitive performance. Our worst-case theoretical algorithm B2SAT&C also shows substantial gains. Even a simple 1D random projection technique (RANDOM CUT) gives non-trivial results on NLP datasets. Algorithms that performed worse than RANDOM CUT (SINGLEL, AFFINITYC, WARD'SM) are not shown. See the full version for complete results which include comparison with GHHC and BIRCH.

Dataset	ImageNet	ImageNetV2	NaBirds	Twitter	Wikipedia	SST-2
Method	ResNet34	ResNet34	ResNet34	Glove	Word2vec	SBERT
Domain	CV	CV	CV	NLP	NLP	NLP
Setting	features	generalization	shift	word	word	sentence
Size	large	small	medium	large	large	medium
B++&C	**.40/.98**	**.51/.99**	**.73/.99**	**.44/.97**	**.40/.96**	**.51/.96**
COMPLETEL	–	**.51/.99**	–	–	–	–
BKMEANS	.37/.98	.39/.98	.71/.99	.27/.96	.38/.96	.45/.95
B++&C($\delta = 0$)	.37/.98	.39/.98	.67/.99	.23/.95	**.40/.96**	.46/.95
B2SAT&C	.37/.98	.39/.98	.67/.99	.23/.95	**.40/.96**	.46/.95
AVERAGEL	–	.38/.98	–	–	–	–
WARD'SM	–	.35/.98	–	–	–	–
GRINCH	.10/.98	.12/.98	.54/.98	.09/.95	.09/.94	.17/.93
ROBUSTSL	–	.17/.98	.16/.97	–	–	.20/.93
PERCH	.07/.97	.13/.98	.43/.98	.01/.94	–	.15/.92
RANDOM CUT	.06/.97	.06/.97	.13/.97	.06/.94	.13/.95	.07/.92
RANDOM	.00/.97	.00/.97	.00/.96	.00/.94	.00/.94	.00/.91

Table 2: Normalized/unnormalized (α_M^*/α_M) similarity-based MW objectives under cosine similarity. On α_M all algorithms give at least 94-97% approximation. For α_M^*, B++&C outperforms other approaches on all medium-large datasets by 2-17%. Among other scalable algorithms BKMEANS shows good average performance. Among non-scalable algorithms HAC methods (COMPLETEL, AVERAGEL, WARD'SM) show competitive performance, while performance of ROBUSTSL drops compared to the distance-based CKMM objective. Our theoretical algorithm B2SAT&C shows the same performance as B++&C($\delta = 0$). Algorithms which performed worse than RANDOM CUT (SINGLEL, AFFINITYC) are not shown. See the full version for complete results.

References

Abboud, A.; Cohen-Addad, V.; and Houdrougé, H. 2019. Subquadratic High-Dimensional Hierarchical Clustering. In *Advances in Neural Information Processing Systems*, 11576–11586.

Alon, N.; Azar, Y.; and Vainstein, D. 2020. Hierarchical Clustering: A 0.585 Revenue Approximation. In *Conference on Learning Theory, COLT 2020, 9-12 July 2020*, volume 125 of *Proceedings of Machine Learning Research*, 153–162. PMLR.

Avdiukhin, D.; Pupyrev, S.; and Yaroslavtsev, G. 2019. Multi-Dimensional Balanced Graph Partitioning via Projected Gradient Descent. *Proc. VLDB Endow.* 12(8): 906–919.

Bateni, M.; Behnezhad, S.; Derakhshan, M.; Hajiaghayi, M.; Kiveris, R.; Lattanzi, S.; and Mirrokni, V. 2017. Affinity Clustering: Hierarchical Clustering at Scale. In *Advances in Neural Information Processing Systems 30*, 6864–6874. Curran Associates, Inc.

Bille, P. 2005. A Survey on Tree Edit Distance and Related Problems. *Theor. Comput. Sci.* 337(1–3): 217–239.

Charikar, M.; Chatziafratis, V.; and Niazadeh, R. 2019. Hierarchical Clustering Better than Average-Linkage. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '19, 2291–2304. USA: Society for Industrial and Applied Mathematics.

Charikar, M.; Chatziafratis, V.; Niazadeh, R.; and Yaroslavtsev, G. 2019. Hierarchical Clustering for Euclidean Data. In *Proceedings of Machine Learning Research*, volume 89 of *Proceedings of Machine Learning Research*, 2721–2730. PMLR.

Chatziafratis, V.; Niazadeh, R.; and Charikar, M. 2018. Hierarchical Clustering with Structural Constraints. *CoRR* abs/1805.09476. URL http://arxiv.org/abs/1805.09476.

Chatziafratis, V.; Yaroslavtsev, G.; Lee, E.; Makarychev, K.; Ahmadian, S.; Epasto, A.; and Mahdian, M. 2020. Bisect and Conquer: Hierarchical Clustering via Max-Uncut Bisection. In Chiappa, S.; and Calandra, R., eds., *The 23rd International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, 3121–3132. PMLR.

Chaudhuri, K.; and Dasgupta, S. 2010. Rates of convergence for the cluster tree. In *Advances in neural information processing systems*, 343–351.

Chaudhuri, K.; Dasgupta, S.; Kpotufe, S.; and Von Luxburg, U. 2014. Consistent procedures for cluster tree estimation and pruning. *IEEE Transactions on Information Theory* 60(12): 7900–7912.

Chen, L.; Zhou, S.; and Ma, J. 2020. Fast Kernel k-means Clustering Using Incomplete Cholesky Factorization. *arXiv preprint arXiv:2002.02846* .

Cochez, M.; and Mou, H. 2015a. Twister tries: Approximate hierarchical agglomerative clustering for average distance in linear time. In *Proceedings of the 2015 ACM SIGMOD international conference on Management of data*, 505–517.

Cochez, M.; and Mou, H. 2015b. Twister Tries: Approximate Hierarchical Agglomerative Clustering for Average Distance in Linear Time. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, 505–517. New York, NY, USA: Association for Computing Machinery.

Cohen-Addad, V.; Kanade, V.; and Mallmann-Trenn, F. 2017. Hierarchical clustering beyond the worst-case. In *Advances in Neural Information Processing Systems*, 6201–6209.

Cohen-addad, V.; Kanade, V.; Mallmann-trenn, F.; and Mathieu, C. 2019. Hierarchical Clustering: Objective Functions and Algorithms. *J. ACM* 66(4).

Dasgupta, S. 2015. A cost function for similarity-based hierarchical clustering. *CoRR* abs/1510.05043.

Deb, A. B.; and Dey, L. 2017. Outlier detection and removal algorithm in K-means and hierarchical clustering. *World Journal of Computer Application and Technology* 5(2): 24–29.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 248–255. Ieee.

Fu, Z.; Hu, W.; and Tan, T. 2005. Similarity based vehicle trajectory clustering and anomaly detection. In *IEEE International Conference on Image Processing 2005*, volume 2, II–602. Ieee.

Geusebroek, J.-M.; Burghouts, G. J.; and Smeulders, A. W. 2005. The Amsterdam library of object images. *International Journal of Computer Vision* 61(1): 103–112.

Ghoshdastidar, D.; Perrot, M.; and von Luxburg, U. 2019. Foundations of Comparison-Based Hierarchical Clustering. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Gower, J. C.; and Ross, G. J. S. 1969. Minimum Spanning Trees and Single Linkage Cluster Analysis. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 18(1): 54–64.

Guérin, J.; Gibaru, O.; Thiery, S.; and Nyiri, E. 2017. Cnn features are also great at unsupervised classification. *arXiv preprint arXiv:1707.01700* .

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Deep Residual Learning for Image Recognition. *CoRR* abs/1512.03385. URL http://arxiv.org/abs/1512.03385.

Heller, K. A.; and Ghahramani, Z. 2005. Bayesian Hierarchical Clustering. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, 297–304. New York, NY, USA: ACM.

Huang, D.; Wang, C.-D.; Wu, J.; Lai, J.-H.; and Kwoh, C. K. 2019. Ultra-scalable spectral clustering and ensemble clustering. *IEEE Transactions on Knowledge and Data Engineering* .

Jain, A. K.; Murty, M. N.; and Flynn, P. J. 1999. Data clustering: a review. *ACM computing surveys (CSUR)* 31(3): 264–323.

Kobren, A.; Monath, N.; Krishnamurthy, A.; and McCallum, A. 2017. A Hierarchical Algorithm for Extreme Clustering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, 255–264. New York, NY, USA: Association for Computing Machinery. ISBN 9781450348874.

Lin, W.-A.; Chen, J.-C.; and Chellappa, R. 2017. A proximity-aware hierarchical clustering of faces. In *2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)*, 294–301. IEEE.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* .

Manning, C. D.; Raghavan, P.; and Schütze, H. 2008. *Introduction to information retrieval*. Cambridge University Press.

Mathieu, E.; Le Lan, C.; Maddison, C. J.; Tomioka, R.; and Teh, Y. W. 2019. Continuous Hierarchical Representations with Poincaré Variational Auto-Encoders. In *Advances in neural information processing systems*, 12544–12555.

McInnes, L.; and Healy, J. 2017. Accelerated hierarchical density based clustering. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, 33–42. IEEE.

McInnes, L.; Healy, J.; and Astels, S. 2017. hdbscan: Hierarchical density based clustering. *The Journal of Open Source Software* 2(11): 205.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 3111–3119.

Min, E.; Guo, X.; Liu, Q.; Zhang, G.; Cui, J.; and Long, J. 2018. A survey of clustering with deep learning: From the perspective of network architecture. *IEEE Access* 6: 39501–39514.

Monath, N.; Kobren, A.; Krishnamurthy, A.; Glass, M. R.; and McCallum, A. 2019a. Scalable Hierarchical Clustering with Tree Grafting. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, 1438–1448. New York, NY, USA: ACM.

Monath, N.; Zaheer, M.; Silva, D.; McCallum, A.; and Ahmed, A. 2019b. Gradient-based Hierarchical Clustering using Continuous Representations of Trees in Hyperbolic Space. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

Moseley, B.; and Wang, J. 2017. Approximation Bounds for Hierarchical Clustering: Average Linkage, Bisecting K-means, and Local Search. In *Advances in Neural Information Processing Systems 30*, 3094–3103. Curran Associates, Inc.

Murtagh, F.; and Contreras, P. 2012. Algorithms for hierarchical clustering: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 2(1): 86–97.

Murtagh, F.; and Contreras, P. 2017. Algorithms for hierarchical clustering: an overview, II. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 7(6): e1219.

Nickel, M.; and Kiela, D. 2017. Poincaré embeddings for learning hierarchical representations. In *Advances in neural information processing systems*, 6338–6347.

Parwez, M. S.; Rawat, D. B.; and Garuba, M. 2017. Big data analytics for user-activity analysis and user-anomaly detection in mobile wireless network. *IEEE Transactions on Industrial Informatics* 13(4): 2058–2065.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532–1543.

Rahimi, A.; and Recht, B. 2007. Random Features for Large-Scale Kernel Machines. In *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, 1177–1184.

Recht, B.; Roelofs, R.; Schmidt, L.; and Shankar, V. 2019. Do imagenet classifiers generalize to imagenet? *arXiv preprint arXiv:1902.10811* .

Reimers, N.; and Gurevych, I. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. URL http://arxiv.org/abs/1908.10084.

Sander, J.; Qin, X.; Lu, Z.; Niu, N.; and Kovarsky, A. 2003. Automatic extraction of clusters from hierarchical clustering representations. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 75–87. Springer.

Shin, S.; Song, K.; and Moon, I. 2019. Hierarchically Clustered Representation Learning. *CoRR* abs/1901.09906.

Socher, R.; Perelygin, A.; Wu, J.; Chuang, J.; Manning, C. D.; Ng, A. Y.; and Potts, C. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, 1631–1642.

Sokal, R. R.; and Michener, C. D. 1958. A statistical method for evaluating systematic relationships. *University of Kansas Science Bulletin* 38: 1409–1438.

Sørensen, T.; Sørensen, T.; Sørensen, T.; SORENSEN, T.; Sorensen, T.; Sorensen, T.; and Biering-Sørensen, T. 1948. A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on Danish commons .

Van Horn, G.; Branson, S.; Farrell, R.; Haber, S.; Barry, J.; Ipeirotis, P.; Perona, P.; and Belongie, S. 2015. Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 595–604.

Wang, D.; and Wang, Y. 2018. An Improved Cost Function for Hierarchical Cluster Trees. *ArXiv* abs/1812.02715.

Wang, S.; Gittens, A.; and Mahoney, M. W. 2019. Scalable kernel K-means clustering with Nyström approximation: relative-error bounds. *The Journal of Machine Learning Research* 20(1): 431–479.

Wang, Y.; and Moseley, B. 2020. An Objective for Hierarchical Clustering in Euclidean Space and Its Connection to Bisecting K-means. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 6307–6314.

Ward, J. H. 1963. Hierarchical Grouping to Optimize an Objective Function. *Journal of the American Statistical Association* 58(301): 236–244.

Wu, C.; Tygert, M.; and LeCun, Y. 2019. A hierarchical loss and its problems when classifying non-hierarchically. *PloS one* 14(12).

Wu, L.; Chen, P.-Y.; Yen, I. E.-H.; Xu, F.; Xia, Y.; and Aggarwal, C. 2018. Scalable spectral clustering using random binning features. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2506–2515.

Yamada, I.; Asai, A.; Sakuma, J.; Shindo, H.; Takeda, H.; Takefuji, Y.; and Matsumoto, Y. 2020. Wikipedia2Vec: An Efficient Toolkit for Learning and Visualizing the Embeddings of Words and Entities from Wikipedia. *arXiv preprint 1812.06280v3* .

Yamada, I.; Shindo, H.; Takeda, H.; and Takefuji, Y. 2016. Joint Learning of the Embedding of Words and Entities for Named Entity Disambiguation. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, 250–259. Association for Computational Linguistics.

Yang, J.; Parikh, D.; and Batra, D. 2016. Joint unsupervised learning of deep representations and image clusters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5147–5156.

Zhang, T.; Ramakrishnan, R.; and Livny, M. 1996. BIRCH: An Efficient Data Clustering Method for Very Large Databases. *SIGMOD Rec.* 25(2): 103–114.

Zhang, Y.; Ahmed, A.; Josifovski, V.; and Smola, A. J. 2014. Taxonomy Discovery for Personalized Recommendation. In *ACM International Conference on Web Search And Data Mining (WSDM)*.

Zhong, S. 2005. Efficient online spherical k-means clustering. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 5, 3180–3185. IEEE.