On the Hardness of Category Tree Construction

Shay Gershtein ✉
Tel Aviv University, Israel

Uri Avron ✉
Tel Aviv University, Israel

Ido Guy ✉
Ben-Gurion University of the Negev, Beer Sheva, Israel

Tova Milo ✉
Tel Aviv University, Israel

Slava Novgorodov ✉
eBay Research, Netanya, Israel

Abstract

Category trees, or taxonomies, are rooted trees where each node, called a category, corresponds to a set of related items. The construction of taxonomies has been studied in various domains, including e-commerce, document management, and question answering. Multiple algorithms for automating construction have been proposed, employing a variety of clustering approaches and crowdsourcing. However, no formal model to capture such categorization problems has been devised, and their complexity has not been studied. To address this, we propose in this work a combinatorial model that captures many practical settings and show that the aforementioned empirical approach has been warranted, as we prove strong inapproximability bounds for various problem variants and special cases when the goal is to produce a categorization of the maximum utility.

In our model, the input is a set of n weighted item sets that the tree would ideally contain as categories. Each category, rather than perfectly match the corresponding input set, is allowed to exceed a given threshold for a given similarity function. The goal is to produce a tree that maximizes the total weight of the sets for which it contains a matching category. A key parameter is an upper bound on the number of categories an item may belong to, which produces the hardness of the problem, as initially each item may be contained in an arbitrary number of input sets.

For this model, we prove inapproximability bounds, of order $\tilde{\Theta}(\sqrt{n})$ or $\tilde{\Theta}(n)$, for various problem variants and special cases, loosely justifying the aforementioned heuristic approach. Our work includes reductions based on parameterized randomized constructions that highlight how various problem parameters and properties of the input may affect the hardness. Moreover, for the special case where the category must be identical to the corresponding input set, we devise an algorithm whose approximation guarantee depends solely on a more granular parameter, allowing improved worst-case guarantees. Finally, we also generalize our results to DAG-based and non-hierarchical categorization.

2012 ACM Subject Classification Theory of computation → Data structures and algorithms for data management; Theory of computation → Approximation algorithms analysis; Theory of computation → Problems, reductions and completeness

Keywords and phrases maximum independent set, approximation algorithms, approximation hardness bounds, taxonomy construction, category tree construction

Digital Object Identifier 10.4230/LIPIcs.ICDT.2022.4

Related Version *Full Version*: https://slavanov.com/research/icdt22_full.pdf

1 Introduction

Category trees, or taxonomies, are rooted trees where each node corresponds to a labeled category defined as a set of related items. Each non-leaf category is more general than its descendants and contains the union of their item sets. Moreover, each item may typically appear in a bounded number of tree branches. Such trees enable browsing-style information access and play a central role in Web platforms. While taxonomists can identify many desirable categories, producing a single categorization in a compact structure to maximize a given utility measure is challenging. Therefore, multiple algorithms for automating construction in various domains, e.g., e-commerce [2, 12], document management [9], and question answering [24], have been proposed, employing a variety of clustering approaches, and crowdsourcing [9, 21]. However, to our knowledge, the complexity of this problem has not been studied w.r.t. a formal model, and solution evaluations were based on user-studies or a similarity score of the tree categories to a collection of desired categories [17, 9, 18], to measure how well these are captured by the much more succinct solution. Based on the latter evaluation method, we propose a model that captures practical settings and show that the aforementioned heuristic approach has been warranted, as we prove strong inapproximability bounds.

Before describing our results, we first define the formal setting.

Model. The input is a set of n sets of items. The solution space consists of rooted trees (we also examine other structures, as described in the sequel). Each node (category) corresponds to a set of items (not necessarily identical to any set in the input), and every non-leaf category contains the union of all the items of its descendants. Ideally, the tree would have, for every input set, a category that is very similar to it. Each input set is weighted to reflect how valuable it is for a solution to contain a matching category. In practice, an input set represents items that match some criteria a user may have in mind when performing a search, and its weight implies the predicted likelihood of seeking these criteria. The sets are derived from a dataset of result sets to search queries, or, more generally, are formed by grouping items w.r.t. shared properties.

This model has multiple variants, defined by two parameters. The first parameter is a *similarity function*, which measures the similarity of an input set and a category. We examine several variations of commonly used set-similarity functions, which extend the original function with a threshold parameter. A similarity score below this parameter is rounded down to 0, to capture the fact that, when the similarity score is too low, no category is identifiable by the user as a matching category, and has no utility. Given such a function, the tree score for a given input set is the maximum similarity score of any category for this set. The overall tree score is the weighted (w.r.t. input weights) sum of the scores for all the input sets. The goal is to produce a tree of the highest score.

The second parameter is a *copy-bound*, which limits the number of *independent* categories any item can belong to, where categories are called independent if no two are on the same path from the root to a leaf. Most real-life platforms set a low copy-bound, to ensure that the categorization is coherent, compact, and easy to navigate. For example, eBay allows listing an item in a single (lowermost) category for free, or two categories for an extra fee [1].

Our bounds also apply to the related problems, where the aim is to produce a flat categorization or more general DAG structures. Flat categorization may be of independent interest, as it also captures the setting where one seeks, given a collection of overlapping sets, a partition that is maximally similar to the original collection. This may be particularly relevant for clustering and partitioning problems in hypergraphs (see Section 7).

Results. For the optimization problem of maximizing the tree score (as defined above), we prove for all examined variants an inapproximability bound of $\tilde{\Theta}(\sqrt{n})$ or $\tilde{\Theta}(n)$, where n is the number of input sets, highlighting how different problem parameters may affect the hardness. These bounds also apply to unweighted inputs and various special cases. On the other hand, we show that finer properties, such as bounds on the cardinality of input sets or the number of intersections among sets, aid in deriving more relaxed parameterized hardness bounds. To that end, we also provide a positive result in the form of an algorithm for the *Exact variant*, where a category must match an input set exactly to contribute to the objective function. The tight performance guarantee of this algorithm depends only on a parameter that measures the number of intersections between the input sets. Importantly, we reduce this variant to the Maximum Independent Set problem, for which, despite its inapproximability, practical solvers have been devised. We demonstrate the practical utility of this result, in [3] and [4], complementary works focusing on constructing an e-commerce category tree that is maximally similar to result sets of user queries. In these works, we show that leveraging these solvers enables finding optimal solutions to real-world instances and extend this approach to algorithms that solve well instances of more general variants.

An essential component in our methods is defining a generalized similarity function with two threshold parameters that address two more granular similarity measures: *precision* and *recall*. Our key results consist of reductions from the Maximum Independent Set problem in hypergraphs, where we integrate into the reduced instance randomized constructions that closely capture the precision and recall parameters. This not only enables us to derive improved results for more subtle special cases but also to capture more refined properties of hard inputs, which we then leverage to prove hardness w.r.t. other similarity functions (we outline how to schematically apply our arguments to derive hardness bounds for similarity functions not examined here).

While we are not aware of any theoretical results directly comparable to ours, Section 7 discusses motivating empirical research and possible applications to hypergraph partitioning.

Lastly, we note that the complementary problem of labeling the resulting categories has been studied in various settings (e.g., [5]), and is outside the scope of our model.

Outline. Section 2 provides the necessary formalism for our model, while Section 3 presents useful theoretical tools. In Section 4 we provide a positive result for a common problem variant. In Section 5 we prove various approximation hardness results derived w.r.t. the generalized similarity function (with recall and precision thresholds). In Section 6 we leverage these results to derive hardness bounds w.r.t. all other similarity functions defined in Section 2. The related work appears in Section 7 and we conclude in Section 8.

Due to space constraints, we defer all formal proofs to the full version of the paper [10].

2 Model

We now define the model underlying our work, followed by a discussion of problem parameters. We conclude this section with illustrative examples of problem instances in our model.

2.1 Problem definition

The two problems we study are the Optimal Category Tree problem (OCT) and the Optimal Category Partition problem (OCP). The input to both problems is $\langle Q, U, W \rangle$, where $Q \subseteq 2^U$ is a set of n sets over a finite universe U, and $W : Q \to [0, 1]$ is a weighting function that assigns a non-negative weight to each set in Q. We use the term *query*, to denote each set

in Q. Note, in advance, that in the definition of the model we discuss two types of element sets: the queries and the sets corresponding to the tree nodes. In general, these sets are not identical (however, these are typically similar, as the objective is, roughly speaking, to maximize the similarity of the two types of sets, as defined formally below).

Both problems have multiple variants defined by two parameters (explained below, in the context of the solution space): a *copy bound* $r \in \mathbb{N}$, and a *similarity function* $\mathcal{S} : [2^U] \times [2^U] \to [0, 1]$. We denote by $OCT^r(\mathcal{S})$ the r-copy OCT problem with similarity function $\mathcal{S}$, and the analogous OCP variant is denoted by $OCP^r(\mathcal{S})$.

We next formally define the solution space for each of the two problems. We start with $OCP^r(\mathcal{S})$, as it is a simpler form of the model for $OCT^r(\mathcal{S})$.

OCP. We call a set of sets over U an *r-weak partition*, if every element appears in at most r sets. A 1-weak partition is a standard partition. The solution space of $OCP^r(\mathcal{S})$ consists of all r-weak partitions of U. Any such solution is termed as a *category partition*, denoted by P, with the sets contained in it termed as *categories*.

Given a query, $q \in Q$, and a category partition, P, we define the *similarity score* of a category $C \in P$ for q as $\mathcal{S}(q, C)$. The score of P for this query is defined by the category that most closely matches the query as $S(q, P) = \max_{C \in P} \mathcal{S}(q, C)$. Note that the root of the tree, as it is a valid category, may also be the most closely matching category for some queries. The overall score of the category partition is defined as $S(Q, P) = \sum_{q \in Q} W(q) \cdot S(q, P)$. This score is the weighted sum of the scores for all queries, where the weight of each score is the corresponding query weight. The objective of the $OCP^r(\mathcal{S})$ problem is to produce a category partition of the maximum score: $\arg\max_P S(Q, P)$.

OCT. The solution space of $OCT^r(\mathcal{S})$ consists of rooted trees, termed *category trees*, where every tree node, termed *category*, contains a subset of U. We abuse notation, and, when clear from context, we use T to denote both the category tree and the set of its categories. Similarly, we use C to denote a category as well as the set of elements it contains.

A category tree must satisfy the following two requirements. First, every non-leaf category contains the union of the sets of elements contained by its child categories (and possibly other elements). The root of the tree, thus, contains all the elements that appear in any category. Second, for each element $e \in U$ there are at most r semi-leaves w.r.t. e, where the semi-leaves w.r.t. e are the most specific categories to which e belongs (i.e. none of the descendants of any such semi-leaf contain e). Note that for a category tree, unlike a category partition, it is no longer true (nor desirable) that an element is contained in at most r categories. Even for $r = 1$, if an element is contained in some category in the tree, it must also be contained in all its ancestor categories. Therefore, the copy-bound is applied to the number of semi-leaves w.r.t. any given element, with the only other nodes containing the element being all the ancestors of these semi-leaves. For $r = 1$ this requirement implies that any given element in the tree is contained only in categories that are all on the same branch.

All definitions of relevant scoring functions are analogous to $OCP^r(\mathcal{S})$. Concretely, the score of a tree, T, for a query, q, is defined as $S(q, T) = \max_{C \in T} \mathcal{S}(q, C)$. The overall score of T is $S(Q, T) = \sum_{q \in Q} W(q) \cdot S(q, T)$. When Q is clear from context, we use the shorthand $S(T)$. The objective of the $OCT^r(\mathcal{S})$ problem is to produce $\arg\max_T S(Q, T)$.

Unweighted variants. We refer to the special case of OCT (OCP) where all weights are uniform as *unweighted* OCT (OCP) and set all weights to 1. Our hardness proofs leverage unweighted inputs, and therefore our hardness bounds also apply to the unweighted case. Accordingly, in our hardness discussions, the reader may assume this context. We directly use weights only in the upper bound we provide in Section 4.

2.2 Similarity functions

We study several similarity functions, that are dependent (in some cases, implicitly) on the following two underlying similarity measures, *precision* $p(q, C) = \frac{|C \cap q|}{|C|}$ and *recall* $r(q, C) = \frac{|C \cap q|}{|q|}$. We distinguish between *cutoff functions* and *threshold functions*. Both have a threshold parameter $\delta \in (0, 1]$ and use an underlying similarity function f. In both cases, the function outputs 0 when $f(q, C) < \delta$. However, when $f(q, C) \geq \delta$ a cutoff function equals $f(q, C)$, whereas a threshold function equals 1. We first focus, however, on the following, more general, threshold function, which sets a separate threshold for each measure.

▶ **Definition 1** (Granular threshold function). *Given parameters $\alpha, \beta \in [0, 1]$, the granular threshold similarity $\mathcal{T}_{\alpha,\beta}$ of a query q and category C is defined as follows: $\mathcal{T}_{\alpha,\beta}(q, C) = 1$ when $p(q, C) \geq \alpha$ and $r(q, C) \geq \beta$, and $\mathcal{T}_{\alpha,\beta}(q, C) = 0$ otherwise.*

We will also study the common similarity functions defined below.

▶ **Definition 2** (Jaccard similarity). *The Jaccard similarity of a category C and a query q is defined as $J(q, C) = \frac{|q \cap C|}{|q \cup C|}$. The threshold Jaccard similarity, with threshold parameter $\delta \in (0, 1]$, is defined as $\hat{J}_\delta(q, C) = 1$ when $J(q, C) \geq \delta$ and $\hat{J}_\delta(q, C) = 0$ otherwise. The cutoff Jaccard similarity, with threshold parameter $\delta \in (0, 1]$, is defined as $\bar{J}_\delta(q, C) = J(q, C)$ when $J(q, C) \geq \delta$ and $\bar{J}_\delta(q, C) = 0$ otherwise.*

▶ **Definition 3** (F_1 score). *The F_1 score of a category C for a query q is defined as the harmonic mean of the precision and the recall: $F_1(q, C) = 2\frac{p(q,C) \cdot r(q,C)}{p(q,C)+r(q,C)}$. The threshold F_1 score, with parameter $\delta \in (0, 1]$, is defined as $\hat{F}_{1(\delta)}(q, C) = 1$ when $F_1(q, C) \geq \delta$ and $\hat{F}_{1(\delta)}(q, C) = 0$ otherwise. The cutoff F_1 score, with threshold $\delta \in (0, 1]$, is defined as $\bar{F}_{1(\delta)}(q, C) = F_1(q, C)$ when $F_1(q, C) \geq \delta$ and $\bar{F}_{1(\delta)}(q, C) = 0$ otherwise.*

Cover terminology. If a category C has the highest score for a query q (if necessary, ties are broken arbitrarily), and that score is not 0, we say that C *covers* q. We call a category that covers at least one query a *covering* category, and a branch containing a covering category is a *covering* branch. A set of categories is *independent* if no two categories are on the same branch (OCP categories are independent). Similarly, a set of queries are *independently-covered*, if each is covered by a different category, and the covering categories are independent. Observe that in unweighted instances (where, as noted earlier, all weights are assumed to be 1) with threshold functions the score equals the number of covered queries.

Note that, all functions defined above share the special case of $\mathcal{T}_{1,1}$ (Definition 1) (equivalent to setting $\delta = 1$ in Definitions 2 and 3), where a query q is covered by a category C only if $q = C$. We refer to this variant as the *Exact variant*.

Canonical form. Any category tree can be reduced to a canonical form, without decreasing the score, by (1) removing non-covering categories, (2) connecting the parent and children of any removed category, and (3) removing from category C and its descendants any element not contained in any query covered by C or categories below C (this may even improve the precision and the score). If a query is covered by multiple categories, one can assign arbitrarily a single category that is said to cover it, and then reduce it to a canonical form, as described above, w.r.t. this assignment. Similarly, adding new categories that do not affect the contents of the existing categories cannot decrease the tree score. This discussion applies analogously to category partitions.

Choices of parameters. For practical applicability, we focus on variants where $r = \Theta(1)$ and the similarity functions have threshold parameters. A low copy-bound ensures a concise categorization, and in many platforms, r is a small constant, typically, 1 (e.g., [1]). This parameter controls the trade-off between the score and the conciseness, and our parameterized bounds hint at a quantification of this trade-off. Threshold parameters capture the fact that, below a certain similarity score, a category has no utility. Without thresholds, trees that cover unacceptably poorly all queries may be mathematically preferable to trees that cover well a smaller number of queries. Nevertheless, to capture more tolerant settings, we also provide approximation bounds for polynomially small threshold parameters.

We also note that, in practice, errors in precision and recall have an asymmetric effect. For example, perfect recall with precision of $\frac{1}{2}$, enables the user to examine all relevant items to identify the best matches, while ignoring every other item. This may be acceptable, especially for smaller categories. However, in the analogous case of perfect precision and recall of $\frac{1}{2}$, other categories may or may not contain better matching items, and the user might waste time looking for non-existing or hard-to-find categories or be unaware of better options. It may, therefore, be tempting, in some applications, to require perfect recall. To that end, we examine this case separately and show that it admits the strictest inapproximability.

More generally, there exists a key tension between the recall and precision thresholds. Consider, as an extreme example, a recall threshold of 1, and a precision threshold of 0 (i.e., perfect recall with no precision requirement). For this variant, a tree consisting only of a root that contains all the elements is an optimal solution. At the other extreme, if we require perfect precision with no constraint on the recall, then an optimal solution is a tree where there is a leaf for each element, containing only that element, and a root connected directly to all the leaves. These edge cases illustrate the following intuitive phenomenon: increasing precision thresholds leads to more granular trees with smaller covering categories, whereas increasing recall thresholds generally produces a more coarse categorization with larger covering categories. These properties of the precision and recall thresholds are formalized and leveraged in our hardness proofs.

2.3 Examples of Problem Instances

We illustrate the OCT setting with $r = 1$ via the following toy examples, depicted in Figure 1. The figure presents two optimal solutions, computed by brute-force, corresponding to two different OCT variants, over the same input, provided on the left side. For convenience, as it is easier to perform arithmetic with integers, we provide integer weights, instead of normalizing into $[0, 1]$, since the normalization of the weights and scores does not affect the complexity of the problem or the performance ratio of the various solutions.

Observe that the overall weight of all four queries is 5, hence this is also an upper bound on the score of any tree for any variant over this input. In addition, observe that, since $r = 1$, we cannot add any branches to either of the depicted trees, without violating the copy-bound constraint.

▶ **Example 4.** The tree T_1, depicted in the middle of the figure, is the optimal solution for the $OCT^1(\mathcal{T}_{0.8,1})$ variant (i.e. precision 0.8 and perfect recall). The categories C_3 and C_4 cover the queries q_2 and q_3, respectively, as they are identical to these queries (and would cover them even for $\alpha = 1$). The category C_1 covers q_1 as its recall score is 1, and 5 out of the 6 items in C_1 are in q_1, hence the precision is $\frac{5}{6} > \alpha$. Note that, we must include f in C_1 since it appears in C_4, and removing f from both categories, would result in C_4 no longer covering q_3. Moreover, there is no incentive to place f elsewhere, since the score, when using a binary function, is not penalized for precision errors if the threshold is exceeded.

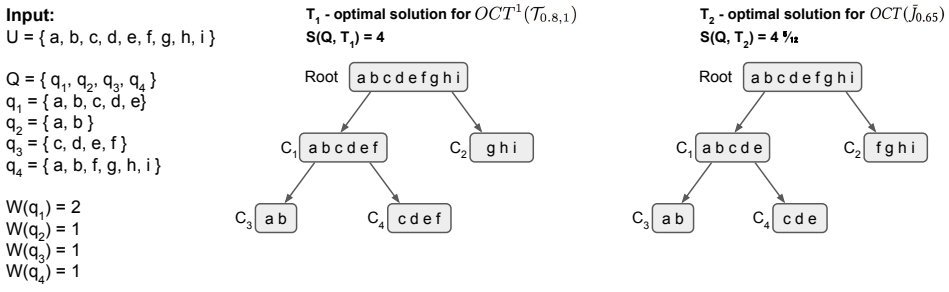

Figure 1 Optimal solutions for two OCT variants over the same input (where, for simplicity, the input weights are not normalized), depicted on the left side. The category tree, T_1, is an optimal solution for the $OCT^1(\mathcal{T}_{0.8,1})$ variant, where C_1 covers q_1, C_3 covers q_2, and C_4 covers q_3, with the overall score of $W(q_1) + W(q_2) + W(q_3) = 4$. The rightmost tree, T_2, is the optimal solution for the cutoff Jaccard variant with $\delta = 0.6$, where C_1 covers q_1 with the score of 1, C_2 covers q_4 with the score of $\frac{2}{3}$, C_3 covers q_2 with the score of 1, and C_4 covers q_3 with the score of $\frac{3}{4}$, resulting in the overall score of $W(q_1) \cdot 1 + W(q_2) \cdot 1 + W(q_3) \cdot \frac{3}{4} + W(q_4) \cdot \frac{2}{3} = 4\frac{5}{12}$.

As for the category C_2, its addition to the tree is optional, since it does not cover any query, despite all its items belonging to the uncovered query, q_4, as we can no longer achieve perfect recall without the items $\{a, b, f\}$. It is easy to verify that there is no way to cover q_4 by adding a matching category above or below C_1, such that the items $\{a, b, f\}$ would be shared by all categories, without decreasing the precision of other queries to values below the threshold.

▶ **Example 5.** We next discuss, T_2, the optimal solution for $OCT^1(\bar{J}_{0.65})$, the cutoff Jaccard variant with $\delta = 0.65$, depicted on the right side of Figure 1. It overlaps with T_1, except for the item f, which is placed in C_2 instead of C_4 and C_1. In this case, compared to the previously examined variant, since Jaccard variants allow for errors in both precision and recall, and also since we use a lower threshold, it is now possible to cover all queries, albeit with imperfect scores. Indeed, every non-root category in T_2 covers a query, as explained in the figure. Moreover, q_1 is the query of the maximal weight, hence it is not surprising that the optimal tree covers it with a perfect score, at the expense of errors in the covers of less significant queries. We note that, in practice, the same category often covers multiple queries. For instance, if we decrease the threshold from 0.65 to 0.4, then C_1 would also cover q_2, as its precision w.r.t. q_2 is exactly 0.4.

3 Preliminaries

We provide here known results and definitions, that we will use in our hardness proofs. We conclude the section by explaining how proofs are tailored to fit both OCT and OCP simultaneously, and discuss generalizing a tree to a DAG.

Notation. To simplify the presentation, we use a "soft-omega" notation, $\tilde{\Theta}(\cdot)$, to hide sub-polynomial factors. Whenever we state that a variant has inapproximability of $\tilde{\Theta}(n^c)$, for some constant $c \in (0, 1]$, this compact notation implies the more formal argument that, for any $\epsilon > 0$, this variant cannot be approximated within a factor of $O(n^{c-\epsilon})$. We note that a solution of score 1 can always be achieved by producing a single category that equals one of the queries (for differently weighted queries we will specifically select the query of the highest weight). Thus, $\tilde{\Theta}(n)$ is the strictest possible inapproximability factor, using this notation.

Complexity Assumptions. We next define the complexity class ZPP, as some of our results use the assumption $ZPP \neq NP$. It is known that $P \subseteq ZPP \subseteq NP$ and that $ZPP \subseteq BPP$, where BPP is the class of problems solvable by a randomized PTIME algorithm with a two-sided error.

▶ **Definition 6.** *The complexity class ZPP contains the problems for which there is a PTIME algorithm that outputs DO NOT KNOW with a probability of less than $1/2$, and outputs the correct answer with the remaining probability.*

MIS. We leverage reductions from the Maximum Independent Set problem (MIS) in *uniform hypergraphs*. In an r-uniform hypergraph, all (hyper)edges are vertex subsets of cardinality r. The special case of $r = 2$ is a graph.

▶ **Definition 7.** *In the Maximum Independent Set problem (MIS) in uniform hypergraphs, the input is a uniform hypergraph $G = (V, E)$, and the objective is to find a vertex set $S \subseteq V$ of maximum cardinality, subject to the constraint that no edge from E is contained in S.*

We have made use of the following known results for MIS, where $n = |V|$.

▶ **Theorem 8** ([11])**.** *The MIS problem in r-uniform hypergraphs, for constant $r \geq 2$, cannot be approximated below a $\tilde{\Theta}(n)$ factor, unless $ZPP = NP$.*

▶ **Theorem 9** ([27], [6])**.** *The MIS problem in graphs has inapproximability of $\tilde{\Theta}(n)$, unless $P = NP$. Moreover, for graphs of sufficiently large constant degree bound d, MIS is hard to approximate below a $\Theta(\frac{d}{\log^2 d})$ factor, unless $BPP = NP$. Furthermore, MIS is NP-hard even for regular graphs of degree 3.*

▶ **Theorem 10** ([7])**.** *For r-uniform hypergraphs with (not necessarily constant) maximum degree d, there exists a $PTIME$ algorithm producing an independent set of size $\Omega(\frac{n}{d^{\frac{1}{r-1}}})$.*

From Theorem 10, we derive the following lemma.

▶ **Lemma 11.** *Given an r-uniform hypergraph $G = (V, E)$, there exists a PTIME algorithm that produces an independent set in G of size $\Omega((\frac{|V|^r}{|E|})^{\frac{1}{r-1}})$.*

Proof of Lemma 11. The average degree of G is $\bar{d} = \frac{r|E|}{|V|}$. Let $V_1 \subseteq V$ denote the set of vertices in G whose degree is at most $d = 2\bar{d}$. A simple counting argument implies that $|V_1| \geq \frac{|V|}{2}$. Consider the sub-hypergraph G_1 of G induced by V_1. Computing G_1 is the first step of the algorithm.

In the second and last step, we apply over G_1 the algorithm from Theorem 10, which produces an independent set S. By Theorem 10, the size of this independent set is

$$|S| \geq \Omega(\frac{\frac{|V|}{2}}{d^{\frac{1}{r-1}}}) = \Omega(\frac{|V|}{(\frac{r|E|}{|V|})^{\frac{1}{r-1}}}) = \Omega((\frac{|V|^r}{|E|})^{\frac{1}{r-1}}).$$ ◀

Hard instances of MIS. When reducing from MIS, we will restrict ourselves to instances where the optimal solution is of size $\tilde{\Theta}(n)$. The $\tilde{\Theta}(n)$ inapproximability of MIS implies that this subset of inputs captures the maximal hardness. Accordingly, in our reductions, assuming this hard set of inputs, we will leverage the fact that one cannot find (in the worst case) an independent set of size $\Omega(n^\epsilon)$ for any $\epsilon > 0$.

Probabilistic Tools. We next define the Hypergeometric and Binomial distributions and present known tail bounds for both. These are useful in the analysis of our randomized reduction.

▶ **Definition 12.** *Consider sampling without replacement n uniformly random and independent samples from a set of N elements containing K special elements, and let X denote the number of special elements in the sample. Then, X is a hypergeometric random variable, denoted as $X \sim H(N, K, n)$, and its probability mass function is* $\Pr(X = i) = \frac{\binom{K}{i}\binom{N-K}{n-i}}{\binom{N}{n}}$.

▶ **Definition 13.** *Consider performing n independent experiments with success probability p. Let X denote the number of successful experiments. Then, X is a binomial random variable, denoted as $X \sim B(n, p)$, with probability mass function is* $\Pr(X = i) = \binom{n}{i} p^i (1-p)^{n-i}$.

We use the following tail-bound for the hypergeometric distribution.

▶ **Lemma 14** ([20]). *If $X \sim H(N, K, n)$, as defined in Definition 12, and letting $u = \frac{K}{N}$, then, for $t > 0$:*

$$\Pr(X \geq (u+t)n) \leq \left(\left(\frac{u}{u+t}\right)^{u+t}\left(\frac{1-u}{1-u-t}\right)^{1-u-t}\right)^{n}.$$

We also use the following Chernoff bound for the binomial distribution.

▶ **Lemma 15** (Chernoff Bound [23]). *If $X \sim B(n, p)$, as defined in Definition 13, then, denoting the expectation $\mu = np$, for $\delta \geq 1$:*

$$\Pr(X > (1+\delta)\mu) < \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu}.$$

OCP and DAGs. Finally, we explain how our hardness reductions for OCT were devised to also apply for OCP, as well as for the more general problem where one is allowed to produce a rooted DAG with analogous combinatorial constraints. Thus, while the hardness analysis focuses on OCT, *the bounds apply for the above two problems as well.*

Loosely speaking, an algorithm that produces a tree has all the capabilities it would need for producing a similar-score partition, along with several additional possibilities to increase the score. Hence, in our analysis, by bounding what is possible for constructing a tree, we also bound what is possible for constructing a partition.

Concretely, observe that, over any given input, any category partition can be transformed into a category tree of the same score, by connecting all the categories to a root. In particular, over any given input, the optimal score, that can be achieved by a category partition, cannot exceed the optimal score by a category tree. Importantly, in all examined variants, we ensure that our hardness bounds are derived over a subset of inputs for which there exists a category tree whose leaf categories induce a category partition of score $\tilde{\Theta}(n)$, implying that the optimal score of both problems is of at least this order, which is roughly maximal (the score for any input cannot exceed $n = |Q|$). It follows that all our approximation hardness bounds for OCT hold for OCP as well.

We note that our hardness proofs also apply to the more general problem where instead of a tree, one is allowed to produce any rooted DAG, maintaining the requirements that a category must contain all its descendants and for each element e there are at most r different paths from the root to a semi-leaf w.r.t. e (recall that a semi-leaf w.r.t. e is a most specific node to which e belongs). This follows from the fact the any such DAG can be converted to a valid tree solution, by removing edges, which does not affect the score.

4 Approximation Algorithm for the Exact Variant

Before diving into the hardness analyses, we provide a positive result based on PTIME approximation algorithms for the Exact variant of (weighted) $OCT^1(\mathcal{T}_{1,1})$ and $OCP^r(\mathcal{T}_{1,1})$. Note that for OCT the algorithm applies when $r = 1$, while for OCP it applies to any constant r. The Exact variant is of special interest because it is a special case of all variants pertaining to all examined similarity functions, where the error threshold is $\delta = 1$ (or $\alpha = \beta = 1$ for $\mathcal{T}_{\alpha,\beta}$). We note that in [3] we show empirically that this algorithm can solve real-world instances optimally, using as a subroutine modern weighted MIS solvers, and extend it to algorithms suited for the more general OCT variants. Moreover, as mentioned, the requirement $r = 1$ is the most prevalent in practice.

In Theorem 17 we prove that it is NP-hard to approximate this variant below a $\tilde{\Theta}(n)$ factor. Hence, one cannot provide any non-trivial approximation guarantees for the general case. Nevertheless, we devise an algorithm with an optimal approximation guarantee of $\tilde{O}(\bar{D})$, where $\bar{D} \in [0, n]$, referred to as the *average (weighted) degree* of the input, is a parameter relating to the number of intersections among the queries. Formally, we define a *conflict* as any pair of queries that intersect and neither is a subset of the other (for OCP only the former condition is relevant). The *degree* $d(q)$ of a query $q \in Q$ is defined as the number of conflicts in the input that contain q. The average degree is the weighted average of all query degrees in the input. That is, $\bar{D} = \frac{\sum_{q \in Q} W(q) \cdot d(q)}{\sum_{q \in Q} W(q)}$.

The proof of the following theorem and the description of the algorithms, which are based on the connection to the weighted MIS problem, appear in the full version of the paper [10].

▶ **Theorem 16.** *There exist $\tilde{\Theta}(\bar{D})$-approximation algorithms for $OCT^1(\mathcal{T}_{1,1})$ and $OCP^r(\mathcal{T}_{1,1})$. This factor is optimal (up to negligible factors), unless $P = NP$.*

The result above shows that the approximation hardness is strongly dependent on the average number of intersections between queries, and indeed, in the subsequent hardness analysis, the only practical bound corresponds to a special case where $\tilde{O}(\bar{D})$ is low (Theorem 17).

We remark that in all examined eBay datasets the average degree did not exceed $\log n$, even for high values of the maximum degree. However, we leave an in-depth empirical evaluation along with devising algorithms for other problem variants to future work.

5 Hardness of $OCT^r(\mathcal{T}_{\alpha,\beta})$

In this section, we prove approximation hardness bounds on $OCT^r(\mathcal{T}_{\alpha,\beta})$ for various ranges of the threshold parameters. We first provide a reduction from MIS to $OCT^r(\mathcal{T}_{\alpha,\beta})$ where $\alpha = \Theta(1)$ and $\beta > \frac{1}{2}$, proving $\tilde{\Theta}(n^{\frac{1}{r+1}})$ inapproximability (n is the number of queries, and r is the copy-bound), unless $ZPP = NP$. For the special case of $OCT^r(\mathcal{T}_{\alpha,1})$ we improve this bound to $\tilde{\Theta}(n)$. For $r = 1$, we strengthen these bounds by using a weaker theoretical assumption and also provide a bound for the case of queries of bounded size.

To prove that the $\tilde{\Theta}(n^{\frac{1}{r+1}})$ inapproximability extends to the case where $\beta \leq \frac{1}{2}$, we use a more involved randomized reduction, and also provide an analysis that captures sub-constant ranges of the threshold parameters to derive a $\tilde{\Theta}((\alpha^{(r+2)}\beta n)^{\frac{1}{r+1}})$ inapproximability bound.

The remainder of this section consists of two subsections, pertaining to the two reductions. Each subsection is further divided into the reduction from MIS, the hardness results it implies, and the intuition underlying the proof. We also explain why different reductions were necessary. All formal proofs are appear in the full version of the paper [10].

5.1 Special cases with $\beta > \frac{1}{2}$

We now describe and analyze the first reduction.

Reduction from MIS. Given an algorithm for $OCT^r(\mathcal{T}_{\alpha,\beta})$, denoted by A, with a (worst-case) approximation guarantee of γ, we devise an algorithm $R = R_A$ for MIS in $(r+1)$-uniform hypergraphs. We compute a lower bound on the size of the independent set (IS) that R produces as a function of the approximation guarantee γ. This implies a lower bound on γ, below which R would produce an IS of size $\Theta(poly(n))$ (n is the number of vertices in the hypergraph), contradicting the hardness of MIS.

The algorithm R consists of a sequence of three procedures, R_1, R_2, and R_3:

1. Given an $(r+1)$-uniform hypergraph, $G = (V, E)$, R_1 transforms it into an instance Q of $OCT^r(\mathcal{T}_{\alpha,\beta})$. The universe of elements for Q consists of three types of elements: an *edge element* for every edge in E, *padding elements*, and $\frac{1-\beta}{2\beta-1}n^r$ *joint elements*. Specifically, for each vertex $v \in V$, we construct a query, q_v, such that $Q = \{q_v \mid v \in V\}$. Every query contains all the joint elements. Moreover, each query, q_v, also contains all the edge elements that correspond to edges incident to v in G. Finally, we add to every query as many unique padding elements as necessary, such that the size of the query is $\frac{n^r}{2\beta-1}$. Every padding element appears in only one query. It follows that every query contains $\frac{1-\beta}{2\beta-1}n^r$ joint elements and $\frac{\beta}{2\beta-1}n^r$ non-joint elements.
2. R_2 consists simply of running A over Q. Let T denote the category tree A outputs.
3. R_3 produces an IS $S \subseteq V$, as follows. Let $\hat{\mathcal{C}}$ denote the set of categories that consists of the lowest (closest to the leaves) covering category of every covering branch in T. Let $\hat{Q}$ denote a set of queries constructed by selecting arbitrarily from every category in $\hat{\mathcal{C}}$ a single query that it covers. Observe that $\hat{Q}$ is an r-weak partition. We denote by $\hat{V} = \{v \in V \mid q_v \in \hat{Q}\}$ the set of vertices that corresponds to the queries in $\hat{Q}$, and denote by $\hat{G}$ the sub-hypergraph of G induced by $\hat{V}$. R_3 computes $\hat{G}$ and applies over it the algorithm from Lemma 11, producing an IS S, which is the final output.

For simplicity, we ignore rounding issues, as rounding the parameters to the nearest rational fraction has a negligible effect, and the number of vertices, n, can be manipulated by adding vertices that are connected to all other vertices.

Hardness bounds. The construction above implies the following hardness bounds.

▶ **Theorem 17.** *The $OCT^r(\mathcal{T}_{\alpha,\beta})$ problem, with constant $\alpha \in (0, 1]$ and $\beta > \frac{1}{2}$, cannot be approximated below a $\tilde{\Theta}(n^{\frac{1}{r+1}})$ factor, unless $ZPP = NP$. For $OCT^r(\mathcal{T}_{\alpha,1})$ this is improved to $\tilde{\Theta}(n)$. For $r = 1$ these bounds hold for the assumption $P \neq NP$. Moreover, $OCT^1(\mathcal{T}_{\alpha,1})$ with maximum query size $d = \Theta(1)$ is hard to approximate below a $\tilde{\Theta}(\alpha d)$ factor, unless $BPP = NP$. Lastly, $OCT^1(\mathcal{T}_{1,1})$ is NP-hard even when all queries are of size exactly 3. The bounds for $OCT^1(\mathcal{T}_{\alpha,1})$ hold even when query intersections are of cardinality at most 1.*

We explain below the intuition underlying the reduction and the proof outline.

Intuition. When $\beta = 1$, there are no joint elements, and each query consists of all the relevant edge elements along with padding elements that ensure its size is exactly n^r. In the Exact variant, every covering branch in T covers exactly one query, and this independently-covered set of queries corresponds to a set of vertices that is independent in G. Therefore, if T covers $poly(n)$ queries, we can find an IS of the same size in G.

When relaxing the precision threshold, α, it becomes possible for the same branch to cover multiple queries. As we want to select one query from each branch to ensure independence, it may no longer be the case that the number of covering branches is of the same order as the solution. Nevertheless, on every covering branch, the covering category C closest to the root must contain all elements of all covered queries on the same branch. If there are many such queries, then C would not satisfy the precision requirement. Intuitively, a branch can cover no more than $O(\frac{1}{\alpha}) = O(1)$ queries. It follows that the set of independently-covered queries, $\hat{Q}$, is of the same order as the score of the tree in this case as well.

Matters are more complicated when the recall threshold β is also relaxed. It is no longer the case that an independently-covered set of queries corresponds to an IS. It is now possible for $(r+1)$ such queries to correspond to an edge in G, as the cover of at least one of these queries can avoid containing that edge-element. Without including joint elements in the reduction, the cover of every query could omit a constant fraction of the edge elements, which amounts to $O(n^{r+1})$ edge elements, such that a large independently-covered set of queries could correspond to even a very dense subgraph in G.

To that end, we show that a cover of a query must include a joint element per every omitted edge element. Since all queries share the joint elements, this hinders the ability of covers in other branches to omit edge elements. It follows that the total number of edges in a subgraph corresponding to an independently-covered set of queries contains at most $O(n^r)$ edges which is the total number of joint elements. Therefore, a tree of high score would correspond to a large vertex set which is also sparse. From this "almost IS" we can derive a somewhat smaller, but still polynomial-sized, IS, using Lemma 11.

Adding joint elements may allow covering more queries on a single branch, as including joint elements in a category contributes to its potential cover of all queries. However, we show that the number of covered queries by a single branch is bounded by a constant.

We ensure that the optimal OCT solution is of score $\tilde{\Theta}(n)$. Thus, if the approximation factor of A is low, the eventually derived IS is large. In particular, we ensure that the tree contains a category partition of the same score so that all bounds also hold for OCP. Observe that the maximum IS in G induces the category partition where every category covers a single query pertaining to a vertex in the set, with all covers including all of the non-joint elements and no joint elements. The categories in this partition satisfy the recall condition as narrowly as possible. Intuitively, this construction means that, while joint elements help an algorithm to an extent, beyond that it must make progress on the MIS problem.

5.2 General threshold parameters

We have examined so far the hardness of various special cases of $OCT^r(\mathcal{T}_{\alpha,\beta})$ where the recall threshold is $\beta > \frac{1}{2}$. In particular, we proved for $\frac{1}{2} < \beta < 1$ inapproximability of $\tilde{\Theta}(n^{\frac{1}{r+1}})$. We next devise a more involved construction to show that this result extends to $\beta \leq \frac{1}{2}$ and also provide more general bounds for polynomially small threshold parameters. We note that since the modified construction is randomized, the bound derived for $r = 1$ does not hold under the weaker assumption of $P \neq NP$, unlike in the first construction.

Modifications. To facilitate a precise discussion, we first define, given a subset of queries Q', the multiplicity $M_{Q'}(e) = |\{q \in Q' \mid e \in q\}|$ of an element e in Q' as the number of queries in Q' that e appears in. The reduction used for Theorem 17 becomes ineffective because in the OCT instance constructed by R, the set of joint elements makes up a $(1-\beta)$-fraction of every query, and for $\beta \leq 1/2$ the set of joint elements becomes large enough, such that a category, that consists exactly of this set, covers all queries, yielding the optimal tree score.

To fix this, we need to alter the construction such that joint elements are not shared by all queries. We want to limit the number of joint elements with high multiplicity in any large query set (we will formalize this high-level statement with concrete thresholds in Lemma 21), to make it hard for a single branch to cover it while retaining properties essential for the hardness proof.

Concretely, we want any single joint element to be shared by many queries, and for a $(1-\beta)$-fraction of every query to consist of joint elements, so that the tree that corresponds to the optimal MIS solution narrowly exceeds the recall requirements. This requires using more joint elements. However, having more joint elements can make $\hat{G}$ less sparse, reducing the size of the produced IS. To achieve these desired properties while minimally increasing the number of joint elements, we devise a randomized reduction. Moreover, we parameterize it to efficiently capture sub-constant ranges of α and β, to aim for a slower decay in the hardness bound, as these thresholds are decreased.

Generalized reduction from MIS. Our revised MIS algorithm denoted by R' consists of a sequence of three procedures, R'_1, R'_2 and R'_3. To avoid a convoluted presentation, we reuse some of the notation, initially defined in the context of the first algorithm R.

1. Given an $(r+1)$-uniform hypergraph, $G=(V,E)$, R'_1 transforms it into an instance $Q=\{q_v \mid v \in V\}$ of $OCT^r(\mathcal{T}_{\alpha,\beta})$. Each query, q_v, contains all the edge elements that correspond to edges incident to v in G, and as many unique padding elements as necessary, such that the number of non-joint elements in every query is exactly n^r. Finally, we distribute $(\frac{1}{\beta}-1)\frac{\log^3 n}{\alpha}n^r$ distinct joint elements to queries via the following randomized scheme. We draw uniformly randomly $(\frac{1}{\beta}-1)n^r$ partitions of Q into $\frac{\log^3 n}{\alpha}$ subsets, each of size $\frac{\alpha n}{\log^3 n}$. Let $\hat{\mathbb{P}}$ denote this set of partitions. In every partition, $\hat{p} \in \hat{\mathbb{P}}$, every set, $\hat{s} \in \hat{p}$, in that partition is assigned a distinct joint element to be included in all queries in the set. Note that the size of each query is now exactly $\frac{n}{\beta}$.
2. The procedure R'_2, same as R_2, runs over Q the given $OCT^r(\mathcal{T}_{\alpha,\beta})$ algorithm A with an approximation guarantee factor of γ. Let T denote the category tree A outputs.
3. Finally, R'_3 is the same as R_3, except for the following modification: if there is a branch in T that covers more than $\tilde{\Theta}(\frac{1}{\alpha})$ queries, then it outputs DO NOT KNOW, and otherwise proceeds as R_3 to produce an IS S.

Generalized hardness bounds. We now state the approximation bounds implied by the revised reduction, followed by the intuition underlying the proof.

▶ **Theorem 18.** *The $OCT^r(\mathcal{T}_{\alpha,\beta})$ problem cannot be approximated below a $\tilde{\Theta}((\alpha^{(r+2)}\beta n)^{\frac{1}{r+1}})$ factor, unless $ZPP=NP$.*

Intuition. The most significant component in the proof is the following technical Lemma.

▶ **Lemma 19.** *W.p. $1-o(1)$ (over the choices of partitions in $\hat{\mathbb{P}}$) the maximum number of queries in Q a single branch (in any tree) can cover is $\tilde{O}(\frac{1}{\alpha})$.*

We wish to show that precision cannot be maintained past a certain number of covered queries on a branch. We use the term *relevant cover* of a query q, to refer to the intersection of q with its covering category C, with the *relevant cover size* being $|q \cap C|$. One must be careful in selecting the query for which the precision condition is invoked, to derive a tight bound. On the one hand, we aim to select a query covered close to the root, so that its

covering category contains the covers of many other queries. On the other hand, we want to select a query whose relevant cover is small. Thus, we first prove that for any branch, there exists a query q covered by C, such that at least a constant fraction of the covered queries on the branch are covered by C or a lower category, and that the average relevant cover size of these queries is smaller than the relevant cover size of q by at most a logarithmic factor.

▶ **Lemma 20.** *Given a branch B that covers k' queries, there exists a query q covered by a category C in B, with the following two properties:*

1. *the set of queries, Q_k, covered by C or categories below C is of cardinality $k = \Theta(k')$.*
2. *let $d \in [1, \frac{1}{\beta}]$ denote the value for which the average relevant cover size of queries in Q_k is $n^r d$, then the relevant cover size of q is at most $(2 \log n) n^r d$.*

Given q and C as in Lemma 20, we derive from the precision condition an upper bound on $|C|$. On the other hand, C contains the union of the k covers of the queries in Q_k, and we show that for $k = \tilde{\omega}(\frac{1}{\alpha})$, the union of the k covers, and thereby C must contain many elements, beyond the upper bound, resulting in a contradiction. The key to proving that C contains many elements is bounding the multiplicity of the joint elements in Q_k. If all elements had constant multiplicity, then an α precision threshold implies that, when the relevant covers are on average of roughly the same size as the relevant cover of q (which is the case following Lemma 20), the number of covered queries is $O(\frac{1}{\alpha})$. To that end, we show that the multiplicity of almost every joint element in Q_k does not exceed $\tilde{O}(\frac{k}{\alpha})$.

▶ **Lemma 21.** *For any set Q_k of $k = \omega(\frac{\log^3 n}{\alpha})$ queries, w.p. $1 - o(1)$, there are at most $\frac{n^r}{2}$ partitions in $\hat{\mathbb{P}}$ where a joint element is assigned to more than $\theta = \frac{\alpha k}{8 \log n}$ queries in Q_k.*

The proof of Lemma 21 consists of a combination of probabilistic arguments. We first prove that this θ bound on the number of partitions holds for a uniformly randomly selected set of k queries w.p. $1 - o(n^{-k})$. Then, by using a union bound argument, it will follow that this bound holds for any selection of k queries w.p. $1 - o(1)$.

To prove the bounds of Lemma 21 for a randomly selected set Q_k of k queries, observe that a joint element is shared by polylogarithmically less than a $\frac{1}{\alpha}$-fraction of the queries in Q. Therefore, its expected multiplicity in Q_k would constitute the same fraction. To bound the probability of significantly deviating from this expectation, we show that the multiplicity of any joint element in Q_k is a hypergeometric random variable, and use a tail bound. Following a different union bound argument, this bound on the probability is extended over every joint element assigned in a given partition in $\hat{\mathbb{P}}$. Finally, since the partitions in $\hat{\mathbb{P}}$ are chosen independently, we use a Chernoff bound to derive an upper bound, that holds with high probability, on the number of partitions in $\hat{\mathbb{P}}$ in which a joint element with high multiplicity was assigned. We show that if these deviations occur sufficiently rarely, as stated in Lemma 21, then the cardinality of C increases as a function of k, deriving the bound $k = \tilde{O}(\frac{1}{\alpha})$.

6 Other Variants

So far we have proven hardness of $OCT^r(\mathcal{T}_{\alpha,\beta})$. In this section, we provide approximation hardness bounds for the remaining OCT variants, via reductions from $OCT^r(\mathcal{T}_{\alpha,\beta})$ and Theorems 17 and 18.

We first show that the $\tilde{\Theta}(n^{\frac{1}{r+1}})$ bound of $OCT^r(\mathcal{T}_{\alpha,\beta})$ with constant thresholds extends to the threshold versions of Jaccard and F_1 scores, with similar inapproximability for sub-constant thresholds as well. We then use these results to derive bounds for the cutoff versions of these functions, which only differ for $\delta = o(1)$.

We formulate our proofs schematically, such that they may be applied to threshold and cutoff variants of other functions.

For threshold functions, we derive the following bounds.

▶ **Theorem 22.** *The variants* $OCT^r(\hat{J}_\delta)$ *and* $OCT^r(\hat{F}_{1(\delta)})$ *cannot be approximated below a* $\tilde{\Theta}((\delta^{r+3}n)^{\frac{1}{r+1}})$ *factor, unless* $ZPP = NP$. *For* $r = 1$, *we have* $\tilde{\Theta}(\sqrt{n})$ *inapproximability, assuming* $P \neq NP$, *for* $OCT^r(\hat{J}_\delta)$ *with* $\delta > \frac{1}{2}$ *and* $OCT^r(\hat{F}_{1(\delta)})$ *with* $\delta > \frac{2}{3}$.

Finally, we provide bounds for cutoff functions, that follow from Theorem 22.

▶ **Theorem 23.** *The variants* $OCT^r(\bar{J}_\delta)$ *and* $OCT^r(\bar{F}_{1(\delta)})$, *with* $\delta \in [0,1)$, *have* $\tilde{\Theta}((\delta^{2r+4}n)^{\frac{1}{r+1}})$ *inapproximability, unless* $ZPP = NP$. *For* $r = 1$, *we have* $\tilde{\Theta}(\sqrt{n})$ *inapproximability, assuming* $P \neq NP$, *for* $OCT^r(\bar{J}_\delta)$ *with* $\delta > \frac{1}{2}$ *and* $OCT^r(\bar{F}_{1(\delta)})$ *with* $\delta > \frac{2}{3}$.

7 Related Work

The construction of category trees/taxonomies has been studied in multiple domains, including e-commerce, document management, and question answering [9, 24, 12]. Many algorithms have been devised for automating taxonomy construction [17, 9, 18] and maintenance, [22, 24, 26] employing different clustering approaches [9, 17], as well as crowdsourcing [21].

In the lines of work specified above, the quality of the resulting taxonomy is assessed along the following two dimensions.

The first dimension of quality assessment is user-study [9, 17], an evaluation which we incorporate w.r.t. our model in the complementary empirical work [3]. This evaluation is naturally entirely subjective.

The second dimension, which is the focus of the present paper, is the similarity of the resulting category tree to a given (combinatorially unrestricted) ground-truth set of items/documents. For example, the F_1 score used in [17, 9, 18] is a variant (without a threshold) of our corresponding F_1 measure for $r = 1$. Similarly, [22] computes recall and F_1 scores for the resulting trees, also with $r = 1$.

To our knowledge, however, no previous work investigates the theoretical complexity of the optimization problem of computing the tree of the highest score. The score is only used as an evaluation measure, to which the algorithm is oblivious. This approach, to an extent, is loosely justified by our worst-case bounds. Nevertheless, we show in [3] and [4], that leveraging the relation we outlined in Section 4 to the weighted MIS problem, allows solving well (and, in some cases, optimally) real-world problem instances, via extensively studied MIS solvers.

Our model differs from clustering models [19, 13] that typically focus on item-similarity, optimizing the similarity within each cluster or the dissimilarity across clusters. Moreover, these models are commonly defined by pairwise similarities, while our model also considers relations of a higher order. Thus, closest to our work in this domain is the field of hypergraph partitioning (clustering) [14, 16]. Specifically, the OCP problem with copy-bound $r = 1$ corresponds to seeking a partition of the vertices that maximizes the weight of (hyper)edges for which there is a similar set in the partition. Relaxing the copy-bound corresponds to overlapping clusters. Importantly, this relation between hypergraph clustering and our model is different from the more artificial relation leveraged in our reductions, where we cluster the hypergraph edges, instead of the vertices. Nevertheless, our proposed framework differs from existing models in several aspects. Notably, hypergraph clustering typically studies a

multi-way cut problem, intending to minimize the weight of the cut edges. Recently [16] suggested that there is a benefit in quantifying how an edge is cut, in terms of which subsets of its vertices are clustered together. Our work is relevant in that respect, as we quantify how similar these subsets are to the original edge.

A work resembling ours in a different aspect is [25], where the objective is to maximize the edge weights inside the cluster (we also maximize the covered "demand", instead of the less natural minimization of uncovered demand). However, the models of [16] and [25] (and many others [15, 8]) are easier to approximate, due to principal technical differences (e.g., bounds on the size and number of clusters), and we are not aware of clustering research that resembles our model or bounds.

8 Conclusion

In this paper, we studied the hardness of computing categorizations with a bounded number of possible repetitions, that best capture a given collection of item sets. We defined a model that captures various practical settings and proved inapproximability results for multiple variants and special cases. We also provided an algorithm for the Exact variant with an approximation guarantee that depends on finer input parameters.

An interesting direction for future work would be to identify more special cases that admit improved performance. Another intriguing avenue of exploration is determining for cases where we showed $\tilde{\Theta}(\sqrt{n})$ hardness, whether one can devise algorithms with matching approximation guarantees or prove stronger bounds.

References

1 https://export.ebay.com/en/start-sell/selling-basics/seller-fees/fees-optional-listing-upgrades/.

2 Rakesh Agrawal, Amit Somani, and Yirong Xu. Storage and querying of e-commerce data. In *VLDB*, pages 149–158, 2001.

3 Uri Avron, Shay Gershtein, Ido Guy, Tova Milo, and Slava Novgorodov. Category Tree Construction from Search Queries in E-Commerce. *https://slavanov.com/research/concat_tr.pdf*.

4 Uri Avron, Shay Gershtein, Ido Guy, Tova Milo, and Slava Novgorodov. ConCaT: Construction of Category Trees from Search Queries in E-Commerce. In *ICDE*, 2021.

5 Slobodan Beliga, Ana Meštrović, and Sanda Martinčić-Ipšić. An overview of graph-based keyword extraction methods and approaches. *JIOS*, 39(1):1–20, 2015.

6 Amey Bhangale and Subhash Khot. UG-hardness to NP-hardness by Losing Half. In *CCC*, 2019.

7 Yair Caro and Zsolt Tuza. Improved lower bounds on k-independence. *Journal of Graph Theory*, 15(1):99–107, 1991.

8 Karthekeyan Chandrasekaran, Chao Xu, and Xilin Yu. Hypergraph k-cut in randomized polynomial time. *Mathematical Programming*, pages 1–29, 2019.

9 Shui-Lung Chuang and Lee-Feng Chien. A practical web-based approach to generating topic hierarchy for text segments. In *CIKM*, page 127–136, 2004.

10 Shay Gershtein, Uri Avron, Ido Guy, Tova Milo, and Slava Novgorodov. On the Hardness of Category Tree Construction (full). *https://slavanov.com/research/icdt22_full.pdf*.

11 Thomas Hofmeister and Hanno Lefmann. Approximating maximum independent sets in uniform hypergraphs. In *Proc. of MFCS*, pages 562–570, 1998.

12 Yi-Hsiang Hsieh, Shih-Hung Wu, Liang-Pu Chen, and Ping-Che Yang. Constructing hierarchical product categories for e-commerce by word embedding and clustering. In *IRI*, pages 397–402, 2017.

13 Anna Huang. Similarity measures for text document clustering. In *NZCSRSC*, volume 4, pages 9–56, 2008.

14 George Karypis, Rajat Aggarwal, Vipin Kumar, and Shashi Shekhar. Multilevel hypergraph partitioning: applications in VLSI domain. *VLSI*, 7(1):69–79, 1999.

15 Tom Leighton, Fillia Makedon, and SG Tragoudas. Approximation algorithms for VLSI partition problems. In *ISCAS*, pages 2865–2868, 1990.

16 Pan Li and Olgica Milenkovic. Inhomogeneous hypergraph clustering with applications. In *NIPS*, pages 2308–2318, 2017.

17 Kunal Punera, Suju Rajan, and Joydeep Ghosh. Automatically learning document taxonomies for hierarchical classification. In *Proc. of WWW*, 2005.

18 Cécile Robin, James O'Neill, and Paul Buitelaar. Automatic taxonomy generation - a use-case in the legal domain, 2017. `arXiv:1710.01823`.

19 Lior Rokach and Oded Maimon. Clustering methods. In *Data mining and knowledge discovery handbook*, pages 321–352. Springer, 2005.

20 Matthew Skala. Hypergeometric tail inequalities: ending the insanity, 2013. `arXiv:1311.5939`.

21 Yuyin Sun, Adish Singla, Dieter Fox, and Andreas Krause. Building hierarchies of concepts via crowdsourcing. *CoRR*, abs/1504.07302, 2015. `arXiv:1504.07302`.

22 Lei Tang, Jianping Zhang, and Huan Liu. Acclimatizing taxonomic semantics for hierarchical content classification. In *Proc. of KDD*, pages 384–393, 01 2006.

23 Eli Upfal. *Probability and computing: randomized algorithms and probabilistic analysis*. Cambridge university press, 2005.

24 Quan Yuan, Gao Cong, Aixin Sun, Chin-Yew Lin, and Nadia Magnenat Thalmann. Category hierarchy maintenance: a data-driven approach. In *SIGIR*, pages 791–800, 2012.

25 Wenxing Zhu and Chuanyin Guo. Local search approximation algorithms for the complement of the min-k-cut problems. 2010.

26 Hai Zhuge and Lei He. Automatic maintenance of category hierarchy. *Future Generation Computer Systems*, 67:1 – 12, 2017.

27 David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. In *Proc. of STOC*, pages 681–690, 2006.