ConCaT: Construction of Category Trees from Search Queries in E-Commerce

Uri Avron
Tel Aviv University
uriavron@mail.tau.ac.il

Shay Gershtein
Tel Aviv University
shayg1@mail.tau.ac.il

Ido Guy
eBay Research
idoguy@acm.com

Tova Milo
Tel Aviv University
milo@cs.tau.ac.il

Slava Novgorodov
eBay Research
snovgorodov@ebay.com

2021 IEEE 37th International Conference on Data Engineering (ICDE) | 978-1-7281-9184-3/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICDE51399.2021.00308

***Abstract*— Category trees play a central role in e-commerce platforms, enabling browsing-style information access. Building category trees that reflect users' dynamic information needs is a challenging task, mostly carried out by in-house taxonomists. This manual construction often leads to trees that are lacking or outdated since it is hard to keep track of market trends, seasonal changes, holidays, and special events.**

To support a browsing experience that better matches the user information needs, and to considerably reduce the manual work performed by taxonomists, we propose CONCAT - a system that leverages the demand-based nature of the query paradigm to automatically build a category tree that is maximally similar to the result sets for search queries. We demonstrate the effectiveness of CONCAT on real-world data, taken from a large e-commerce platform, by interacting with the ICDE'21 participants who act both as the consumers and the taxonomists.

I. INTRODUCTION

Product category trees (also called, in some cases, product taxonomies), play a central role in large-scale e-commerce platforms, such as eBay, Walmart, Amazon, or Taobao. Such taxonomies enable browsing-style information access and often drive the mapping of items into a structured set of key-value pairs (e.g., memory size for smartphones or sleeve length for shirts). Building and maintaining product category trees, that reflects users' dynamic information needs and interests, is a challenging task, mostly carried out by in-house taxonomists. This manual construction often leads to category trees that are lacking or outdated since it is hard to keep track of market trends, seasonal changes, holidays, and special events [1].

An alternative, more flexible information access paradigm is the search-based paradigm, where users submit queries to the platform's search engine. This mechanism allow users to formulate a more precise query that better reflects their intent [2]. Nevertheless, for many users it is often hard to formulate a precise query, resulting in high usage statistics of category trees as a viable alternative.

To support a browsing experience that better matches the user information needs, and to considerably reduce the manual work performed by taxonomists, we propose CONCAT - a system that leverages the demand-based nature of the query paradigm to automatically build a category tree that is maximally similar to result sets for user queries.

From a practical point of view, such a category tree has several advantages. First, it more effectively addresses the needs of consumers. Second, our approach provides an automated solution, which saves taxonomists valuable time and effort. Lastly, the real-time performance of CONCAT allows adapting the tree with high-frequency, to ensure that the categories are updated to match the most recent market trends.

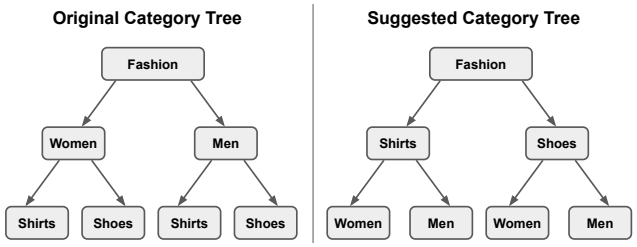

Fig. 1. Sample category trees for the electronics domain

Consider the following toy example in the context of an online fashion store, that sells shirts and shoes. The left side of Figure 1 depicts the existing category tree, where the two top categories are "Women" and "Men", as is traditionally the case in fashion taxonomies. Correspondingly, the two subcategories, "Shirts" and "Shoes", appear both under the "Women" and "Men" categories. However, there are large subsets of items, both in the shoes and shirts categories, that match both genders. Since in real-world platforms, each item is, typically, allowed to appear in at most one branch of the category tree, to ensure a compact categorization, a consequence is that the shared items are partitioned across the two subtrees, corresponding to the two genders.

Assume that the workload of user queries submitted to the site's search engine indicates that the most common queries are "shirts" and "shoes". The result sets for such queries should arguably be all the shirts/shoes regardless of the assigned gender (in the actual preprocessing stage of CONCAT, *we use more sophisticated methods to ensure that the result sets are indicative of the user intent). However, when a consumer, instead of using the search engine, uses the category tree, to browse the selection of relevant shirts, she must examine two non-adjacent categories in the tree (and be aware of a second relevant category existing). Moreover, even for the query "women's shirts", the relevant category is incomplete, as some shirts, that match both genders, are assigned to the men's shirts category.*

To address both issues, and provide a solution where users can more frequently view a category that consists of all the relevant items they seek in a given search session, the platform

978-1-7281-9184-3/21/$31.00 ©2021 IEEE DOI
10.1109/ICDE51399.2021.00308

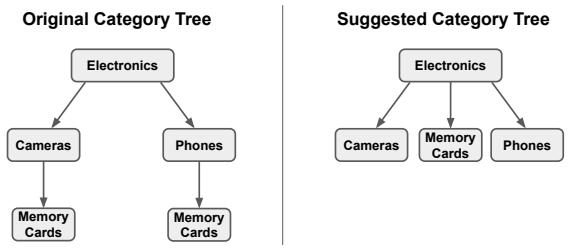

Fig. 2. Sample category trees for the fashion domain

should instead use the tree depicted on the right side of Figure 1. The category "shirts" contains all the shirts relevant for both genders. In addition, if users are indeed interested in viewing a category that matches a specific gender, then these are still available below the more general "shirts" category. Note that now, in the new tree, there are no dedicated categories for men or women that contain all the products relevant for a specific gender. However, if the search queries indicate that such result sets are rarely sought, then this absence is negligible compared to the improvement for more common queries.

The following example demonstrates a similar scenario, where a somewhat different modification is needed.

Consider an online electronics store that sells cameras, phones, and various accessories for these products, such as memory cards. The existing tree, depicted on the left side of Figure 2, has two separate categories for memory cards: "Cameras"→"Memory Cards" and "Phones"→"Memory Cards". However, all memory cards are suitable for both product types.

Assume that the most searched query is "memory cards", and that complete sets of all accessories for a given product type are rarely searched. The tree depicted on the right side of Figure 2 would then better serve the consumer needs, as it has a separate category containing all memory cards. The fact that these items no longer appear under the general categories has little effect, as the memory cards are rarely searched as part of more general item sets.

The trees presented in Figures 1 and 2 are based on real categories trees in large e-commerce platforms. To handle the large workloads, the trees are typically constructed by several taxonomists that work in parallel, each assigned to categorize a different domain, which leads to deficiencies as illustrated above. Our system allows platforms to create trees that fix such discrepancies. Moreover, the automatic construction has real-time performance, compared to manual error-prone constructions that may take weeks or months.

Model The problem we study is called the Optimal Category Tree problem (OCT). The input for OCT consists of result sets for search-queries. The solution space is composed of rooted trees, in which every node contains a subset of the items, and represents a category. In a valid tree, every non-leaf category contains as a subset the union of the items in its child categories. Moreover, every item appears in only one most-specific category (typically, a leaf-category) along with all its ancestor categories. The ideal goal is to have in the tree, for every query, a category (tree node) that is very similar to its result set. For some queries, it may be more important to have a matching category, therefore the input sets may be weighted to reflect the significance of the corresponding queries. This model has multiple variants, based on the set-similarity function, used by the platform, for measuring the similarity of an input set and a category (e.g., the *Jaccard similarity index* or *the* F_1 *score*). The score of a tree, w.r.t. a given query, is the similarity score of its result set and the category that is most similar to it. The objective is to produce a tree that maximizes the weighted (w.r.t. the query weights) sum of the scores over all the queries.

We have studied the model above both from a theoretical and a practical perspective. Namely, we proved in [3] strict hardness of approximation bounds for this setting, as well for several of its generalizations and related problem. Whereas here we provide two heuristic algorithms, which apply generically to multiple problem variants, along with improved algorithms, with theoretical guarantees, for special cases. The effectiveness of these algorithms was verified over large datasets, both public and private, from large real-world e-commerce platforms. These algorithms serve as the core modules in the CONCAT framework, as described in the sequel.

Demonstration Overview We demonstrate the operation of CONCAT over real-world e-commerce data. Our demonstration reenacts a scenario where an e-commerce platform redesigns its online store and wishes to construct a category tree, that most closely matches the item sets sought by users, as indicated by the search queries. The audience will play two separate roles: (1) consumers who submit queries to the platform's search engine, and (2) taxonomists that utilize CONCAT to construct trees that best match these queries.

II. TECHNICAL BACKGROUND

We first formally define the Optimal Category Tree problem (OCT) and briefly discuss our key theoretical results and corresponding algorithms.

A. Formal Problem Definition

Input The Optimal Category Tree problem (OCT) takes as input $\langle Q, W \rangle$, where $Q \subseteq 2^U$ is a set of n sets over a universe of items U, and each set in Q is assigned a non-negative weight by $W : Q \mapsto \mathbb{R}$. Each input set typically corresponds to a result set for a user query, and its weight indicates its prominence (e.g., based on query frequency). Since our model fully captures any query by its result set, we use the term *query* when referring to each set in Q.

Problem variants The OCT problem has multiple variants based on the *similarity function*, $\mathcal{S} : [2^U] \times [2^U] \mapsto [0, 1]$, used for the objective function. The variant with similarity function $\mathcal{S}$ is denoted by $OCT(\mathcal{S})$. To capture the fact that, below a certain similarity score, a category has no utility w.r.t. the given query, since it is not recognizable as a relevant match by the user, we extend the similarity functions with a threshold

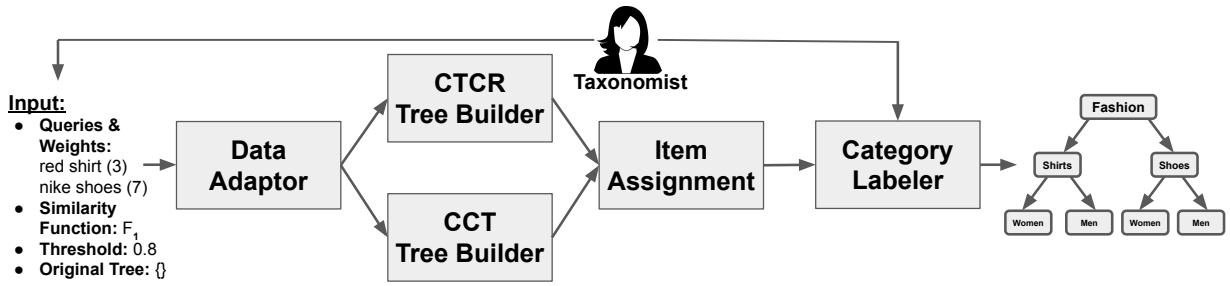

Fig. 3. System Architecture

parameter $\delta \in (0, 1]$, such that similarity scores below the threshold are rounded down to 0.

Solution space A solution to an OCT instance is a *category tree* - a rooted tree, where every node contains a subset of U and represents a *category*. A valid category tree satisfies the following two requirements. First, every non-leaf category contains the union of the item sets in its child categories (and possibly other items). The root of the tree, thus, contains all the items that appear in any category, with the categories becoming more specific (smaller), as one moves down the tree towards the leaves. Second, each item in the tree belongs to exactly one most-specific category, along with all its ancestors. Thus, every item appears only in categories that are consecutively placed on some branch in the category tree, where a branch is a simple path from the root to a leaf.

Objective Given a query, $q \in Q$, and a category tree, T, the *similarity score* of a category $C \in T$ over q is $\mathcal{S}(q, C)$. The score of T over this query is $S(q, T) = \max_{C \in T} \mathcal{S}(q, C)$. This definition captures the fact that a user seeks the category that most closely matches her (implicit) query. The overall score of T is defined as $S(Q, W, T) = \sum_{q \in Q} W(q) \cdot S(q, T)$. The weights are reflected in the objective function, such that it is preferable to have matching categories for queries of higher weight. The objective is to produce a category tree of the maximum score: $\arg\max_T S(Q, W, T)$.

Cover terminology. In the sequel, we say that a category *covers* a query if their similarity score exceeds the threshold. A query is *covered* if any category in the tree covers it.

B. Theoretical Bounds and Algorithms

In [3], we proved that all examined problem variants have strict NP-hard inapproximability bounds, leaving room only for impractical worst-case guarantees. Moreover, we showed that the hardness also applies to more general settings, thus, we cannot hope to provide practical worst-case guarantees, even by a reasonable relaxation of our model. Theoretically, these hardness results may not inspire much optimism as to the solution quality a PTIME algorithm can achieve. Nevertheless, we devised two heuristic algorithms, both used by ConCaT. When testing their effectiveness over large real-world datasets of e-commerce search queries, the achieved scores far surpassed the worst-case theoretical bounds.

The first algorithm is the Category Tree Conflict Resolver ($CTCR$) algorithm, based on the following approach: we identify pairs and triplets of queries, referred to as *conflicts*, such that, for each conflict, it is mathematically impossible for any tree to cover all its queries simultaneously. We then leverage algorithms for the Maximum Independent Set (MIS) problem, to compute a conflict-free subset of the queries, which we aim to cover entirely. Although MIS is inapproximable, there are practical algorithms that solve it efficiently. The tree structure is then derived, based on the containment relations of the queries in the conflict-free set. Finally, we greedily assign items to categories and condense the tree by removing redundant items and categories.

The second algorithm is the Clustering-based Category Tree (CCT) algorithm, based on the following approach: we identify a suitable vector representation (embedding) for each query, and employ an agglomerative clustering algorithm over the query embeddings, to derive the optimal tree structure, where each category is marked with the queries it should aim to cover. Then, we use the same procedure as in $CTCR$, to greedily assign items and condense the tree.

Both algorithms apply schematically to any reasonable similarity function. Moreover, ConCaT supports a more general model where each query may have its own threshold that a covering category must exceed. Lastly, we note that ConCaT can mix into the input an existing tree, such that the produced tree will not be radically different, maintaining consistency.

III. System Architecture

We implemented ConCaT using Python and Flask. The system architecture is depicted in Figure 3. First, ConCaT receives as input the search queries, along with the desired similarity function and its threshold parameter. Next, the Data Adaptor computes the result sets, which it then cleans and preprocesses into a collection of weighted item sets. Then, over this input, the two algorithms, $CTCR$ and CCT, are employed (recall that the greedy item assignment procedure is shared by both algorithms and is, thus, depicted as a separate module). The tree with the superior score out of the two produced trees is then selected. Over this tree, the Category Labeler automatically derives candidates for meaningful labels for each category, based on its item set and the queries it covers. Lastly, the taxonomist examines the final tree and may adjust the input or impose soft constraints, to prompt the system to recompute the solution.

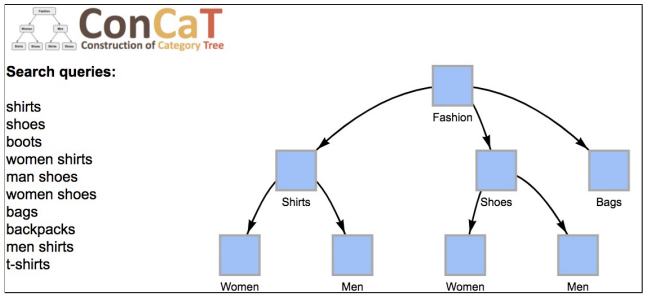

Fig. 4. Suggested tree based on the given search queries

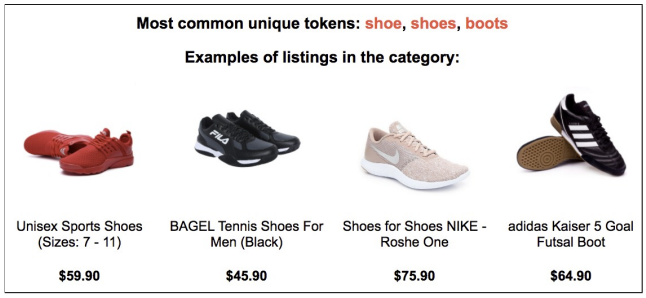

Fig. 5. Sample listings assigned to the "Shoes" category

IV. Demonstration Scenario

We demonstrate over real-world e-commerce data the operation of ConCaT, an interactive system that constructs category trees from a workload of user queries. We reenact a scenario where an e-commerce platform redesigns its online store, and wishes to construct a category tree, that most closely matches the item sets sought by users. The audience will play two separate roles: (1) consumers who submit queries to the platform's search-engine, and (2) taxonomists that utilize ConCaT to construct trees that best match these queries. Our system allows taxonomists to impose various soft constraints, prompting ConCaT to efficiently recompute the solution.

First, we will present a UI that enables the audience to type in various search queries, and view the result sets. Then, ConCaT will produce a category tree based on these sets, as depicted in Figure 4. Finally, the taxonomist may reemploy and tweak the solution, by adjusting the configuration.

Input: To start the demonstration the audience submits multiple search queries over fashion and electronics item repositories, taken from large real-world e-commerce platforms. Once all search queries are submitted, the UI presents a list of pre-searched queries, from which the users may select additional queries to add to the input. The audience may indicate next to each query its weight, to indicate how frequently it is assumed to be submitted. Next, ConCaT employs the Data Adaptor, and presents, in a new window, the final input. We will also explain the operation of the algorithm used to derive it.

Solution Generation: After exploring the input data, the taxonomists will run our algorithms, which produce in real-time the category tree, along with label candidates for each category, and accompanying statistics, indicating which categories cover which queries (and the relevant similarity scores, along with the overall score). Together with the audience we will use the UI to examine the proposed solution and the corresponding statistics. For example, taxonomists may select on one of the categories, which prompts the UI to display its cover statistics. Double clicking the category opens a new window, which includes various views of the item sets, including the view depicted in Figure 5, showing the most common tokens in the item set, and a small representative subset of the items (the users may also scroll through the complete list of items).

Interactive adjustments: Upon examining the solution, if some categorization decisions are unsatisfactory, taxonomists may adjust the input accordingly. This prompts ConCaT to recompute on-the-fly the relevant parts of the solution, presenting the new tree with the most recent changes highlighted. Moreover, the taxonomists may add to the input an existing tree (which is the actual tree used by the e-commerce platform relevant to the given dataset), and indicates on a slider the extent to which this categorization should be preserved. The Data Adaptor then adds the existing categories as additional input sets weighting according to the indication on the slider.

Related Work The effective construction of category trees/taxonomies has been the focus of research in a variety of domains, including e-commerce, document management, and question answering [4]–[7]. Aiming at assisting domain-expert taxonomists in this difficult task, multiple algorithms for automating the category tree construction [4], [8] and maintenance [5], [9], [10] have been proposed, employing a variety of clustering approaches, as well as crowdsourcing [11]. More broadly, the e-commerce setting has lately received much attention from researchers on a variety of problems [12], [13]. To our knowledge, we are the first to provide a tree construction algorithm from query logs in any domain.

Acknowledgments: This work has been partially funded by the Israel Science Foundation, the Binational US-Israel Science Foundation, Tel Aviv University Data Science center, and eBay Israel.

References

[1] Y. Zhang, A. Ahmed, V. Josifovski, and A. Smola, "Taxonomy discovery for personalized recommendation," in *WSDM*, 2014, pp. 243–252.
[2] D. Jiang, J. Pei, and H. Li, "Mining search and browse logs for web search: A survey," *TIST*, vol. 4, no. 4, pp. 1–37, 2013.
[3] "ConCaT (hardness)," https://slavanov.com/research/concat_proofs.pdf.
[4] S.-L. Chuang and L.-F. Chien, "A practical web-based approach to generating topic hierarchy for text segments," in *CIKM*, 2004.
[5] Q. Yuan, G. Cong, A. Sun, C.-Y. Lin, and N. M. Thalmann, "Category hierarchy maintenance: a data-driven approach," in *SIGIR*, 2012.
[6] C. Zhang, F. Tao, X. Chen, J. Shen, M. Jiang, B. Sadler, M. Vanni, and J. Han, "Taxogen: Unsupervised topic taxonomy construction by adaptive term embedding and clustering," in *Proc. of KDD*, 2018.
[7] J. Shang, X. Zhang, L. Liu, S. Li, and J. Han, "Nettaxo: Automated topic taxonomy construction from text-rich network," in *Proc. of WWW*, 2020.
[8] K. Punera, S. Rajan, and J. Ghosh, "Automatically learning document taxonomies for hierarchical classification," in *Proc. of WWW*, 2005.
[9] H. Zhuge and L. He, "Automatic maintenance of category hierarchy," *Future Generation Computer Systems*, vol. 67, pp. 1 – 12, 2017.
[10] I. Hasson, S. Novgorodov, G. Fuchs, and Y. Acriche, "Category recognition in e-commerce using sequence-to-sequence hierarchical classification," in *WSDM*, 2021.
[11] Y. Sun, A. Singla, D. Fox, and A. Krause, "Building hierarchies of concepts via crowdsourcing," 2015.
[12] S. Gershtein, T. Milo, G. Morami, and S. Novgorodov, "Minimization of classifier construction cost for search queries," in *SIGMOD*, 2020.
[13] S. Gershtein, T. Milo, and S. Novgorodov, "Inventory reduction via maximal coverage in e-commerce," in *EDBT*, 2020.