A Hierarchical Dirichlet Model for Taxonomy Expansion for Search Engines

Jingjing Wang* Changsung Kang † Yi Chang † Jiawei Han *

*University of Illinois at Urbana - Champaign
Urbana, IL 61801

†Yahoo! Labs
701 First Avenue, Sunnyvale, CA 94089

*{jwang112, hanj}@illinois.edu †{ckang, yichang}@yahoo-inc.com

ABSTRACT

Emerging trends and products pose a challenge to modern search engines since they must adapt to the constantly changing needs and interests of users. For example, vertical search engines, such as Amazon, eBay, Walmart, Yelp and Yahoo! Local, provide business category hierarchies for people to navigate through millions of business listings. The category information also provides important ranking features that can be used to improve search experience. However, category hierarchies are often manually crafted by some human experts and they are far from complete. Manually constructed category hierarchies cannot handle the ever-changing and sometimes long-tail user information needs. In this paper, we study the problem of how to expand an existing category hierarchy for a search/navigation system to accommodate the information needs of users more comprehensively. We propose a general framework for this task, which has three steps: 1) detecting meaningful missing categories; 2) modeling the category hierarchy using a hierarchical Dirichlet model and predicting the optimal tree structure according to the model; 3) reorganizing the corpus using the complete category structure, *i.e.*, associating each webpage with the relevant categories from the complete category hierarchy. Experimental results demonstrate that our proposed framework generates a high-quality category hierarchy and significantly boosts the retrieval performance.

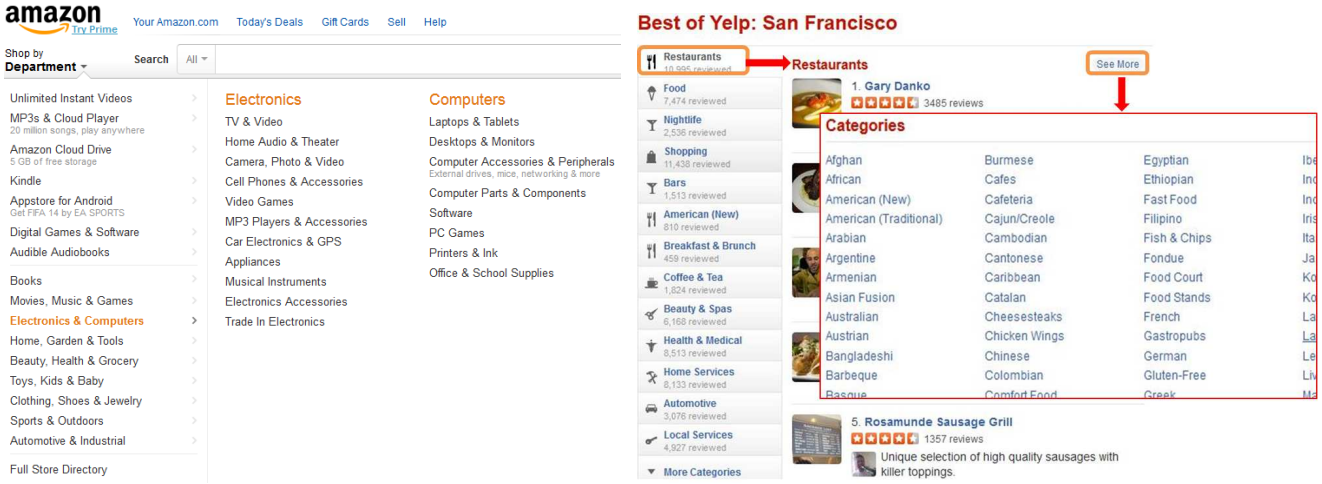
1. INTRODUCTION

Taxonomies have been fundamental to organizing knowledge and information for centuries[24]. Nowadays with the vast development of web technology, almost all the modern websites with search/navigation features have adopted taxonomies to improve user experience. Online retailers such as `Amazon`[1] and `Zappos`[2] classify their goods under different departments. Consumers can navigate through the category hierarchy to locate the items that they want to buy. A consumer can also type in the search box a category query like "offce chairs" and get a list of ranked results about office chairs. Local search providers such as `Yelp`[3] and `Yahoo! Local`[4] also provide a business category hierarchy to facilitate navigation through business listings. In addition, direct business/category search is supported as well. Fig. 1 shows a snapshot of the taxonomy for Amazon and Yelp.

Taxonomies play two essential roles in online search engines. The first one is straightforward: page navigation. A webpage is associated with its relevant categories. Therefore, under each category in the taxonomy are the related webpages linked to it. Once a user navigates to a particular category, he or she can browse those pages and delve into the ones of interest. The other one is not as explicit: taxonomies provide useful features for ranking in the retrieval process. To illustrate, let's assume a simple tf-idf weighting scheme for the ranking function. Suppose we add the relevant categories of each document to the content of the document, for example, we may add "fast food" to the content of an In-N-Out Burger business listing. Even if the original business page does not contain the term "fast food", there will be an exact bi-gram match when a category query "fast food" is issued because of the added category information. In commercial search engines, more sophisticated ranking schemes are used and both local and structural features extracted from the category hierarchy are utilized to facilitate information retrieval.

Unfortunately, constructing a complete taxonomy (or category hierarchy) for a search engine is extremely difficult. A taxonomy is often manually constructed by human experts. Not only is this step very expensive, but it is also impossible to get a comprehensive taxonomy due to the sheer amount of information. Human experts may miss emerging categories and long-tail categories. Also, the category names selected by human experts may not be consistent with actual queries used by users, which may affect the search quality for the search engine. To illustrate how a missing category can affect search quality, consider a category Water Park, which is currently missing in Yelp's taxonomy. The search ranking results using Water Park as query (with Sunnyvale, CA as location) are shown in Fig. 2. Obviously, only the second result (*California Splash Water Park*) is a water park which is 33 miles away from Sunnyvale. The third result is a dog park while the others including the advertisement are all swimming pools that happen to contain the keywords

[1]http://www.amazon.com/
[2]http://www.zappos.com/
[3]http://www.yelp.com/
[4]http://local.search.yahoo.com/



(a) A Snapshot of Amazon Taxonomy

(b) A Snapshot of Yelp Taxonomy

Figure 1: Taxonomies in Search Engines

water and park. In fact, there is a popular water park *Raging Waters* located right in San Jose, CA which is only 17 miles away that is not shown even in the top 30 results. The main cause of this problem is that the relevant water parks "cannot" be categorized as water parks since the category is completely missing in the taxonomy (*Raging Waters* is currently categorized as Amusement Parks).

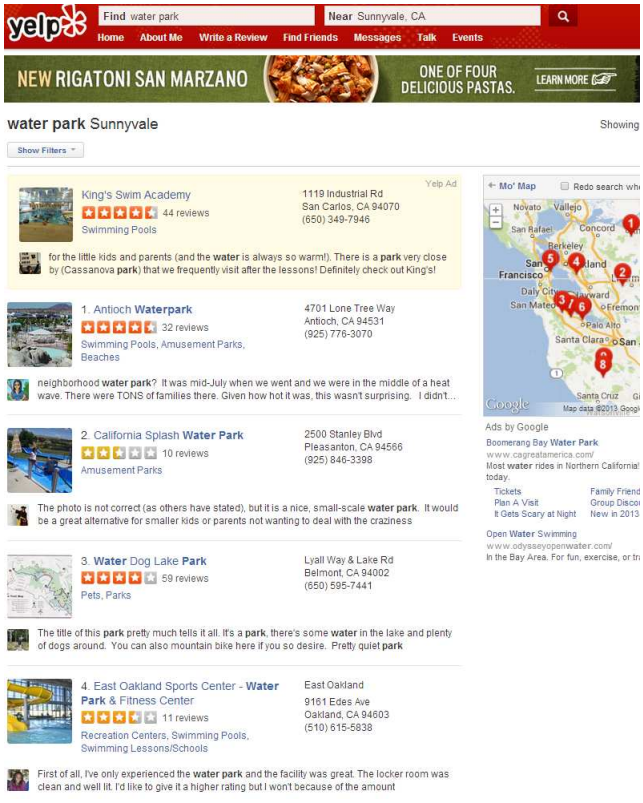


Figure 2: Water Park *near* Sunnyvale, CA

In this paper, we study the problem of how to expand an existing category hierarchy[5] inherent in a search/navigation system to accommodate the information need of users. We propose a general framework for this task including three steps: 1) detecting meaningful new categories from user queries; 2) modeling the category hierarchy using a hierarchical Dirichlet model and predicting the optimal tree structure according to the model; 3) reorganizing the corpus using the complete category hierarchy, *i.e.*, associating each document with the relevant categories from the complete hierarchy. Our major contributions are outlined as follows.

- We introduce a unified framework to expand an existing category hierarchy which can be applied to any search/navigation system.
- We propose a novel hierarchical Dirichlet model to capture the structural dependency inherent in a taxonomy and formulate a structure learning problem which can be efficiently solved by the maximum spanning tree algorithm.
- Comprehensive experiments are conducted on a large-scale commercial local search engine. The results demonstrate the effectiveness of our framework.

The rest of the paper is organized as follows. Section 2 formally defines the problem. We introduce our framework for taxonomy expansion in Section 3 and discuss related work in Section 4. Section 5 analyzes the properties of our framework and discusses some practical issues. We report out experimental results in Section 6 and conclude our study in Section 7.

2. PROBLEM FORMULATION

In this section, we formally define the problem of taxonomy expansion for search engines. The notations used in this paper are listed in Table 1.

[5]In this paper, we use "taxonomy", "category hierarchy", "category tree" interchangeably; and "missing category", "new category" interchangeably

Table 1: Notations Used in this Paper

Symbol	Description
$\mathbf{C}$	category set
root	the pseudo root node in the taxonomy
$\mathbf{V}$	vocabulary
$\mathcal{D}$	online corpus
$\mathbf{H}$	taxonomy of an online corpus
d	item page
$\mathbf{t}$	bag-of-words representation of an item page
$\mathbf{c}$	relevant categories for an item page d
c	a category
q	a query
$\mathbf{R}_q$	clicked collection for query q
$\boldsymbol{\phi}_c$	the multinomial representation of category c

DEFINITION 1 (CATEGORY SET). *A Category set* $\mathbf{C}$ *is the set of categories in the online search/navigation system.*

An existing category set $\mathbf{C}^u$ contains the current set of categories which are actively used where some categories might be missing. $\mathbf{C}^m$ contains the set of categories that are currently missing and unknown. A complete category set $\mathbf{C}^c = \mathbf{C}^u \bigcup \mathbf{C}^m$ denotes the complete set of categories which we want to recover.

In our problem setting, $\mathbf{C}^u$ is given by human experts while $\mathbf{C}^c$ is the one we should identify (Section 3.1).

DEFINITION 2 (ITEM PAGE). *An item page* $d = (\mathbf{t}, \mathbf{c})$ *is a webpage which contains a bag-of-words description* $\mathbf{t}$ *of a product or a business, etc. and a set of relevant categories* $\mathbf{c}$ *to this page.*

With different category sets, the representation of an item page has two versions:
$d^u = (\mathbf{t}, \mathbf{c}^u)$, where $\mathbf{c}^u$ is the set of relevant categories tagged to the item page by either business owners or content providers.
$d^c = (\mathbf{t}, \mathbf{c}^c)$, where $\mathbf{c}^c$ is the set of categories we will tag to the item page with the complete taxonomy that will be constructed. Specifically, $d^c.\mathbf{c}^c$ is initially unknown. We will augment $d^u.\mathbf{c}^u$ to $d^c.\mathbf{c}^c$ (accordingly, d^u to d^c) by our model (Section 3.3).

DEFINITION 3 (ONLINE CORPUS). *An online corpus* $\mathcal{D} = (\mathbf{D}, \mathbf{C}, \mathbf{H})$ *contains a set of indexed item pages* $\mathbf{D} = \{d_1, d_2, ...\}$, *a category set* $\mathbf{C}$ *associated with the corpus, and a category hierarchy* $\mathbf{H} = \{\langle c, parent(c)\rangle | c \in \mathbf{C} \backslash root^6\}$. *The hierarchy* $\mathbf{H}$ *consists of a set of child-parent relations.*

Similarly as before, we have $\mathcal{D}^u = (\mathbf{D}^u, \mathbf{C}^u, \mathbf{H}^u)$ and $\mathcal{D}^c = (\mathbf{D}^c, \mathbf{C}^c, \mathbf{H}^c)$ defined on the existing hierarchy and the to-be-constructed complete hierarchy, respectively. They key aspect in our framework lies in how to expand $\mathbf{H}^u$ to $\mathbf{H}^c$ (Section 3.2).

DEFINITION 4 (CLICKED COLLECTION). *Given a query* q, *the item pages that have been clicked form a clicked collection* $\mathbf{R}_q = \{d | d$ *is clicked for the query* $q, d \in \mathbf{D}\}$ *for this query.*

[6] we add a pseudo root node to the category set for a neat tree notation.

Note that the clicked collection is defined in an aggregate manner. A query q could be issued multiple times to a search engine. As long as a page has ever been clicked for q, it is added to $\mathbf{R}_q$.

We are now able to formulate our taxonomy expansion problem as follows.

PROBLEM 1 (TAXONOMY EXPANSION). *Given an online corpus* $\mathcal{D}^u$ *with the existing taxonomy, expand the category set* $\mathbf{C}^u$ *to a complete set* $\mathbf{C}^c$, *the hierarchy* $\mathbf{H}^u$ *to a complete hierarchy* $\mathbf{H}^c$, *and augment each item page* $d^u \in \mathbf{D}^u$ *to* d^c *which forms* $\mathbf{D}^c$, *to obtain the updated corpus* $\mathcal{D}^c = (\mathbf{D}^c, \mathbf{C}^c, \mathbf{H}^c)$.

3. A GENERAL FRAMEWORK FOR TAXONOMY EXPANSION

Our objective is to construct a complete taxonomy for an online corpus and associate each document in the corpus with the relevant categories from this complete taxonomy. As indicated in our problem definition, the taxonomy expansion problem can be divided into three sub-problems: missing category discovery; hierarchy reconstruction; and item page re-tagging. Fig.3 shows the overall framework.

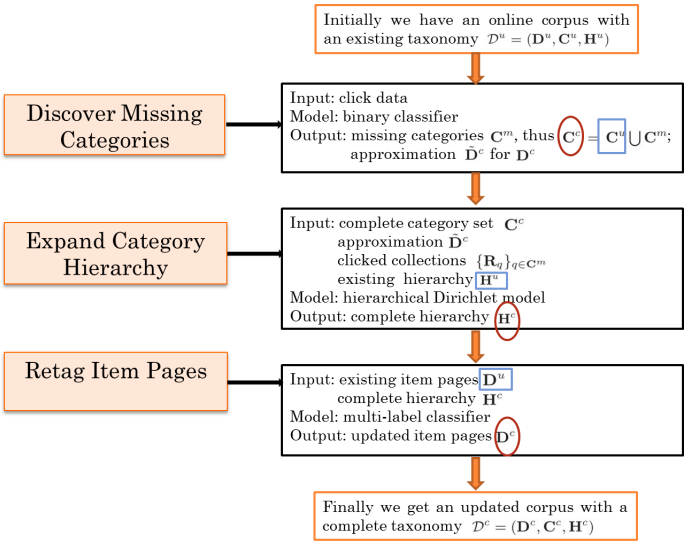


Figure 3: An Overview of the Taxonomy Expansion Framework

3.1 A Classifier for Missing Category Discovery

Discovering categories that are missing from the current corpus is the first step of our taxonomy expansion framework. The goal is to identify a set $\mathbf{C}^m$ of missing categories to have $\mathbf{C}^c = \mathbf{C}^u \cup \mathbf{C}^m$. Since our taxonomy is for a search engine, categories in the taxonomy should be aligned with what users are searching for. Thus, we let $\mathbf{C}^m$ be a subset of user queries $\mathbf{Q}$ for the search engine. The problem can be cast as a binary text classification problem where we classify user queries into two classes: unique names and category

names. For example, in the local search domain, a famous restaurant name such as "The French Laundry" is classified as a unique name while a type of cuisine such as "Chinese Restaurants" is classified as a category name. After we build a classifier g, we have $\mathbf{C}^m = \{q \mid g(q) = 1, q \in \mathbf{Q})\} \setminus \mathbf{C}^u$.

Obtaining an enough amount of labeled data to train a high-quality classifier is very costly. Thus, we propose a semi-supervised learning method which uses the combination of labeled data and search click log data. We leverage user click data to augment labeled training data. A key observation is that

users tend to click more documents for category names (as query) than unique names per search session.

For example, given a category name query "Chinese Restaurants", the search results page often shows many relevant Chinese restaurants. Hence, users explore the search results page by clicking some of the results until their information needs are satisfied. On the other hand, given a unique name query, there are only a few perfectly relevant results (such as the official Website for the entity in the query) in the page and users end up clicking only those few links. Based on this observation, we create pseudo-labeled data where a label is assigned to a query based on the *average clicks (AC)* per query session: a category name if the AC of the query is larger than a threshold α, a unique name if the AC of the query is less than another threshold β.

Our training data is $\mathbf{T} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_M, y_M)\}$ where $\mathbf{x}_i$ is a feature vector (including unigrams, bigrams and the average click counts) of a query i and y_i is a label (1 for a category name and 0 for a unique name). $\mathbf{T}$ is the union of labeled data $\mathbf{T}_{label}$ and pseudo-labeled data $\mathbf{T}_{pseudo}$ where y_i in $\mathbf{T}_{label}$ is provided by human experts and y_i in $\mathbf{T}_{pseudo}$ is decided by the average clicks per query session (1 if AC$>\alpha$, 0 if AC$<\beta$).

Many machine learning algorithms can be applied to our training data $\mathbf{T}$ to generate a classifier. We use Linear Support Vector Machine (SVM) due to its high accuracy and speed.

With the discovered missing categories, we construct $\tilde{\mathbf{D}}^c$ to approximate $\mathbf{D}^c$. $\tilde{\mathbf{D}}^c$ is generated as follows. Recall that the missing categories come from user queries. For each missing category c, we concatenate it to the category set $d^u.\mathbf{c}^u$ for all $d^u \in \mathbf{R}_c$ (the clicked collection for q). Therefore each d^u is augmented with the queries (categories) it is clicked for. And the augmented d^u's form an approximation $\tilde{\mathbf{D}}^c$ for $\mathbf{D}^c$. For example, consider a page $d^u = (\mathbf{t}, \{\text{Amusement Parks}\}) \in \mathbf{D}^u$ for *Raging Waters*. Suppose that d^u has been clicked for queries Water Parks and Waterslides and both these two queries are in $\mathbf{C}^m$. Then, an item page $\tilde{d^c}$ in $\tilde{\mathbf{D}}^c$ corresponding to d^u is
$(\mathbf{t}, \{\text{Amusement Parks, Water Parks, Waterslides}\})$.

3.2 A Hierarchical Dirichlet Model for Taxonomy Expansion

The goal of this step is to arrange the updated category set to a new hierarchy while preserving the existing category structure. So far we have obtained the complete category set $\mathbf{C}^c$, a noisy item page collection $\tilde{\mathbf{D}}^c$ tagged with the complete category set, and the existing category hierarchy $\mathbf{H}^u$. We construct the hierarchy from $\tilde{\mathbf{D}}^c$ and $\mathbf{C}^c$, with $\mathbf{H}^u$ as the constraint.

We propose a hierarchical Dirichlet model to capture the generating process of a taxonomy based on the content of the item pages and search click log data from users. We formulate the problem of finding the optimal tree as a structure learning problem elegantly solved by the Maximum Spanning Tree (MST) algorithm for directed graphs. We preserve the existing hierarchy by restricting the set of parent candidates for each category. This not only fully respects the experts' knowledge but also effectively prunes the search space and greatly reduces the prediction cost.

3.2.1 Modeling the Taxonomy

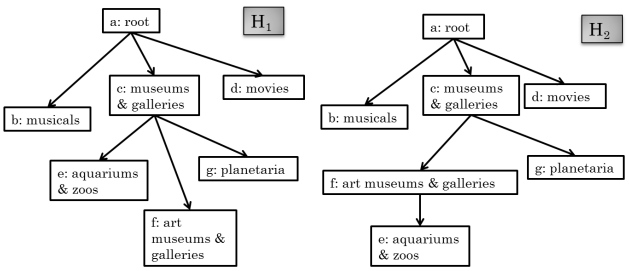
In a category tree, each node represents a category. We consider each node as a random variable and the category tree as a Bayesian network. Then the joint probability distribution of the nodes $\mathbf{N}$ depending on a particular tree structure $\mathbf{H}$ with model parameters Θ (note that we have not yet specified our model so we use Θ to represent all the model parameters for now) is

$$\mathbf{P}(\mathbf{N}|\mathbf{H}, \Theta) = \mathbf{P}(root) \prod_{n \in \mathbf{N} \setminus root} \mathbf{P}(n|parent_{\mathbf{H}}(n), \Theta)$$

where $parent_{\mathbf{H}}(n)$ is the parent node of n in $\mathbf{H}$. This is actually the likelihood of the tree structure $\mathbf{H}$ given model parameters Θ. Maximizing the likelihood with respect to the tree structure $\mathbf{H}$ gives the optimal tree:

$$\mathbf{H}^* = \arg\max_{\mathbf{H}} \mathbf{P}(\mathbf{N}|\mathbf{H}, \Theta)$$

We illustrate the structure learning problem in Example 1.



(a) Tree 1 (b) Tree 2

Figure 4: Two Examples of Possible Tree Structures

EXAMPLE 1 (LEARNING THE OPTIMAL STRUCTURE). *As shown in Fig. 4, suppose we have 7 category nodes $\{a, b, ..., f\}$. The likelihood of the two possible tree structures are*

$$\mathbf{P}(a, ..., f|\mathbf{H}_1, \Theta) = \mathbf{P}(a)\mathbf{P}(b|a)\mathbf{P}(c|a)\mathbf{P}(d|a)\mathbf{P}(e|c)\mathbf{P}(f|c)\mathbf{P}(g|c)$$

and

$$\mathbf{P}(a, ..., f|\mathbf{H}_2, \Theta) = \mathbf{P}(a)\mathbf{P}(b|a)\mathbf{P}(c|a)\mathbf{P}(d|a)\mathbf{P}(f|c)\mathbf{P}(g|c)\mathbf{P}(e|f)$$

respectively[7]*. The $\mathbf{H}$ in $\{\mathbf{H}_1, \mathbf{H}_2, ...\}$ which gives the maximum likelihood will be output as the optimal $\mathbf{H}^*$.*

[7]All the conditional probabilities should contain Θ as the condition as well, *i.e.*., each term should be $\mathbf{P}(n|parent_{\mathbf{H}}(n), \Theta)$ rather than $\mathbf{P}(n|parent_{\mathbf{H}}(n))$. Here we omit it for brevity. In the rest of paper we omit it in the same manner as long as there is no ambiguity

3.2.2 Category Representation

The problem now reduces to how to represent each category and the conditional probability of a category node given its parent. We consider the following two factors when we design our model.

First, recall the definition of an item page in Section 2. Each item page has been tagged with its relevant categories by either business owners or content providers. This valuable information should be carefully utilized as supervision. Second, a taxonomy in a search engine provides an organization of categories which should correctly represents human cognition. Therefore our taxonomy still adopts the widely used *is-a* relationship[4] although we do not impose this relationship strictly for every pair of child and parent. This requires our model to enforce certain similarity between a child and a parent. With these intuitions in mind, we use the multinomial distribution to model each category and the Dirichlet distribution to model the conditional probability of a child category given its parent.

With an online corpus $\mathcal{D}$, we denote the collection of item pages belonging to a category c by $\mathbf{D}_c$. $\mathbf{D}_c = \{d|c \in d.\mathbf{c}, d \in \mathcal{D}.\mathbf{D}\}$. We fit a uni-gram language model to $\mathbf{D}_c$ to get the random variable (distribution) $\boldsymbol{\phi}_c = \{\phi_{c,t}\}_{t\in\mathbf{V}}$ *s.t.* $\sum_{t\in\mathbf{V}} \phi_{c,t} = 1$ for c, where $\mathbf{V}$ is the vocabulary for our corpus. $\boldsymbol{\phi}_c$ is then used to represent the category c.

To capture the *is-a* relationship between c and $parent(c)$, we would like to prefer a model where the expectation of the distribution for the child is exactly the distribution for the parent, *i.e.*, $\mathbb{E}(\phi_c) = \phi_{parent(c)}$. This naturally leads to the Dirichlet distribution[12]. The Dirichlet distribution is a distribution over distributions. Specifically, it is defined over a simplex where each point in the simplex represents a multinomial distribution. We define that the conditional probability of a category node c given its parent $parent(c)$ comes from a Dirichlet distribution:

$$\boldsymbol{\phi}_c|\boldsymbol{\phi}_{parent(c)} \sim Dir(\boldsymbol{\phi}_{parent(c)}; \alpha)$$

where α is the concentration paramenter for the Dirichlet distribution, which determines how "concentrated" the probability mass of a sample is likely to be. We set α to the same value for every node since we do not have extra knowledge. So our model only has one parameter α, *i.e.*, $\Theta = \alpha$. Thus we have $\mathbf{P}(\boldsymbol{\phi}_c|\boldsymbol{\phi}_{parent(c)}, \alpha) = \frac{1}{Z}\prod_{t\in\mathbf{V}} \phi_{c,t}^{\alpha\phi_{parent(c),t}-1}$ where $Z = \frac{\prod_{t\in\mathbf{V}}\Gamma(\alpha\phi_{parent(c),t})}{\Gamma(\sum_{t\in\mathbf{V}})\alpha\phi_{parent(c),t}}$ is a normalization factor and $\Gamma(\cdot)$ is the standard Gamma distribution[8].

By definition, the Dirichlet distribution ensures that $\mathbb{E}(\boldsymbol{\phi}_c) = \boldsymbol{\phi}_{parent(c)}$.

3.2.3 Optimizing the Likelihood

In this section, we optimize the likelihood $\mathbf{P}(\mathbf{N}|\mathbf{H}, \Theta)$ to get the optimal tree structure under our model assumption.

$$\begin{aligned}\mathbf{H}^* &= \arg\max_{\mathbf{H}} \mathbf{P}(\mathbf{N}|\mathbf{H}, \Theta) \\ &= \arg\max_{\mathbf{H}} \prod_{n\in\mathbf{N}} \mathbf{P}(n|parent_{\mathbf{H}}(n), \Theta) \\ &= \arg\max_{\mathbf{H}} \sum_{n\in\mathbf{N}} \log \mathbf{P}(n|parent_{\mathbf{H}}(n), \Theta)\end{aligned}$$

If we consider the category nodes as the vertices in a graph, and assign a weight $\log \mathbf{P}(n_1|n_2)$ to every edge $\langle n_1, n_2\rangle$ with $n_1 \in \mathbf{N} \setminus root$ and $n_2 \in \mathbf{N}$, the optimization problem becomes finding the maximum spanning tree in the directed complete graph with nodes being all the categories. We apply the Chu-Liu/Edmonds' algorithm[6] to solve the problem. Basically it is a greedy algorithm with two steps: selecting best entering edges and breaking cycles. We maintain a set M of maximum-weight entering edges. Initially M is empty. The algorithm selects an arbitrary node which does not yet have an entering edge in M, finds the maximum-weight entering edge for this node and adds it to M. Do this until M contains a cycle. Then we contract the cycle to a pseudo-node and proceed the same way to add maximum-weight edges to M. Once there is no node left, we break the cycles (pseudo-nodes) by removing the minimum-weight edge in each cycle.

3.2.4 Candidate Pruning

So far we have completed our discussion on how to construct the taxonomy for an online corpus $\mathcal{D}$, given its category set $\mathbf{C}$ and the collection of its indexed item pages $\mathbf{D}$. Let $\mathbf{D} = \tilde{\mathbf{D}}^c$, $\mathbf{C} = \mathbf{C}^c$, and run the model, we will get the complete hierarchy $\mathbf{H}^c$. Instead of reporting this hierarchy as our final result, we introduce a pruning step before the tree structure search.

Pruning is very important here for two reasons. First, we want to preserve the existing structure exactly as it is. Second, for a complete graph, even the most efficient implementation of Edmond's algorithm runs in $O(n^2)$ time[8] where n is the number of vertices. It is desired to involve a pruning process to reduce the search space.

To restrict the tree to be an expansion of the existing tree, for each existing category node c, we only keep the entering edge $(c, parent_{\mathbf{H}^u}(c))$. For a new category node, the candidate set of its parents can also be pruned. Since the missing categories come from user queries, we only consider the relevant categories of the clicked pages as possible parents. Formally, the tree structure $\mathbf{H}^c$ is optimized under the following two constraints:

$$\begin{cases} parent_{\mathbf{H}^c}(c) = parent_{\mathbf{H}^u}(c), \text{if } c \in C^u \\ parent_{\mathbf{H}^c}(c) \in \{c'|c' \in \bigcup_{d\in\mathbf{R}_c} d.\mathbf{c}\}, \text{if } c \in C^m \end{cases}$$

where $\mathbf{R}_c$ is the set of pages clicked for query c.

3.3 Item Page Re-tagging

The last step of our taxonomy expansion framework is to augment the category set $\mathbf{c}^u$ for each item page $d^u = (\mathbf{t}, \mathbf{c}^u)$ with relevant new categories to obtain $\mathbf{c}^c$. We first identify a set $\mathbf{c}^r$ of potential new categories for d^u. Let $d^r = (\mathbf{t}, \mathbf{c}^r)$.

[8]Note that the above distribution does not apply to categories which have *root* as parent. We assign a uniform Dirichlet distribution to $\mathbf{P}(\boldsymbol{\phi}_c|root)$.

Then we apply a multi-label classifier h to obtain the relevant new category set $\mathbf{c}^m = h(d^r) \subset \mathbf{c}^r$. Then we have $\mathbf{c}^c = \mathbf{c}^u \cup \mathbf{c}^m$.

3.3.1 Identifying Potential New Categories

We may use the entire set of new categories $\mathbf{C}^m$ as the potential categories for every page d and skip this step. However, this naive approach is inefficient when $\mathbf{C}^m$ is large. Hence, we propose to generate a compact set of potential new categories by leveraging the category hierarchy $\mathbf{H}^c$ from the previous step.

Given $\mathbf{H}^c$, the set of potential new categories for $d^u = (\mathbf{t}, \mathbf{c}^u)$ is

$$\mathbf{c}^r = \mathbf{C}^m \cap \left\{ \cup_{n \in \mathbf{c}^u} \left(descendants_{\mathbf{H}^c}(n) \cup siblings_{\mathbf{H}^c}(n) \right) \right\}$$

where $descendants_{\mathbf{H}}(n)$ is the set of all descendants of a node n in the tree structure $\mathbf{H}$ and $siblings_{\mathbf{H}}(n)$ is the set of all siblings of a node n in the tree structure $\mathbf{H}$. In other words, we consider all new categories that are either a descendant or a sibling of any category in the given category set $\mathbf{c}^u$ as potential new categories.

3.3.2 Predicting Relevant Categories with A Multi-label Classifier

Given a set of potential new categories $\mathbf{c}^r$ for d^u, we use the multi-label classification method proposed in [11] to obtain relevant new categories. The method is based on a set of highly predictive features including centroid-based similarity features (similarity between the term distribution of a category and the term distribution of a page), click features and features derived from relationships among categories. The method is applied to $d^r = (\mathbf{t}, \mathbf{c}^r)$ to generate a set of relevant categories $\mathbf{c}^m = h(d^r)$, which is a subset of $\mathbf{c}^\mathbf{r}$. Finally, the category set $\mathbf{c}^u$ of d is augmented with $\mathbf{c}^m$.

To illustrate the above process, consider a page $d^u = (\mathbf{t}, \{\text{Museums \& Galleries}, \text{Tourist Attractions}\})$. Suppose that this page is about the Science Museums category and the category Science Museums is missing in $\mathbf{H}^u$ and is now included in $\mathbf{H}^c$. Also, suppose that Science Museums and History Museums are the only new categories under Museums & Galleries and there are no sibling new categories of Museums & Galleries and Tourist Attractions. Then, $\mathbf{c}^r = \{\text{Science Museum}, \text{History Museums}\}$. Suppose that the multi-label classifier h correctly produces science museums as the output:

$$h\Big((\mathbf{t}, \{\text{Science Museum}, \text{History Museums}\})\Big) = \{\text{Science Museum}\}.$$

Then, we have an updated item page $d = (\mathbf{t}, \{\text{Museums \& Galleries}, \text{Tourist Attractions}, \text{Science Museums}\})$.

4. RELATED WORK

To the best of our knowledge, our overall problem setting is novel and there is no previous work addressing the task of taxonomy expansion for a hierarchically organized corpus in a search/navigation system. Yet our work is closely related to taxonomy induction. Taxonomy induction from text has been an active research field for long. We review the recent literature which are mostly relevant to our work. They are roughly categorized into the following three classes. There is no hard boundary between these methods and sometimes there could be hybrid approaches.

Hierarchical Topic Modeling There has been a substantial amount of research on adapting topic models to learn concept hierarchies from text. Studies along this line such as hLDA [9], hPAM[17], nonparametric hLDA[2], nonparametric PAM[14], hHDP[26], SSHLDA[16] hLLDA[20] generally fit a generative topic model for a text corpus with the bag-of-words assumption. They consider every single word as an observation and infer hierarchically related topics under the model assumption. Each topic is represented by a multinomial word distribution and forms a node in the concept hierarchy. These models are mostly unsupervised and the semantic meaning of each topic must be annotated with human interaction. In addition, the topic models mentioned above involve expensive inference which does not scale up well for large corpus. In contrast, our task requires each node in the hierarchy to capture exactly a pre-defined category and the problem scale is in the order of millions of documents.

Hierarchical Clustering Hierarchical clustering represents a group of data mining methods for data analysis. It is also well explored to do taxonomy induction, when each cluster is considered as a concept node. The general idea is that words occurring in similar contexts are more likely to be grouped to the same cluster. Hierarchical clustering for taxonomy induction takes either a divisive or agglomerative manner. Wang *et al.* [25] proposed a term co-occurrence network to group terms that highly co-occur with each other to the same topic divisively. Liu *et al.* [15] adopted the Bayesian Rose Tree[3] to build a concept hierarchy for a given set of keywords agglomeratively. Similar as topic modeling, the specific semantic meaning of each cluster cannot be controlled. Additionally, hierarchical clustering requires the number of clusters for each level *a priori*, which makes it inapplicable in our task since we cannot assume we know the number of nodes in each level before we know the structure.

Linguistics Based Methods Linguistics based methods are widely used to induce a lexical taxonomy from text. Syntactic structure of a sentence is carefully analyzed by applying NLP techniques such as part-of-speech tagging[23], dependency parsing[19], association rule mining[1], seed pattern (IS-A, PART-OF, SIBLING-OF) extraction etc. to extract domain terms and the relations, based on which a taxonomy is induced [13, 21, 18]. In the taxonomy construction process, a classification model is often involved to predict the parent for each term. The nature of our task distinguishes it from the lexical taxonomy induction for 1) rather than a keyword/phrase, each node in our hierarchy is a category which has a collection of supporting documents to describe it; 2) unlike the lexical terminologies, a child-parent category pair in an search engine has very low chance to explicitly appear in a sentence from a certain text corpus.

Another piece of related work which falls out of the above categories is proposed by Fountain *et al.* [7]. They apply the hierarchical random graph model[5] to infer a taxonomy over 541 nouns. They took a similar philosophy of maximizing likelihood over all the possible tree structures and employ the Markov Chain Monte Carlo Sampling algorithm[10] to get the optimal structure. However, the model applies only

to the scenario where the nouns live at the leave nodes. It cannot provide labels for the internal nodes. In addition, the model also suffers from scalability issues.

5. DISCUSSIONS

In this section, we analyze the properties of our model and discuss the implementation issues.

5.1 Global v.s. Local Optimum

Our taxonomy expansion model achieves the global optimum in the sense of maximizing likelihood. It not only incorporates the probabilistic nature of the problem but also gives exact solution. We cast the category tree into a Bayesian network and formulate the problem of finding the optimal tree as a structure learning problem. Under our model assumption on the conditional probabilities, the optimization in the inference stage is solved by Chu-Liu/Edmond's algorithm precisely.

5.2 Batch-Incremental v.s. Instance Incremental

Our framework should be considered as batch-incremental. After we obtain the set of missing categories, we expand the category hierarchy all at once. We allow hierarchies within the missing categories if the likelihood prefers so. This avoids the possible issues caused by insertion order that any instance-incremental methods would have to deal with. For *e.g.*, if we insert Korean BBQ as a child of Restaurants before we insert Korean Restaurants, there will be no chance that Korean BBQ be a child of Korean Restaurants.

5.3 Flexibility

In the above sections, we discussed a scenario that there is a set of missing categories to be added to a corpus with an existing hierarchy which we want to preserve. In fact, our model is not limited to this scenario. It could be used to refine the existing hierarchy as well. Recall that in the candidate pruning step, we enforce each existing category to have only one possible parent candidate. However, if we adopt the new category candidate pruning rule for the existing categories, our model will be able to correct possible mistakes in an existing hierarchy. In fact, the graph representation and candidate pruning make our model extremely flexible to any pre-defined constraints. In general, any constraint that can be decomposed to edges (which is usually the case) can be easily achieved by candidate pruning.

5.4 Smoothing the Category Distribution

In our taxonomy expansion model, each category c is represented by a multinomial distribution $\boldsymbol{\phi}_c$. The probability $\mathbf{P}(\boldsymbol{\phi}_c|\boldsymbol{\phi}_{parent_{\mathbf{H}}(c)})$ is governed by the Dirichlet distribution, which is defined on the open $(|\mathbf{V}|-1)$-dimensional simplex. This requires that $\boldsymbol{\phi}_c$ does not contain any zero component. However, $\boldsymbol{\phi}_c$'s are sparse in most cases due to the huge vocabulary size. This can be fixed by imposing a smoothing step on the $\boldsymbol{\phi}_c$'s. We apply Dirichlet prior smoothing [27] with prior $\mu = 0.01 \sum_{\mathbf{t}\in d, d\in\mathbf{D}} |\mathbf{t}|$, *i.e.*, 1% of the length of the corpus.

6. EXPERIMENTS

Table 2: Precision and recall of the missing category classifiers. catNaiveBayes is a baseline Naive Bayes model. catSVM is our proposed classifier.

	Precision	Recall
catNaiveBayes	0.82	0.79
catSVM	0.94	0.91

Table 3: Examples of category names and unique names classified by our classifier. New category names (that is, category names that do not exist in $\mathbf{C}^u$) are bolded.

Category Names	Unique Names
science museums,	fuki sushi, best buy,
seafood restaurants,	target, french laundry,
batting cages, electronics,	fry's electronics,
szechuan restaurants	costco

We introduce our data and report our experimental results in this section. First we examine the quality of the discovered missing categories. Then we evaluate our category hierarchy based on annotations from human judges. At the end, we report the statistics of a ranking relevance test which validates that the complete hierarchy boosts ranking performance significantly. We also do a case study to illustrate this effect.

6.1 Data

We use datasets from a commercial local search engine which contains 21, 590, 869 business listings and 1715 existing categories. There are 391, 389 unique terms after filtering out the terms occurring less than 20 times. We collected 6-months click logs for missing category discovery (Section 3.1), category representation (Section 3.2.2), candidate pruning (Section 3.2.4) and the generation of click-based features (Section 3.3.2).

6.2 Missing Category Discovery

In this section, we evaluate the classifier proposed in Section 3.1. We obtain labeled data $\mathbf{T}_{label}$ from human experts and pseudo-labeled data $\mathbf{T}_{pseudo}$ from search click logs. We have $|\mathbf{T}_{label}| = 12K$ and $|\mathbf{T}_{pseudo}| = 50K$. For the feature vector, we use 79K unigrams/bigrams and the average clicks per query session. We use SVM^{light} as the training algorithm.

The performance of the classifier is evaluated by labeled test data generated by human experts. In the test data, there are 2.6K (category, label) pairs. As a baseline method, we evaluate a Naive Bayes model that was used as a query classifier for the search engine. The Naive Bayes model is also a semi-supervised model that uses click feedback data to generate pseudo-labels. However, it was trained with older click logs and only used unigrams as features. It is widely known that SVM usually outperforms Naive Bayes. The purpose of the comparison with Naive Bayes in this experiment is not to qualitatively compare SVM and Naive Bayes but to quantitatively assess the impact of the extra features (bigrams and the click feature).

Table 2 shows the comparison of precision-recall of the two methods. Our proposed method significantly outperforms

the baseline and achieves very high precision and recall. We note that the use of a large amount of user click data for label propagation and the new feature, average clicks per query session are critical to the success of our classifier. Table 3 shows some interesting examples of the new category names discovered by our classifier.

6.3 Category Tree Evaluation

In this section, we evaluate the generated complete category hierarchy $\mathbf{C}^c$ with both qualitative and quantitative study. The only parameter in our model is the concentration parameter α. From the experiments we found that our model is highly insensitive to α. So we fix α to be $0.1|\mathbf{V}|$, *i.e.*, 10% of the size of the the vocabulary, for all the evaluation.

6.3.1 Methods for Comparison

As discussed in related work, the taxonomy expansion problem setting that we study is new, there are no directly comparable algorithms. We adapt a classification-based model as a baseline.

classification We first obtain the missing categories using the same classifier as in Section 3.1. For each new category, we extract the candidate parent set using the same candidate pruning method in Section 3.2.4. Then we do multi-class classification for each new category and output its parent. The features we use include text-based features such as cosine similarity between the category distributions and click-based features such as the co-occurrence count of a child category and parent category in pages of $\tilde{D}^c$.

We examine three variations of our model.

$\mathbf{DIR_{tf}}$ The category distribution takes the term distribution ϕ_c directly. This is the basic version of our model.

$\mathbf{DIR_{tfidf}}$ The category distribution takes the term distribution weighted by the inverse document frequency (idf). In this version, each term has a tfidf score. The category distribution is a multinomial distribution from normalizing the tfidf scores.

$\mathbf{DIR_{sub}}$ The category distribution takes the term distribution weighted by the sub-collection inverse document frequency (idf). By sub-collection we mean the union of the pages belonging to each category in the candidate parent set. Formally, the sub-collection used for a new category c is $\bigcup_{c' \in candidate(c)} \{d|d \in \mathbf{D}_{c'}\}$, where $candidate(c)$ is the candidate parent set for c and $\mathbf{D}_{c'}$ is the set of pages belonging to category c'. For example, suppose we have a new category named Oyster Bar. The candidate parent set could be like {Food & Dining, Restaurants, Seafood Restaurants, Caribbean Restaurants}. Apparently the best parent is Seafood Restaurant since the first two are too broad and the last one is not quite relevant. However, these four categories could all achieve high probabilities of generating Oyster Bar due to the common term "restaurant", which has very high weight in the term distribution. If we use the sub-collection (the collection of pages belonging to the four candidate categories) idf to penalize these high-frequency but low-distinguish-power terms, the category Seafood Restaurant would stand out with the seafood related terms.

6.3.2 Qualitative Study

We show a snapshot of the Entertainment & Arts sub-tree of the complete category hierarchy in Fig. 6.3.2. The missing categories are parenthesized for distinction from the existing categories. Apparently the complete hierarchy makes more sense. We see a lot of meaningful new categories. For example, it would not be a good experience once you want to enjoy the movie "Spider-Man" in IMAX but get results about which all you can tell is they are movie theaters. You can also imagine how a contemporary artist would get really upset when he is looking for museums of contemporary art but gets overwhelmed by those museums with only antiques.

6.3.3 Quantitative Study

To evaluate our model quantitatively, we obtain judgements from human editors which contain 557 missing categories. With varying threshold for the classification scores (for the baseline) and the conditional probabilities $\mathbf{P}(c|parent_{\mathbf{H}^c}(c))$ (for DIR_{tf}, $\text{DIR}_{\text{tfidf}}$, DIR_{sub}), we obtain the precision-recall curve shown in Fig. 6. Here we define precision as the proportion of correct predictions: $p = \frac{\#\text{missing categories with correct predicted parents}}{\#\text{categories evaluated}}$ and recall as the proportion of missing categories retrieved: $r = \frac{\#\text{missing categories retrieved}}{\#\text{missing categories}}$.

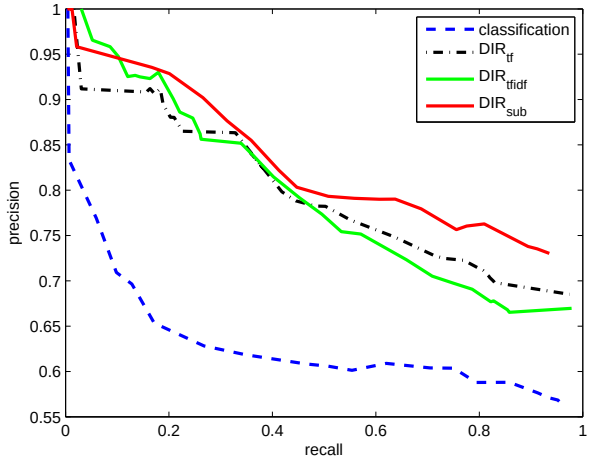


Figure 6: Precision-Recall Curve for Category Tree Evaluation. All the three variations of our model outperform the baseline significantly.

All the three variations of our model significantly outperform the classification-based model. This is because of the intuitive assumption underlying our model which better describes the dependency between a child category and its parent: the expected distribution of a category should be the distribution of its parent. Therefore, unlike the symmetric textual similarity features, our model considers the direction of the relation between two nodes explicitly. At the same time, it models the aggregate behavior of the distributions of all the children for a category.

We observe that $\text{DIR}_{\text{tfidf}}$ performs slightly better when the recall is below 0.1 while DIR_{sub} generally performs better for higher recall. In our experiment setting, a lower recall corresponds to a higher classification score/conditional probability, which indicates a well alignment between the category distributions of a child and a parent. In this case, since all the terms are well aligned, the high-frequency but low-distinguish-power terms have little effect on the prediction. Penalizing these terms might introduce noise and worsen the performance. As the the classification score/conditional

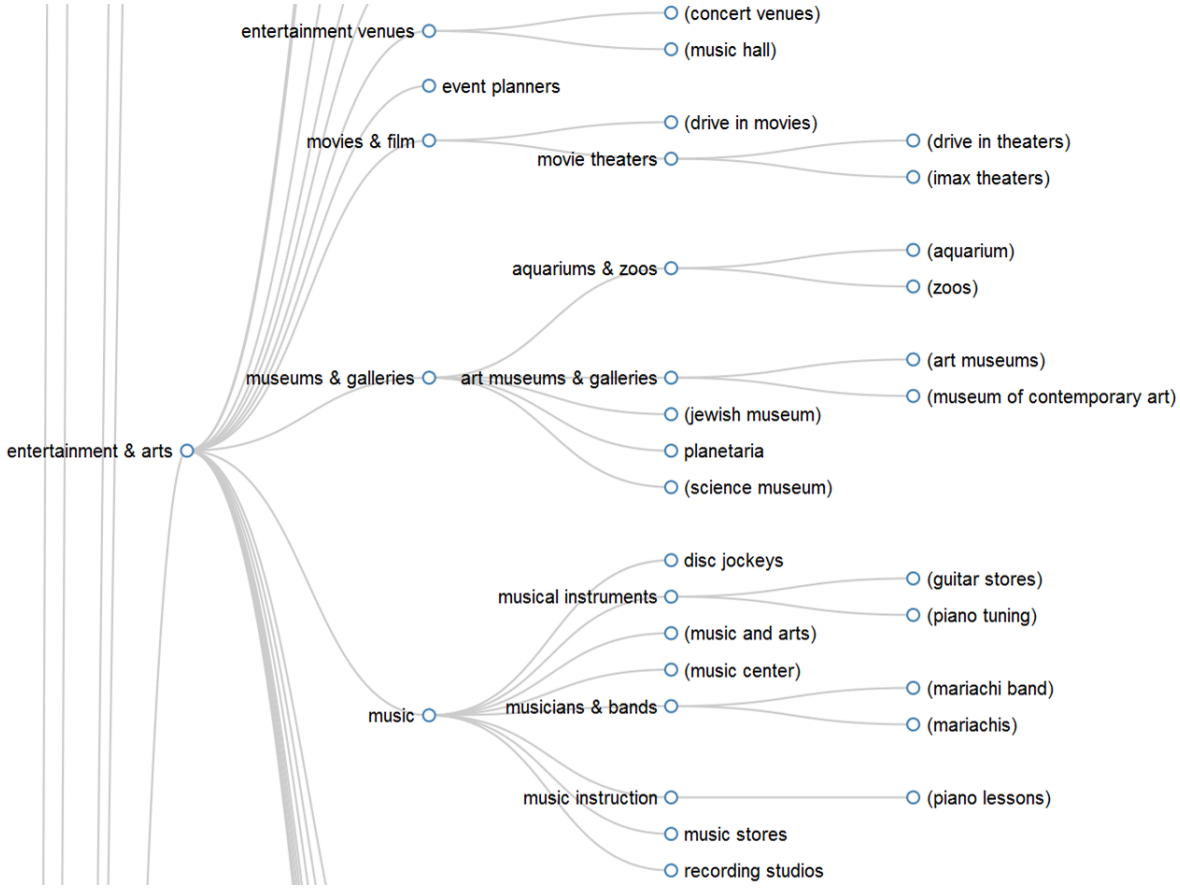


Figure 5: A Snapshot of the Complete Category Hierarchy

probability decreases, these terms would make bigger impact. Then the sub-collection idf weighting would be effective while whole-corpus idf weighting becomes insufficient. This also explains why there is no significant difference between DIR_{tf} and $\text{DIR}_{\text{tfidf}}$.

6.4 Impact on Search Ranking Relevance

Finally, we evaluate the impact of our taxonomy expansion framework on search ranking relevance. A *ranking function* r takes a ranking feature vector $\mathbf{X}_{q,d}$ for a (query q, document d) pair as input and outputs a score. After we obtain $\mathcal{D}^c$ by the item page re-tagging step, we do not re-train a new ranking function. Instead, the difference between $\mathcal{D}^u$ and $\mathcal{D}^c$ is reflected in the feature vector $\mathbf{X}_{q,d}$. For example, CategoryMatch(q, d) is an important ranking feature that represents how well the query q matches the category set $\mathbf{c}$ of d. We have CategoryMatch(Water Parks, (t, {Amusement Parks})) = 0 and CategoryMatch(Water Parks, (t, {Amusement Parks, Water Parks})) = 1 (1 if the query matches at least one of the categories perfectly and 0 otherwise). Each of $\mathcal{D}^u$ and $\mathcal{D}^c$ may generate a different value of this feature for the same item page. Thus, ranking results may differ. We are comparing two search ranking results based on the two search indices generated from $\mathcal{D}^u$ and $\mathcal{D}^c$ respectively.

We randomly sample 100 categories from $\mathbf{C}^m$. We use the category names as queries and generate search ranking results from the two search indices as described above. Let *oldRank* and *newRank* denote the search ranking results generated from $\mathcal{D}^u$ and $\mathcal{D}^c$ respectively. We obtain binary labels (good or bad) for the retrieved results for the sampled queries from human experts. Table 4 shows the comparison of precision@k of *oldRank* and *newRank*. We have significant ranking improvements in all top 5 positions.

Table 4: Ranking improvements measured by precision@k

	p@1	p@2	p@3	p@4	p@5
oldRank	0.79	0.77	0.75	0.73	0.72
newRank	0.87	0.83	0.81	0.78	0.78
gain	9.0%	8.7%	8.2%	7.4%	8.0%

6.5 A Case Study of the Query "Water Park"

Following the previous section on ranking relevance, we do a case study on the query Water Park with location Sunnyvale. Table 5 shows the top 5 retrieved results from $\mathcal{D}^u$ and $\mathcal{D}^c$ respectively. Before we incorporate the missing category Water Park, the top results retrieved are natural parks and swim centers. Once we leverage the complete category set, the tops results become much more relevant.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we address the problem of taxonomy expansion for search engines. Starting from an existing online

Table 5: A Case Study for Water Park with location Sunnyvale

	retrieved results from $\mathcal{D}^u$	retrieved results from $\mathcal{D}^c$
1	Panama Park, Sunnyvale, CA	Great America, Santa Clara, CA
2	Fairwood Park, Sunnyvale, CA	Raging Waters, San Jose, CA
3	Ponderosa Park, Sunnyvale, CA	Waterworld California, Concord, CA
4	Lakewood Park, Sunnyvale, CA	Rapids Water Slides, Pleasanton, CA
5	Washington Park/Swim Center, Sunnyvale, CA	Theater, San Francisco, CA

corpus and an inherent taxonomy, we design a unified framework which discovers missing categories from user queries, expands the existing taxonomy and augments each document's relevant categories in the online corpus. The key aspect of our approach involves a highly intuitive hierarchical Dirichlet model which models the generating process of a taxonomy. Extensive experimental study validates the quality of the generated taxonomy. Evaluation on a ranking relevance test also demonstrates that a better taxonomy could boost retrieval performance significantly.

While a tree structure is widely adopted in many applications, for some categories it could make more sense if we allow multiple parents for them due to the high flexibility of languages and modern concepts. In the future, we would like to extend our work to a more general scenario where a category could have multiple parents.

8. REFERENCES

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. of 20th Intl. Conf. on VLDB*, pages 487–499, 1994.

[2] D. M. Blei, T. L. Griffiths, and M. I. Jordan. The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies. *Journal of the ACM (JACM)*, 57(2):7, 2010.

[3] C. Blundell, Y. W. Teh, and K. A. Heller. Bayesian rose trees. *arXiv preprint arXiv:1203.3468*, 2012.

[4] R. J. Brachman. What is-a is and isn't: An analysis of taxonomic links in semantic networks. *IEEE Computer*, 16(10):30–36, 1983.

[5] A. Clauset, C. Moore, and M. E. Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98–101, 2008.

[6] J. Edmonds. Optimum branchings. *Journal of Research of the National Bureau of Standards B*, 71(4):233–240, 1967.

[7] T. Fountain and M. Lapata. Taxonomy induction using hierarchical random graphs. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 466–476. Association for Computational Linguistics, 2012.

[8] H. N. Gabow, Z. Galil, T. Spencer, and R. E. Tarjan. Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. *Combinatorica*, 6(2):109–122, 1986.

[9] T. Griffiths. Hierarchical topic models and the nested chinese restaurant process. *Advances in neural information processing systems*, 16:106–114, 2004.

[10] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[11] C. Kang, J. Lee, and Y. Chang. Predicting primary categories of business listings for local search. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, CIKM '12, pages 2591–2594, New York, NY, USA, 2012. ACM.

[12] S. Kotz, N. Balakrishnan, and N. Johnson. *Continuous Multivariate Distributions, Models and Applications*. Continuous Multivariate Distributions. Wiley, 2004.

[13] Z. Kozareva, E. Riloff, and E. H. Hovy. Semantic class learning from the web with hyponym pattern linkage graphs. In *ACL*, volume 8, pages 1048–1056, 2008.

[14] W. Li, D. Blei, and A. McCallum. Nonparametric bayes pachinko allocation. *arXiv preprint arXiv:1206.5270*, 2012.

[15] X. Liu, Y. Song, S. Liu, and H. Wang. Automatic taxonomy construction from keywords. In *KDD*, pages 1433–1441, 2012.

[16] X.-L. Mao, Z.-Y. Ming, T.-S. Chua, S. Li, H. Yan, and X. Li. Sshlda: a semi-supervised hierarchical topic model. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 800–809. Association for Computational Linguistics, 2012.

[17] D. Mimno, W. Li, and A. McCallum. Mixtures of hierarchical topics with pachinko allocation. In *Proceedings of the 24th international conference on Machine learning*, pages 633–640. ACM, 2007.

[18] R. Navigli, P. Velardi, and S. Faralli. A graph-based algorithm for inducing lexical taxonomies from scratch. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Three*, pages 1872–1877. AAAI Press, 2011.

[19] J. Nivre, J. Hall, J. Nilsson, A. Chanev, G. Eryigit, S. Kübler, S. Marinov, and E. Marsi. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135, 2007.

[20] Y. Petinot, K. McKeown, and K. Thadani. A hierarchical model of web summaries. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers - Volume 2*, HLT '11, pages 670–675, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[21] S. P. Ponzetto and M. Strube. Deriving a large scale taxonomy from wikipedia. In *Proceedings of the 22nd national conference on Artificial intelligence - Volume 2*, AAAI'07, pages 1440–1445. AAAI Press, 2007.

[22] R. Ramesh and R. Kishore. *Ontologies: a handbook of principles, concepts and applications in information systems*. Integrated series in information systems. Springer, 2007.

[23] A. Ratnaparkhi. A maximum entropy model for Part-Of-speech tagging. In E. Brill and K. Church, editors, *Proceedings of the Empirical Methods in Natural Language Processing*, pages 133–142, 1996.

[24] D. Stewart. *Building Enterprise Taxonomies*. Mokita Press, 2011.

[25] C. Wang, M. Danilevsky, N. Desai, Y. Zhang, P. Nguyen, T. Taula, and J. Han. A phrase mining framework for recursive construction of a topical hierarchy. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '13, pages 437–445, New York, NY, USA, 2013. ACM.

[26] E. Zavitsanos, G. Paliouras, and G. A. Vouros. Non-parametric estimation of topic hierarchies from texts with hierarchical dirichlet processes. *The Journal of Machine Learning Research*, 12:2749–2775, 2011.

[27] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems (TOIS)*, 22(2):179–214, 2004.