Building Hierarchies of Concepts via Crowdsourcing

Yuyin Sun
University of Washington
sunyuyin@cs.washington.edu

Adish Singla
ETH Zurich
adish.singla@inf.ethz.ch

Dieter Fox
University of Washington
fox@cs.washington.edu

Andreas Krause
ETH Zurich
krausea@ethz.ch

Abstract

Hierarchies of concepts are useful in many applications from navigation to organization of objects. Usually, a hierarchy is created in a centralized manner by employing a group of domain experts, a time-consuming and expensive process. The experts often design one single hierarchy to best explain the semantic relationships among the concepts, and ignore the natural uncertainty that may exist in the process. In this paper, we propose a crowdsourcing system to build a hierarchy and furthermore capture the underlying uncertainty. Our system maintains a distribution over possible hierarchies and actively selects questions to ask using an information gain criterion. We evaluate our methodology on simulated data and on a set of real world application domains. Experimental results show that our system is robust to noise, efficient in picking questions, cost-effective, and builds high quality hierarchies.

1 Introduction

Hierarchies of concepts and objects are useful across many real-world applications and scientific domains. Online shopping portals such as [Amazon, 2015] use product catalogs to organize their products into a hierarchy, aiming to simplify the task of search and navigation for their customers. Sharing the goal of organizing objects and information, hierarchies are prevalent in many other domains such as in libraries to organize books [Dewey, 1876] or web portals to organize documents by topics. Concept hierarchies also serve as a natural semantic prior over concepts, helpful in a wide range of Artificial Intelligence (AI) domains, such as natural language processing [Bloehdorn *et al.*, 2005] and computer vision [Deng *et al.*, 2009; Lai *et al.*, 2011b].

Task-dependent hierarchies, as in product catalogs, are expensive and time-consuming to construct. They are usually built in a centralized manner by a group of domain experts. This process makes it infeasible to create separate hierarchies for each specific domain. On the other hand, in the absence of such specific hierarchies, many applications use a general-purpose pre-built hierarchy (for example, WordNet [Fellbaum, 1998]) that may be too abstract or inappropriate for specific needs. An important question in this context is thus *How can we cost-efficiently build task-dependent hierarchies without requiring domain experts?*

Attempts to build hierarchies using fully automatic methods [Blei *et al.*, 2003] have failed to capture the relationships between concepts as perceived by people. The resulting hierarchies perform poorly when deployed in real-world systems. With the recent popularity of crowdsourcing platforms, such as Amazon Mechanical Turk (AMT), efforts have been made in employing non-expert workers (the crowd) at scale and low cost, to build hierarchies guided by human knowledge. Chilton *et al.* [2013] propose the CASCADE workflow that converts the process of building a hierarchy into the task of multi-label annotation for objects. However, acquiring multi-label annotations for objects is expensive and might be uninformative for creating hierarchies. This leads to the second question *How can we actively select simple and useful questions that are most informative to the system while minimizing the cost?*

Most existing methods (including CASCADE) as well as methods employing domain experts usually generate only a single hierarchy aiming to best explain the data or the semantic relationships among the concepts. This ignores the natural ambiguity and uncertainty that may exist in the semantic relationships, leading to the third question *How can we develop probabilistic methods that can account for this uncertainty in the process of building the hierarchy?*

Our Contributions. In this paper, we propose a novel crowdsourcing system for inferring hierarchies of concepts, tackling the questions posed above. We develop a principled algorithm powered by the crowd, which is robust to noise, efficient in picking questions, cost-effective, and builds high quality hierarchies. We evaluate our proposed approach on synthetic problems, as well as on real-world domains with data collected from AMT workers, demonstrating the broad applicability of our system.

The remainder of this paper is structured as follows: After discussing related work in Section 2, we will present our method in Section 3, continue with experiments in Section 4, and conclude in Section 5.

2 Related Work

Concept hierarchies have been helpful in solving natural language processing tasks, for example, disambiguating word

sense in text retrieval [Voorhees, 1993], information extraction [Bloehdorn *et al.*, 2005], and machine translation [Knight, 1993]. Hierarchies between object classes have also been deployed in the computer vision community to improve object categorization with thousands of classes and limited training images [Rohrbach *et al.*, 2011], scalable image classification [Deng *et al.*, 2009; 2013; Lai *et al.*, 2011b], and image annotation efficiency [Deng *et al.*, 2014]. In these methods, it is usually assumed that the hierarchies have already been built, and the quality of the hierarchies can influence the performance of these methods significantly.

The traditional way of hierarchy creation is to hire a small group of experts to build the hierarchy in a centralized manner [Fellbaum, 1998], which is expensive and time consuming. Therefore, people develop automatic or semi-automatic methods to build hierarchies. For instance, vision based methods, such as Sivic *et al.* [2008] and Bart *et al.* [2008], build an object hierarchy using visual feature similarities. However, visually similar concepts are not necessarily similar in semantics.

Another type of methods for hierarchy creation is related to ontology learning from text and the web [Buitelaar *et al.*, 2005; Wong *et al.*, 2012; Carlson *et al.*, 2010]. The goal of ontology learning is to extract terms and relationships between these concepts. However, the focus of these techniques is on coverage, rather than accuracy, and the hierarchies that can be extracted from these approaches are typically not very accurate. Since the taxonomy is the most important relationship among ontologies, many works have been focusing on building taxonomy hierarchies. For example, co-occurrence based methods [Budanitsky, 1999] use word co-occurrence to define the similarity between words, and build hierarchies using clustering. These methods usually do not perform well because they lack in common sense. On the other hand, template based methods [Hippisley *et al.*, 2005] deploy domain knowledge and can achieve higher accuracy. Yet, it is hard to adapt template based methods to new domains. Knowing the fact that humans are good at common sense and domain adaptation, involvement of humans in hierarchy learning is highly necessary and desirable.

The popularity of crowdsourcing platforms has made cheap human resources available for building hierarchies. For example, CASCADE [Chilton *et al.*, 2013] uses multi-label annotations for items, and deploys label co-occurrence to generate a hierarchy. DELUGE [Bragg *et al.*, 2013] improves the multi-label annotation step in CASCADE using decision theory and machine learning to reduce the labeling effort. However, for both pipelines, co-occurrence of labels does not necessarily imply a connection in the hierarchy. Furthermore, both methods can build only a single hierarchy, not considering the uncertainty naturally existing in hierarchies.

Orthogonal to building hierarchies, Mortensen *et al.* [2006] use crowdsourcing to verify an existing ontology. Their empirical results demonstrate that non-expert workers are able to verify structures within a hierarchy built by domain experts. Inspired by their insights, it is possible to gather information of the hierarchy structure by asking simple true-or-false questions about the "ascendant-descendant" relationship between two concepts. In this work, we propose a novel method of hierarchy creation based on asking such questions, and fusing the information together.

3 Approach

The goal of our approach is to learn a hierarchy over a domain of concepts, using input from non-expert crowdsourcing workers. Estimating hierarchies through crowdsourcing is challenging, since answers given by workers are inherently noisy, and, even if every worker gives her/his best possible answer, concept relationships might be ambiguous and there might not exist a single hierarchy that consistently explains all the workers' answers. We deal with these problems by using a Bayesian framework to estimate probability distributions over hierarchies, rather than determining a single, best guess. This allows our approach to represent uncertainty due to noisy, missing, and possibly inconsistent information. Our system interacts with crowdsourcing workers iteratively while estimating the distribution over hierarchies. At each iteration, the system picks a question related to the relationship between two concepts in the hierarchy, presents it to multiple workers on a crowdsourcing platform, and then uses the answers to update the distribution. The system keeps asking questions until a stopping criterion is reached. In this work we set a threshold for the number of asked questions.

3.1 Modelling Distributions over Hierarchies

The key challenge for estimating distributions over hierarchies is the huge number of possible hierarchies, or trees, making it intractable to directly represent the distribution as a multinomial. Consider the number of possible hierarchies of N concepts is $(N+1)^{N-1}$ (we add a fixed root node to the concept set), which results in 1,296 trees for 5 concepts, but already $2.3579e+09$ trees for only 10 concepts. We will now describe how to represent and estimate distributions over such a large number of trees.

Assume that there are N concept nodes indexed from 1 to N, a fixed root node indexed by 0, and a set of possible directed edges $\mathcal{E} = \{e_{0,1}, \ldots, e_{i,j}, \ldots, e_{N,N}\}$ indexed by (i,j), where $i \neq j$. A hierarchy $T \subset \mathcal{E}$ is a set of N edges, which form a valid tree rooted at the 0-th node (we use the terms hierarchy and tree interchangeably). All valid hierarchies form the sample space $\mathcal{T} = \{T_1, \ldots, T_M\}$. The prior distribution π^0 over $\mathcal{T}$ is set to be the uniform distribution.

Due to the intractable number of trees, we use a compact model to represent distributions over trees:

$$P(T|W) = \frac{\prod_{e_{i,j} \in T} W_{i,j}}{Z(W)}, \tag{1}$$

where $W_{i,j}$ is a non-negative weight for the edge $e_{i,j}$, and $Z(W) = \sum_{T' \in \mathcal{T}} \prod_{e_{i,j} \in T'} W_{i,j}$ is the partition function. Given W, inference is very efficient. For example, $Z(W)$ can be analytically computed, utilizing the Matrix Theorem [Tutte, 1984]. This way, we can also analytically compute marginal probabilities over the edges, i.e., $P(e_{i,j})$. The tree with the highest probability can be found via the famous algorithm of Chu and Liu [1965]. A uniform prior is incorporated by initially setting all $W_{i,j}$ to be the same positive value.

Our system maintains a posterior distribution over hierarchies. Given a sequence of questions regarding the structure of the target hierarchy $q^1, \ldots, q^{(t)}, \ldots$, along with the answers $a^1, \ldots, a^{(t)}, \ldots$ the posterior $P(T|W^{(t)})$ at time t is obtained by Bayesian inference

$$P(T|W^{(t)}) \propto P(T|W^{(t-1)}) f(a^{(t)}|T). \tag{2}$$

Hereby, $f(a^{(t)}|T)$ is the likelihood of obtaining answer $a^{(t)}$ given a tree T, specified below to simplify the notation.

So far, we have not made any assumptions about the form of questions asked. Since our system works with non-expert workers, the questions should be as simple as possible. As discussed above, we resort to questions that only specify the relationship between pairs of concepts. We will discuss different options in the following sections.

3.2 Edge Questions

Since a hierarchy is specified by a set of edges, one way to ask questions could be to ask workers about immediate parent-child relationships between concepts, which we call *edge questions*. Answers to edge questions are highly informative, since they provide direct information about whether there is an edge between two concepts in the target hierarchy.

Let $e_{i,j}$ denote the question of whether there is an edge between node i and j, and $a_{i,j} \in \{0, 1\}$ denote the answer for $e_{i,j}$. $a_{i,j} = 1$ indicates a worker believes there is an edge from node i to j, otherwise there is no edge. The likelihood function for edge questions is defined as follows:

$$f(a_{i,j}|T) = \begin{cases} (1-\gamma)^{a_{i,j}} \gamma^{1-a_{i,j}}, \text{ if } e_{i,j} \in T \\ \gamma^{a_{i,j}} (1-\gamma)^{1-a_{i,j}}, \text{ otherwise} \end{cases}, \tag{3}$$

where γ is the noise rate for wrong answers. Substituting (3) into (2) leads to an analytic form to update edge weights:

$$W^{(t)}_{i',j'} = \begin{cases} W^{(t-1)}_{i',j'} (\frac{1-\gamma}{\gamma})^{(2a_{i,j}-1)}, \text{ if } i' = i \wedge j' = j \\ W^{(t-1)}_{i',j'}, \text{ otherwise} \end{cases}. \tag{4}$$

An edge question will only affect weights for that edge. Unfortunately, correctly answering such questions is difficult and requires global knowledge of the complete set (and granularity) of concepts. For instance, while the statement "Is *orange* a direct child of *fruit* in a food item hierarchy?" might be true for some concept sets, it is not correct in a hierarchy that also contains the more specific concept *citrus fruit*, since it separates *orange* from *fruit* (see also Fig. 3).

3.3 Path Questions

To avoid the shortcomings of edge questions, our system resorts to asking less informative questions relating to general, ascendant-descendant relationships between concepts. These *path questions* only provide information about the existence of directed paths between two concepts and are thus, crucially, independent of the set of available concepts. For instance, the path question "Is *orange* a type of *fruit*?" is true independent of the existence of the concept *citrus fruit*. While such path questions are easier to answer, they are more challenging to use when estimating the distribution over hierarchies.

To see, let $p_{i,j}$ denote a path question and $a_{i,j} \in \{0, 1\}$ be the answer for $p_{i,j}$. $a_{i,j} = 1$ indicates a worker believes there is a path from node i to j. The likelihood function is

$$f(a_{i,j}|T) = \begin{cases} (1-\gamma)^{a_{i,j}} \gamma^{1-a_{i,j}}, \text{ if } p_{i,j} \in T \\ \gamma^{a_{i,j}} (1-\gamma)^{1-a_{i,j}}, \text{ otherwise} \end{cases}, \tag{5}$$

where $p_{i,j} \in T$ simply checks whether the path $p_{i,j}$ is contained in the tree T.

Unfortunately, the likelihood function for path questions is not conjugate of the prior. Therefore, there is no analytic form to update weights. Instead, we update the weight matrix by performing approximate inference. To be more specific, we find a W^* by minimizing the KL-divergence [Kullback and Leibler, 1951] between $P(T|W^*)$ and the true posterior:

$$W^* = \arg\min_W KL(P(T|W^{(t)}) \| P(T|W)) \tag{6}$$

It can be shown that minimizing the KL-divergence can be achieved by minimizing the following loss function

$$L(W) = -\sum_{T \in \mathcal{T}} P(T|W^{(t)}) \log P(T|W). \tag{7}$$

Directly computing (7) involves enumerating all trees in $\mathcal{T}$, and is therefore intractable. Instead, we use a Monte Carlo method to estimate (7), and minimize the estimated loss to update W. To be more specific, we will generate i.i.d. samples $\widetilde{\mathcal{T}} = (T_1, \ldots, T_m)$ from $P(T|W^{(t)})$, which defines an empirical distribution $\tilde{\pi}^{(t)}$ of the samples,

$$L_{\tilde{\pi}}(W) = -\sum_{T \in \widetilde{\mathcal{T}}} \tilde{\pi}(T) \log P(T|W), \tag{8}$$

the negative log-likelihood of $P(T|W)$ under the samples.

Sampling Hierarchies from the Posterior

If a weight matrix W is given, sampling hierarchies from the distribution defined in (1) can be achieved efficiently, for example, using a loop-avoiding random walk on a graph with W as the adjacency matrix [Wilson, 1996]. Therefore, we can sample hierarchies from the prior $P(T|W^{(t-1)})$. Noticing that the posterior defined in (2) is a weighted version of $P(T|W^{(t-1)})$, we can generate samples for the empirical distribution $\tilde{\pi}$ via importance sampling, that is, by reweighing the samples from $P(T|W^{(t-1)})$ with the likelihood function $f(a^{(t)}|T)$ as importance weights.

Regularization

Since we only get samples from $P(T|W^{(t)})$, the estimate of (8) can be inaccurate. To avoid overfitting to the sample, we add an ℓ_1-regularization term to the objective function. We also optimize $\Lambda = \log W$ rather than W so as to simplify notation. The final objective is as follows:

$$L^{\boldsymbol{\beta}}_{\tilde{\pi}}(\Lambda^{(t)}) = -\sum_{T \in \mathcal{T}} \tilde{\pi}^{(t)}(T) \log P(T|\Lambda^{(t)}) + \sum_{i,j} \beta |\lambda_{i,j}|. \tag{9}$$

Algorithm 1 Weight Updating Algorithm

Input: $W^{(t-1)}$, an answer $a^{(t)}$, thr for stopping criterion
Non-negative regularization parameters β
Output: $W^{(t)}$ that minimizes (9)
Generate samples $T'_1, \ldots, T'_m$ from $P(T|W^{(t-1)})$
Use importance sampling to get empirical distribution $\tilde{\pi}$
Initialize $\Lambda^0 = \mathbf{0}$, $l = 1$.
repeat
  For each (i, j), set $\delta_{i,j} = \arg\min$ (19);
  Update $\Lambda^{(l)} = \Lambda^{(l-1)} + \Delta$;
  $l = l + 1$;
until $|\Delta| \leq thr$
return $W^{(t)} = \exp(\Lambda^{(l)})$

Optimization Algorithm

We iteratively adjust Λ to minimize (9). At each iteration, the algorithm adds Δ to the original Λ, resulting in $\Lambda' = \Lambda + \Delta$. We optimize Δ to minimize an upper bound on the change in $L^{\beta}_{\tilde{\pi}}$, given by

$$L^{\boldsymbol{\beta}}_{\tilde{\pi}}(\Lambda') - L^{\boldsymbol{\beta}}_{\tilde{\pi}}(\Lambda) \leq \sum_{i,j} [-\delta_{i,j}\tilde{P}(e_{i,j}) + \frac{1}{N} P(e_{i,j}|\Lambda)(e^{N\delta_{i,j}} - 1) + \beta(|\lambda'_{i,j}| - |\lambda_{i,j}|)] + C, \quad (10)$$

where $\tilde{P}(e_{i,j}) = \sum_{T \in \mathcal{T}: e_{i,j} \in T} \tilde{\pi}(T)$ is the empirical marginal probability of $e_{i,j}$, and C is a constant w.r.t. $\delta_{i,j}$. The derivation is presented in [Sun *et al.*, 2015].

Minimizing the upper bound in (19) can be done by analysing the sign of $\lambda_{i,j} + \delta_{i,j}$. By some calculus, it can be seen that the $\delta_{i,j}$ minimizing (19) must occur when $\delta_{i,j} = -\lambda_{i,j}$, or when $\delta_{i,j}$ is either

$$\frac{1}{N} \log \frac{(P(e_{i,j}) - \beta)}{P(e_{i,j}|\Lambda)}, \text{ if } \lambda_{i,j} + \delta_{i,j} \geq 0, \text{or}$$
$$\frac{1}{N} \log \frac{(P(e_{i,j}) + \beta)}{P(e_{i,j}|\Lambda)}, \text{ if } \lambda_{i,j} + \delta_{i,j} \leq 0.$$

This leaves three choices for each $\delta_{i,j}$ – we try out each and pick the one leading to the best bound. This can be done independently per $\delta_{i,j}$ since the objective in (19) is separable. The full algorithm to optimize Λ based on a query answer is given in Algorithm 1.

Theoretical Guarantee

Even though we only minimize a sequence of upper bounds, we prove that (see [Sun *et al.*, 2015]) Algorithm 1 in fact convergences to the true maximum likelihood solution:

Theorem 1. *Assume β is strictly positive. Then Algorithm 1 produces a sequences $\Lambda^{(1)}, \Lambda^{(2)}, \ldots$ such that*

$$\lim_{\ell \to \infty} L^{\beta}_{\tilde{\pi}}(\Lambda^{(\ell)}) = \min_{\Lambda} L^{\beta}_{\tilde{\pi}}(\Lambda).$$

Let $\hat{\Lambda}$ be the solution of Algorithm 1. We next show that if we generate enough samples m, the loss of the estimate $\hat{\Lambda}$ under the true posterior will not be much higher than that obtained by any distribution of the form (1).

Theorem 2. *Suppose m samples $\tilde{\pi}$ are obtained from any tree distribution π. Let $\hat{\Lambda}$ minimize the regularized log loss $L^{\beta}_{\tilde{\pi}}(\Lambda)$ with $\beta = \sqrt{\log(N/\delta)/(m)}$. Then for every Λ it holds with probability at least $1 - \delta$ that*

$$L_{\pi}(\hat{\Lambda}) \leq L_{\pi}(\Lambda) + 2\|\Lambda\|_1 \sqrt{\log(N/\delta)/m}$$

Theorem 4 shows that the difference in performance between the density estimate computed by minimizing w.r.t. $L^{\beta}_{\tilde{\pi}}$ and w.r.t. the best approximation to the true posterior becomes small rapidly as the number of samples m increases.

3.4 Active Query Selection

At each interaction with workers, the system needs to pick a question to ask. The naive approach would be to pick questions randomly. However random questions are usually not very informative since they mostly get *No* answers, while *Yes* answers are more informative about the structure. Instead, we propose to select the question p^* that maximizes information gain over the current distribution $\pi^{(t)}$, i.e.

$$p^* = \arg\max_{p_{i,j}} \min(H(\tilde{\pi}^{(t+1)}_{p_{i,j},1}), H(\tilde{\pi}^{(t+1)}_{p_{i,j},0})) \quad (11)$$

where $H(\cdot)$ is the entropy and $\tilde{\pi}^{(t+1)}_{p_{i,j},a_{i,j}}$ is the posterior distribution over trees after knowing the answer for $p_{i,j}$ as $a_{i,j}$. Note that this criterion chooses the question with the highest information gain using the *less* informative answer, which we found to be more robust than using the expectation over answers. Since we cannot compute the entropy exactly due to the size of the sampling space, we reuse the trees from the empirical distribution $\tilde{\pi}$ to estimate information gain.

3.5 Adding New Nodes to the Hierarchy

New concepts will sometimes be introduced to a domain. In this case, how should the new concept be inserted into an existing hierarchy? A wasteful way would be to re-build the whole hierarchy distribution from scratch using the pipeline described in the previous sections. Alternatively, one might consider adding a row and column to the current weight matrix and initializing all new entries with the same value. Unfortunately, uniform weights do not result in uniform edge probabilities and do thus not correctly represent the uninformed prior over the new node's location.

We instead propose a method to estimate the weight matrix W that correctly reflects uncertainty over the current tree and the location of the new node. After building a hierarchy for $N+1$ nodes, the learned weight matrix is denoted as W_N. We can sample a sequence of trees $(T_1, \ldots, T_m)$ from the distribution $P(T|W_N)$. Now we want to insert a new node into the hierarchy. Since there is no prior information about the position of the new node, we generate a new set of trees by inserting this node into any location in each tree in $(T_1, \ldots, T_m)$. This new set represents a sample distribution that preserves the information of the previous distribution while not assuming anything about the new node. The weight matrix $W^{(0)}_{N+1}$ that minimizes KL-divergence to this sample can then be estimated using the method described in Section 3.3.

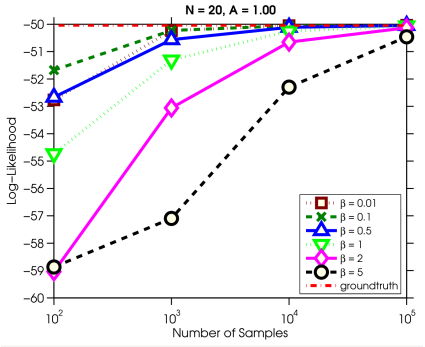

Figure 1: Weight estimation performance for hierarchies with 20 nodes. X-axis is the number of samples given to the algorithm, and β is the regularization coefficient.

4 Experiments

In our experiments, we evaluate four aspects of our approach: (1) the performance of approximate inference to estimate the weights representing distributions over trees; (2) the efficiency of active vs. random query strategies in selecting questions; (3) comparison to existing work; and (4) the ability to build hierarchies for diverse application domains.

4.1 Sample Based Weight Estimation

To evaluate the ability of Algorithm 1 to estimate a weight matrix based on samples generated from a distribution over trees we proceeded as follows. We first sample a "ground truth" weight matrix W, then sample trees according to that weight matrix, followed by estimating the weight matrix W^* using Algorithm 1, and finally evaluate the quality of the estimated matrix. To do so, we sample an additional test set of trees from $P(T|W)$ and compute the log-likelihood of these trees given W^*, where $P(T|W^*)$ is defined in (1). For calibration purpose, we also compute the log-likelihood of these trees under the ground truth specified by $P(T|W)$.

Fig. 1 shows the performance for $N = 20$ nodes, using different values for the regularization coefficient β and sample size m. Each data point is an average over 100 runs on different weights sampled from a Dirichlet distribution (to give an intuition for the complexity of the investigated distributions, when sampling 1 Million trees according to one of the weights, we typically get about $900,000$ distinct trees). The red line on top is the ground truth log-likelihood. As can be seen, the algorithm always converges to the ground truth as the number of samples increases. With an appropriate setting of β, the proposed method requires about $10,000$ samples to achieve a log-likelihood that is close to the ground truth. We also tested different tree sizes N, and got quite similar performance. Overall, we found that $\beta = 0.01$ works robustly across different N and use that value in all the following experiments.

4.2 Active vs. Random Queries

To evaluate the ability of our technique to recover the correct hierarchy, we artificially generate trees and test the performance for different tree sizes, different noise rates for path queries, and active vs. random path queries. To simulate a worker's response to a query, we first check whether the query path is part of the ground truth tree, and then flip the answer randomly using a pre-set noise rate γ. To evaluate how well our approach estimates the ground truth tree, we use the marginal likelihood of the tree edges and compute the Area Under the Curve (AUC) using different thresholds on the likelihood to predict the existence or absence of an edge in the ground truth tree. The marginal likelihood of an edge, $P(e_{i,j}|W) = \sum_{T \in \mathcal{T}, e_{i,j} \in T} P(T|W)$, can be computed in closed form based on the conclusion of the Matrix Theorem [Tutte, 1984]. We also tested different evaluation measures, such as the similarity between the MAP tree and the ground truth tree, and found them to all behave similarly.

Different sized trees of $\{5, 10, 15\}$ nodes are tested to see how the method performs as the problem gets larger. We also test different noise rates, including 0%, 5%, and 10% to verify the robustness of the method. The number of samples for updating the weight matrix is fixed to $10,000$ across all experiments. For each setting, 10 different random ground truth trees are generated. The average results are reported in Fig. 2. The X-axis is the number of questions asked, and the Y-axis is the AUC. $AUC = 1$ means that all edges of the ground truth tree have higher probabilities than any other edges according to the estimated distribution over trees.

As can be seen in the figures, active queries always recover the hierarchy more efficiently than their random counterparts (random queries are generated by randomly choosing a pair of nodes). If there is no noise in the answers, our approach always recovers the ground truth hierarchy, despite the sample based weight update. Note that, in the noise-free setting, the exact hierarchy can be perfectly recovered by querying all N^2 pairs of concepts. While the random strategy typically requires about twice that number to recover the hierarchy, our proposed active query strategy always recovers the ground truth tree using less than N^2 samples.

As N gets larger, the difference between active and random queries becomes more significant. While our active strategy always recovers the ground truth tree, the random query strategy does not converge for trees of size 15 if the answers are noisy. This is due to insufficient samples when updating the weights, and because random queries are frequently answered with *No*, which provides little information. The active approach, on the other hand, uses the current tree distribution to determine the most informative query and generates more peaked distributions over trees, which can be estimated more robustly with our sampling technique. As an indication of this, for trees of size 15 and noise-free answers, 141 out of the 200 first active queries are answered with "yes", while this is the case for only 54 random queries.

4.3 Comparison to Existing Work

We compare our method with the most relevant systems DELUGE [Bragg *et al.*, 2013] and CASCADE [Chilton *et al.*, 2013], which also use crowdsourcing to build hierarchies. CASCADE builds hierarchies based on multi-label categorization, and DELUGE improves the multi-label classification performance of CASCADE. We will thus compare to DELUGE, using their evaluation dataset [Bragg *et al.*, 2013]. This

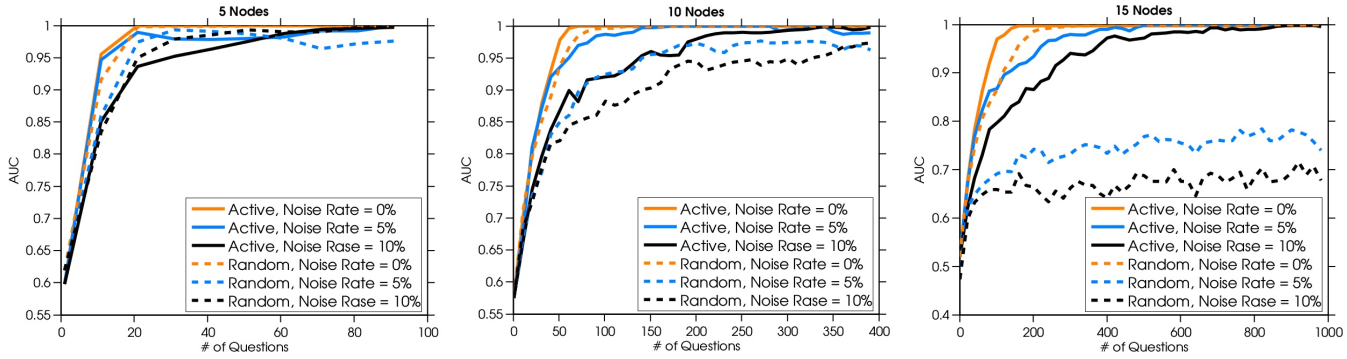

Figure 2: Experimental results comparing active query (solid lines) and random query (dashed lines) strategies for tree sizes ranging from (left) 5 nodes to (right) 15 nodes, using three different noise rates for answers to questions.

dataset has 33 labels that are part of the fine-grained entity tags [Ling and Weild, 2012]. The WordNet hierarchy is used as the ground truth hierarchy.

DELUGE queries many items for each label from a knowledge base, randomly selects a subset of 100 items, labels items with multiple labels using crowdsourcing, and then builds a hierarchy using the label co-occurrence. To classify items into multi-labels, it asks workers to vote for questions, which are binary judgements about whether an item belongs to a category. DELUGE does active query selection based on the information gain, and considers the label correlation to aggregate the votes and build a hierarchy. We use the code and parameter settings provided by the authors of DELUGE.

We compare the performance of our method to DELUGE using different amounts of votes. We compare the following settings: 1) Both methods use 1,600 votes; 2) DELUGE uses 49,500 votes and our method uses 6,000 votes. For the first setting, we pick 1,600 votes for both, as suggested by the authors because DELUGE's performance saturates after that many votes. In the second setting, we compute the results of using all the votes collected in the dataset to see the best performance of DELUGE. We choose 6,000 votes for our method because its performance becomes flat after that.

We compare both methods using AUC as the evaluation criterion. Using 1,600 votes, our method achieves a value of 0.82, which is slightly better than DELUGE with an AUC of 0.79. However, DELUGE does not improve significantly beyond that point, reaching an AUC of 0.82 after 49,500 votes. Our approach, on the other hand, keeps on improving its accuracy and reaches an AUC of 0.97 after only 6,000 queries. This indicates that, using our approach, non-expert workers can achieve performance very close to that achieved by experts (AUC = 1). Furthermore, in contrast to our method, DELUGE does not represent uncertainty over hierarchies and requires items for each label.

4.4 Real World Applications

In these experiments we provide examples demonstrating that our method can be applied to different tasks using AMT. The following pipeline is followed for all three application domains: collect the set of concepts; design the "path question" to ask; collect multiple answers for all possible "path questions"; estimate hierarchies using our approach. Collecting answers for all possible questions enabled us to test different settings and methods without collecting new data for each experiment. AMT is used to gather answers for "path questions". The process for different domains is almost identical: We ask workers to answer "true-or-false" questions regarding a path in a hierarchy. Our method is able to consider the noise rate of workers. We estimate this by gathering answers from 8 workers for each question, then take the majority vote as the answer, and use all answers to determine the noise ratio for that question. Note that noise ratios not only capture the inherent noise in using AMT, but also the uncertainty of people about the relationship between concepts. 5 different path questions are put into one Human Intelligence Task (HIT). Each HIT costs $0.04. The average time for a worker to finish one HIT is about 4 seconds.

The process of building the hierarchies is divided into two consecutive phases. In the first phase, a distribution is built using a subset of the concepts. In the second phase, we use the process of inserting new concepts into the hierarchy, until all concepts are represented. For the body part dataset, we randomly chose 10 concepts belonging to the first phase. For online Amazon shopping and RGBD object data, the initial set is decided by thresholding the frequency of the words used by workers to tag images (15 nodes for Amazon objects and 23 nodes for RGBD objects). The learned MAP hierarchies are shown in Fig. 3.

Representing Body Parts

Here, we want to build a hierarchy to visualize the "is a part of" relationship between body parts. The set of body part words are collected using Google search. An example path question would be "Is *ear* part of *upper body*?". The MAP tree after asking 2,000 questions is shown in the left panel of Fig. 3. As can be seen, the overall structure agrees very well with people's common sense of the human body structure. Some of the nodes in the tree are shown in red, indicating edges whose marginal probability is below 0.75. These edges also reflect people's uncertainty in the concept hierarchy. For example, it is not obvious whether *ear* should be part of the *head* or *face*, the second most likely placement. Similarly, it is not clear for people whether *ankle* should be part of *foot* or *leg*, and whether *wrist* should be part of *arm* or *hand*.

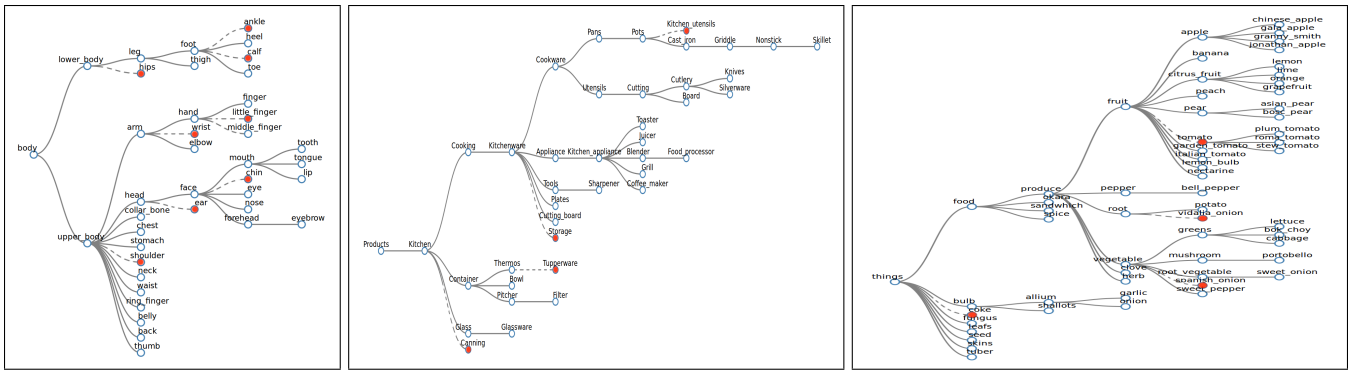

Figure 3: MAP hierarchies representing body parts, amazon kitchen products, and food items (left to right). Red nodes indicate items for which the parent edge has high uncertainty (marginal probability below 0.75). Videos showing the whole process of hierarchy generation can be found on our project page: `http://rse-lab.cs.washington.edu/projects/learn-taxonomies`.

An obvious mistake made by the system is that *ring finger* and *thumb* are connected to the *upper body* rather than *hand*. This is caused by questions such as "Is *ring finger* part of *arm*?", which only 1 out of 8 workers answered with yes. Hence the concept of *ring finger* or *thumb* is not placed into a position below *arm*.

Online Shopping Catalogue

The second task is to arrange kitchen products taken from the Amazon website. There are some existing kitchen hierarchies, for example, the Amazon hierarchy [Amazon, 2015]. However, the words used by Amazon, for example, "Tools-and-Gadgets", might be quite confusing for customers. Therefore, we collected the set of words used by workers in searching for products. We provide AMT workers images of products, and ask them to write down words they would like to see in navigating kitchen products. Some basic preprocessing is done to merge plural and singular of the same words, remove obviously wrong words, and remove tags used less than 5 times by workers because they might be some nicknames used by a particular person. We also remove the word "set", because it is used by workers to refer to a "collection of things" (e.g., pots set, knives set), but not related to the type of products shown in the pictures. The path questions have the form "Would you try to find *pots* under the category of *kitchenware*?" The learned MAP tree is shown in the middle panel of Fig. 3.

Food Item Names

This experiment investigates learning a hierarchy over food items used in a robotics setting [Lai *et al.*, 2011a], where the goal is to learn names people use in a natural setting to refer to objects. Here, AMT workers were shown images from the RGBD object dataset [Lai *et al.*, 2011a] and asked to provide names they would use to refer to these objects. Some basic pre-processing was done to remove noisy tags and highly infrequent words. The path questions for this domain are of the form "Is it correct to say all *apples* are *fruits*?".

The MAP tree is shown in the right panel of Fig. 3. Again, while the tree captures the correct hierarchy mostly, high uncertainty items provide interesting insights. For instance, *tomato* is classified as *fruit* in the MAP tree, but also has a significant probability of being a *vegetable*, indicating people's uncertainty, or disagreement, about this concept. Meanwhile, the crowd of workers was able to uncover very non-obvious relationships such as *allium* is a kind of *bulb*.

5 Conclusion

We introduced a novel approach for learning hierarchies over concepts using crowdsourcing. Our approach incorporates simple questions that can be answered by non-experts without global knowledge of the concept domain. To deal with the inherent noise in crowdsourced information, people's uncertainty, and possible disagreement about hierarchical relationships, we develop a Bayesian framework for estimating posterior distributions over hierarchies. When new answers become available, these distributions are updated efficiently using a sampling based approximation for the intractably large set of possible hierarchies. The Bayesian treatment also allows us to actively generate queries that are the most informative given the current uncertainty. New concepts can be added to the hierarchy at time point, automatically triggering queries that enable the correct placement of these concepts. It should also be noted that our approach lends itself naturally to manual correction of errors in an estimated hierarchy: by setting the weights inconsistent with a manual annotation to zero, the posterior over trees automatically adjusts to respect this constraint.

We investigated several aspects of our framework and demonstrated that it is able to recover high-quality hierarchies for real world concepts using AMT. Importantly, by reasoning about uncertainty over hierarchies, our approach is able to unveil confusion of non-experts over concepts, such as whether *tomato* is a *fruit* or *vegetable*, or whether the *wrist* is part of a person's *arm* or *hand*. We believe that these abilities are extremely useful for applications where hierarchies should reflect the knowledge or expectations of regular users, rather than domain experts. Example applications could be search engines for products, restaurants, and robots interacting with people who use various terms to relate to the same objects in the world. Investigating such application

cases opens interesting avenue for future research. Other possible future directions include explicit treatment of synonyms and automatic detection of inconsistencies.

Acknowledgement

This work was funded in part by the Intel Science and Technology Center for Pervasive Computing, ARO grant W911NF-12-1-0197, ERC StG 307036 and the Nano-Tera.ch program as part of the Opensense II project. We would like to thank Jonathan Bragg for generously sharing the data and code. We would also like to thank Tianyi Zhou for the helpful discussion.

References

[Amazon, 2015] Amazon. `http://www.amazon.com/gp/site-directory/ref=nav_sad`, 2015. [Online; accessed 07-Feb-2015].

[Bart *et al.*, 2008] E. Bart, I. Porteous, P. Perona, and M. Welling. Unsupervised learning of visual taxonomies. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[Blei *et al.*, 2003] D.M. Blei, A.Y. Ng, and M.I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[Bloehdorn *et al.*, 2005] S. Bloehdorn, A. Hotho, and S. Staab. An ontology-based framework for text mining. In *LDV Forum-GLDV Journal for computational linguistics and language technology, 2005, Vol.20, No.1*, pages 87–112, 2005.

[Bragg *et al.*, 2013] J. Bragg, Mausam, and D. Weld. Crowdsourcing multi-label classification for taxonomy creation. In *Proceedings of the First AAAI Conference on Human Computation and Crowdsourcing, HCOMP 2013, November 7-9, 2013, Palm Springs, CA, USA*, 2013.

[Budanitsky, 1999] A. Budanitsky. Lexical semantic relatedness and its application in natural language processing. Technical report, University of Toronto, 1999.

[Buitelaar *et al.*, 2005] P. Buitelaar, P. Cimiano, and B. Magnini. *Ontology Learning from Text: An Overview*, pages 3–12. IOS Press, 2005.

[Carlson *et al.*, 2010] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E.R. Hruschka, and T.M. Mitchell. Toward an architecture for never-ending language learning. In *In AAAI*, 2010.

[Chilton *et al.*, 2013] L. Chilton, G. Little, D. Edge, D. Weld, and J. Landay. Cascade: Crowdsourcing taxonomy creation. In *CHI 2013 Conference on Human Factors in Information Systems*, 2013.

[Chu and Liu, 1965] Y. Chu and T. Liu. On the shortest arborescence of a directed graph. *Science Sinica*, 14, 1965.

[Deng *et al.*, 2009] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[Deng *et al.*, 2013] J. Deng, J. Krause, and L. Fei-Fei. Fine-grained crowdsourcing for fine-grained recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[Deng *et al.*, 2014] J. Deng, O. Russakovsky, J. Krause, M. Bernstein, A. Berg, and L. Fei-Fei. Scalable multi-label annotation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3099–3102, 2014.

[Dewey, 1876] M. Dewey. *Classification and Subject Index for Cataloguing and Arranging the Books and Pamphlets of a Library*. eBook, 1876.

[Fellbaum, 1998] C. Fellbaum. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.

[Hippisley *et al.*, 2005] A. Hippisley, D. Cheng, and K. Ahmad. The head-modifier principle and multilingual term extraction. *Natural Language Engineering*, 11(2):129–157, June 2005.

[Knight, 1993] K. Knight. Building a large ontology for machine translation. In *Proceedings of the Workshop on Human Language Technology*, pages 185–190, 1993.

[Kullback and Leibler, 1951] S. Kullback and R.A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, pages 49–86, 1951.

[Lai *et al.*, 2011a] K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[Lai *et al.*, 2011b] K. Lai, L. Bo, X. Ren, and D. Fox. A scalable tree-based approach for joint object and pose recognition. In *Twenty-Fifth Conference on Artificial Intelligence (AAAI)*, August 2011.

[Ling and Weild, 2012] X. Ling and D.S. Weild. Fine-grained entity resolution. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[Mortensen *et al.*, 2006] J.M. Mortensen, M.A. Musen, and N.F. Noy. Developing crowdsourced ontology engineering tasks: An iterative process, 2006.

[Rohrbach *et al.*, 2011] M. Rohrbach, M. Stark, and B. Schiele. Evaluating knowledge transfer and zero-shot learning in a large-scale setting. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1641–1648, 2011.

[Sivic *et al.*, 2008] J. Sivic, B. Russell, A. Zisserman, I. Ecole, and N. Suprieure. Unsupervised discovery of visual object class hierarchies. In *In Proc. CVPR*, 2008.

[Sun *et al.*, 2015] Yuyin Sun, Adish Singla, Dieter Fox, and Andreas Krause. Building hierarchies of concepts via crowdsourcing (extended version). `http://arxiv.org/abs/1504.07302`, 2015.

[Tutte, 1984] W. Tutte. *Graph Theory*. Addison-Wesley, 1984.

[Voorhees, 1993] E. Voorhees. Using wordnet to disambiguate word senses for text retrieval. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1993.

[Wilson, 1996] D. B. Wilson. Generating random spanning trees more quickly than the cover time. In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, pages 296–303, 1996.

[Wong *et al.*, 2012] W. Wong, W. Liu, and M. Bennamoun. Ontology learning from text: A look back and into the future. *ACM Computing Surverys*, 44(4):20:1–20:36, 2012.

A Derivation of the Objective Function

$$
\begin{aligned}
& L^{\beta}_{\tilde{\pi}}(\Lambda') - L^{\beta}_{\tilde{\pi}}(\Lambda) && (12)\\
=& \sum_{T\in\mathcal{T}} \tilde{\pi}(T)\log P(T|\Lambda) - \sum_{T\in\mathcal{T}} \tilde{\pi}(T)\log P(T|\Lambda') + \sum_{i,j}\beta(|\lambda'_{i,j}| - |\lambda_{i,j}|) && (13)\\
=& \sum_{T\in\mathcal{T}} \tilde{\pi}(T)\log \frac{\exp(\sum_{e_{i,j}\in T}\lambda_{i,j})}{\exp(\sum_{e_{i,j}\in T}\lambda_{i,j}+\delta_{i,j})} + \sum_{T\in\mathcal{T}} \tilde{\pi}(T)\log\frac{Z(\Lambda')}{Z(\Lambda)} + \sum_{i,j}\beta(|\lambda'_{i,j}| - |\lambda_{i,j}|) && (14)\\
=& \sum_{T\in\mathcal{T}} \tilde{\pi}(T)\sum_{e_{i,j}\in T} -\delta_{i,j} + \log\frac{Z(\Lambda')}{Z(\Lambda')} + \sum_{i,j}\beta(|\lambda'_{i,j}| - |\lambda_{i,j}|) && (15)\\
=& \sum_{i,j} -\delta_{i,j}\tilde{P}(e_{i,j}) + \log\frac{\sum_{T'\in\mathcal{T}}\exp(\sum_{e_{i,j}\in T'}\lambda_{i,j}+\delta_{i,j})}{\sum_{T\in\mathcal{T}}\exp(\sum_{e_{i,j}\in T}\lambda_{i,j})} + \sum_{i,j}\beta(|\lambda'_{i,j}| - |\lambda_{i,j}|) && (16)\\
=& \sum_{i,j} -\delta_{i,j}\tilde{P}(e_{i,j}) + \log\sum_{T\in\mathcal{T}} P(T|\Lambda)\exp(\sum_{e_{i,j}\in T}\delta_{i,j}) + \sum_{i,j}\beta(|\lambda'_{i,j}| - |\lambda_{i,j}|) && (17)\\
=& \sum_{i,j} -\delta_{i,j}\tilde{P}(e_{i,j}) + \log\sum_{T\in\mathcal{T}} P(T|\Lambda)\exp(\sum_{e_{i,j}\in T}\frac{1}{N}N\delta_{i,j}) + \sum_{i,j}\beta(|\lambda'_{i,j}| - |\lambda_{i,j}|) && (18)\\
\leq& \sum_{i,j} -\delta_{i,j}\tilde{P}(e_{i,j}) + \frac{1}{N}P(e_{i,j})(e^{N\delta_{i,j}}-1) + \beta(|\lambda'_{i,j}| - |\lambda_{i,j}|) && (19)
\end{aligned}
$$

(2) $\rightarrow$ (3) uses the definition of $P(T|\Lambda)$.
(3) $\rightarrow$ (4) uses the fact that

$$\sum_{T\in\mathcal{T}}\tilde{\pi}(T)\sum_{e_{i,j}\in T} -\delta_{i,j} = \sum_{i,j} -\delta_{i,j}\sum_{T\in\mathcal{T}:e_{i,j}\in T}\tilde{\pi}(T) = \sum_{i,j} -\delta_{i,j}\tilde{P}(e_{i,j}),$$

where $\tilde{P}(e_{i,j})$ is the marginal likelihood of the edge $e_{i,j}$.

To get (7) $\rightarrow$ (8), we use an inequality that, if $x_j \in \mathbb{R}$ and $p_j \geq 0$ with $\sum_j p_j \leq 1$, then

$$\exp(\sum_j p_j x_j) - 1 \leq \sum_j p_j(e^{x_j}-1).$$

Such that

$$\exp(\sum_{e_{i,j}\in T}\frac{1}{N}N\delta_{i,j}) \leq 1 + \sum_{e_{i,j}\in T}\frac{1}{N}(e^{N\delta_{i,j}}-1).$$

The it follows

$$
\begin{aligned}
& \log\sum_{T\in\mathcal{T}} P(T|\Lambda)\exp(\sum_{e_{i,j}\in T}\frac{1}{N}N\delta_{i,j}) && (20)\\
\leq& \log(1+\sum_{T\in\mathcal{T}} P(T|\Lambda)\sum_{e_{i,j}\in T}\frac{1}{N}(e^{N\delta_{i,j}}-1)) && (21)\\
=& \log(1+\frac{1}{N}\sum_{i,j}P(e_{i,j})(e^{N\delta_{i,j}}-1)) && (22)\\
\leq& \frac{1}{N}\sum_{i,j}P(e_{i,j})(e^{N\delta_{i,j}}-1) && (23)\\
& && (24)
\end{aligned}
$$

(11) $\rightarrow$ (12) is true because $\log(1+x) \leq x, \forall x \geq -1$, and

$$\frac{1}{N}\sum_{i,j}P(e_{i,j})(e^{N\delta_{i,j}}-1) \geq \frac{1}{N}\sum_{i,j} -P(e_{i,j}) = -1.$$

B Minimization of (19)

Case 1: $\delta_{i,j} = -\lambda_{i,j}$. Such that (19) becomes

$$\sum_{i,j} \lambda_{i,j}\tilde{P}(e_{i,j}) + \frac{1}{N}P(e_{i,j})(e^{-N\lambda_{i,j}} - 1) \tag{25}$$

Case 2: $\lambda_{i,j} + \delta_{i,j} \geq 0$. Such that (19) becomes

$$\sum_{i,j} -\delta_{i,j}\tilde{P}(e_{i,j}) + \frac{1}{N}P(e_{i,j})(e^{N\delta_{i,j}} - 1) + \beta\delta_{i,j} \tag{26}$$

Take derivative of (26), and set it to be 0:

$$-\tilde{P}(e_{i,j}) + P(e_{i,j})e^{N\delta_{i,j}} + \beta = 0,$$

the solution is

$$\frac{1}{N}\log\frac{\tilde{P}(e_{i,j}) - \beta}{P(e_{i,j})}.$$

Case 3: $\lambda_{i,j} + \delta_{i,j} < 0$. Such that (19) becomes

$$\sum_{i,j} -\delta_{i,j}\tilde{P}(e_{i,j}) + \frac{1}{N}P(e_{i,j})(e^{N\delta_{i,j}} - 1) - \beta\delta_{i,j} \tag{27}$$

Take derivative of (27), and set it to be 0:

$$-\tilde{P}(e_{i,j}) + P(e_{i,j})e^{N\delta_{i,j}} - \beta = 0,$$

the solution is

$$\frac{1}{N}\log\frac{\tilde{P}(e_{i,j}) + \beta}{P(e_{i,j})}.$$

C Proof of Theorem 1

Theorem 3. *Assume β is strictly positive. Then Algorithm 1 produces a sequences $\Lambda^{(1)}, \Lambda^{(2)}, \ldots$ such that*

$$\lim_{\ell\to\infty} L^{\beta}_{\tilde{\pi}}(\Lambda^{(\ell)}) = \min_{\Lambda} L^{\beta}_{\tilde{\pi}}(\Lambda).$$

Proof. First let us define Λ^+ and Λ^- in terms of Λ as follows: for each (i,j), if $\lambda_{i,j} \geq 0$, then $\lambda^+_{i,j} = \lambda_{i,j}$ and $\lambda^-_{i,j} = 0$, and if $\lambda_{i,j} \leq 0$, then $\lambda^+_{i,j} = 0$ and $\lambda^-_{i,j} = -\lambda_{i,j}$. Λ'^+, Λ'^-, $\Lambda^{(\ell)+}$, $\Lambda^{(\ell)-}$, etc. are defined analogously.

Let $F_{i,j}$ denote the (i,j) component in (19), For any Λ and Δ, we have the following:

$$|\lambda + \delta| - |\lambda| = \min\{\delta^+ + \delta^- | \delta^+ \geq -\lambda^+, \delta^- \geq -\lambda^-, \delta^+ - \delta^- = \delta\} \tag{28}$$

Plugging into the definition of $F_{i,j}$ gives:

$$\begin{aligned} F_{i,j}(\Lambda, \Delta) =& -\delta_{i,j}\tilde{P}(e_{i,j}) + \frac{1}{N}P(e_{i,j})(e^{N\delta_{i,j}} - 1) + \beta(|\lambda_{i,j} + \delta_{i,j}| - |\lambda_{i,j}|) && (29)\\ =& \min\{G_{i,j}(\Lambda, \Delta^+, \Delta^-) | \delta^+_{i,j} \geq -\lambda^+_{i,j}, \delta^-_{i,j} \geq \lambda^-_{i,j}, \delta^+_{i,j} - \delta^-_{i,j} = \delta_{i,j}\}, && (30)\end{aligned}$$

where

$$G_{i,j}(\Lambda, \Delta^+, \Delta^-) = (\delta^-_{i,j} - \delta^+_{i,j})\tilde{P}(e_{i,j}) + \frac{1}{N}P(e_{i,j})(e^{N(\delta^+_{i,j} - \delta^-_{i,j})} - 1) + \beta(\delta^+_{i,j} + \delta^-_{i,j}) \tag{31}$$

So, by (19),

$$\begin{aligned} L^{\beta}_{\tilde{\pi}}(\Lambda^{(\ell+1)}) - L^{\beta}_{\tilde{\pi}}(\Lambda^{(\ell)}) \leq& \sum_{i,j} F_{i,j}(\Lambda^{\ell}, \Delta) && (32)\\ =& \sum_{i,j} \min_{\delta_{i,j}} F_{i,j}(\Lambda^{\ell}, \Delta) && (33)\\ =& \sum_{i,j} \min\{G_{i,j}(\Lambda^{\ell}, \Delta^+, \Delta^-) | \delta^+_{i,j} \geq -\lambda^+_{i,j}, \delta^-_{i,j} \geq \lambda^-_{i,j}, \delta^+_{i,j} - \delta^-_{i,j} = \delta_{i,j}\}\} && (34)\end{aligned}$$

Note that $G_{i,j}(\Lambda, \mathbf{0}, \mathbf{0}) = 0$, so none of the terms in this sum can be positive. So the Λ^ℓ's have a convergent subsequence converging to some $\hat{\Lambda}$ such that

$$\sum_{i,j} \min\{G_{i,j}(\Lambda^\ell, \Delta^+, \Delta^-) | \delta^+_{i,j} \geq -\lambda^+_{i,j}, \delta^-_{i,j} \geq \lambda^-_{i,j}, \delta^+_{i,j} - \delta^-_{i,j} = \delta_{i,j}\}\} = 0. \tag{35}$$

It is easy to verify that minimizing $L^\beta_{\tilde{\pi}}(\Lambda)$ is the dual problem of the following convex program:

$$\max_{p_1,\ldots,p_N \in \mathbb{R}^{+N}} \sum_{i=1}^{N} H(p_i) \tag{36}$$

$$s.t. \sum_{i=0}^{N} p(e_{i,j}) = 1, \forall j \tag{37}$$

$$\tilde{P}(e_{i,j}) - P(e_{i,j}) \leq \beta, \forall (i,j) \tag{38}$$

$$P(e_{i,j}) - \tilde{P}(e_{i,j}) \leq \beta, \forall (i,j). \tag{39}$$

We will show that $\hat{\Lambda}^+$ and $\hat{\Lambda}^-$ together with $P(T|\hat{\Lambda})$ satisfy the KKT condition of the previous convex program, and thus form a solution to the prime problem as well as to the dual, the minimization of $L^\beta_{\tilde{\pi}}$. For $P(T|\hat{\Lambda})$, these conditions work out to be the following for all (i,j):

$$\hat{\lambda}^+_{i,j} \geq 0, \tilde{P}(e_{i,j}) - P(e_{i,j}) \leq \beta, \hat{\lambda}^+_{i,j}(\tilde{P}(e_{i,j}) - P(e_{i,j}) - \beta) = 0 \tag{40}$$

$$\hat{\lambda}^-_{i,j} \geq 0, P(e_{i,j}) - \tilde{P}(e_{i,j}) \leq \beta, \hat{\lambda}^+_{i,j}(P(e_{i,j}) - \tilde{P}(e_{i,j}) - \beta) = 0 \tag{41}$$

Since $G_{i,j}(\hat{\Lambda}, \mathbf{0}, \mathbf{0}) = 0$, by (35), if $\hat{\lambda}_{i,j} > 0$ then $G_{i,j}(\Lambda, \Delta^+, \mathbf{0})$ is nonnegative in a neighborhood of $\delta^+_{i,j} = 0$, and so has a local minimum at this point. Such that

$$\frac{\partial G_{i,j}(\Lambda, \Delta^+, \mathbf{0})}{\partial \delta^+_{i,j}}|_{\delta^+_{i,j}=0} = -\tilde{P}(e_{i,j}) + P(e_{i,j}) + \beta = 0. \tag{42}$$

If $\hat{\lambda}^+_{i,j} = 0$, then (35) gives that $G_{i,j}(\hat{\Lambda}, \mathbf{0}, \mathbf{0}) = 0$ for $\delta^+_{i,j} \geq 0$. Thus $\partial G_{i,j}(\Lambda, \Delta^+, \mathbf{0})$ cannot be decreasing at $\delta^+_{i,j} = 0$. Therefore, the partial derivative above must be nonnegative. Altogether, these prove (40). (41) can be proved analogously.

As a whole, we proved that

$$\lim_{\ell \to \infty} L^\beta_{\tilde{\pi}}(\Lambda^{(\ell)}) = L^\beta_{\tilde{\pi}}(\hat{\Lambda}) = \min_{\Lambda} L^\beta_{\tilde{\pi}}(\Lambda).$$

□

D Proof of Theorem 2

Lemma 1. *Suppose samples $\tilde{\pi}$ are obtained from any tree distribution π. Then*

$$|L_{\tilde{\pi}}(\Lambda) - L_\pi(\Lambda)| \leq \sum_{i=0}^{N} \sum_{j=1}^{N} |\lambda_{i,j}||\tilde{P}(e_{i,j}) - P(e_{i,j})|,$$

where $P(e_{i,j}) = \sum_{T \in \mathcal{T}: e_{i,j} \in T} \pi(T)$ and $\tilde{P}(e_{i,j}) = \sum_{T \in \mathcal{T}: e_{i,j} \in T} \tilde{\pi}(T)$.

Proof.

$$L_{\tilde{\pi}}(\Lambda) = \sum_{T \in \mathcal{T}} \tilde{\pi}(T) \log \frac{\exp \sum_{e_{i,j} \in T} \lambda_{i,j}}{Z(\Lambda)} \tag{43}$$

Such that

$$|L_{\tilde{\pi}}(\Lambda) - L_\pi(\Lambda)| = |\sum_{i,j} \lambda_{i,j}(\tilde{P}(e_{i,j}) - P(e_{i,j}))| \leq \sum_{i=0}^{N} \sum_{j=1}^{N} |\lambda_{i,j}||\tilde{P}(e_{i,j}) - P(e_{i,j})|. \tag{44}$$

□

Lemma 2. *Suppose samples $\tilde{\pi}$ are obtained from any tree distribution π. Assume that $|P(e_{i,j}) - \tilde{P}(e_{i,j})| \leq \beta_{i,j}, \forall (i,j)$, Let $\hat{\Lambda}$ minimize the regularized log loss $L^\beta_{\tilde{\pi}}(\Lambda)$. The for every Λ it holds that*

$$L_\pi(\hat{\Lambda}) \leq L_\pi(\Lambda) + 2 \sum_{i=0}^{N} \sum_{j=1}^{N} \beta |\lambda_{i,j}|.$$

Proof.

$$L_\pi(\hat{\Lambda}) \leq L_{\tilde{\pi}}(\hat{\Lambda}) + \sum_{i,j} \beta|\hat{\lambda}_{i,j}| = L^\beta_{\tilde{\pi}}(\hat{\Lambda}) \tag{45}$$

$$\leq L^\beta_{\tilde{\pi}}(\Lambda) = L_{\tilde{\pi}}(\Lambda) + \sum_{i,j} \beta|\lambda_{i,j}| \tag{46}$$

$$\leq L_\pi(\Lambda) + 2\sum_{i,j} \beta|\lambda_{i,j}|. \tag{47}$$

□

(45) to (46) is tree because of the optimality of $\hat{\Lambda}$. (46) to (47) follow from Lemma 1.

Theorem 4. *Suppose m samples $\tilde{\pi}$ are obtained from any tree distribution π. Let $\hat{\Lambda}$ minimize the regularized log loss $L^\beta_{\tilde{\pi}}(\Lambda)$ with $\beta = \sqrt{\log(N/\delta)/m}$. Then for every Λ it holds with probability at least $1-\delta$ that*

$$L_\pi(\hat{\Lambda}) \leq L_\pi(\Lambda) + 2\|\Lambda\|_1\sqrt{\log(N/\delta)/m}$$

Proof. By Hoeffding's inequality, for a fixed pair of (i,j), the probability that $P(e_{i,j}) - \tilde{P}(e_{i,j})$ exceeds β is at most $e^{-2\beta^2 m} = \frac{\delta}{N^2}$. By the union bound, the probability of this happening for any pair of (i,j) is at most δ. Then the theorem follows from Lemma 2. □