# Discovering Hierarchical Structure for Sources and Entities

**Aditya Pal**
apal@us.ibm.com
IBM Research

**Nilesh Dalvi**
nileshd@fb.com
Facebook, Inc.

**Kedar Bellare**
kedarb@fb.com
Facebook, Inc.

## Abstract

In this paper, we consider the problem of jointly learning hierarchies over a set of sources and entities based on their containment relationship. We model the concept of hierarchy using a set of latent binary features and propose a generative model that assigns those latent features to sources and entities in order to maximize the probability of the observed containment. To avoid fixing the number of features beforehand, we consider a non-parametric approach based on the Indian Buffet Process. The hierarchies produced by our algorithm can be used for completing missing associations and discovering structural bindings in the data. Using simulated and real datasets we provide empirical evidence of the effectiveness of the proposed approach in comparison to the existing hierarchy agnostic approaches.

## Introduction

In this paper, we consider the problem of jointly learning hierarchies over a set of data sources and a set of entities based on their containment relationship. We illustrate the problem using an example. Let $W$ be a set of websites and $R$ be a set of restaurants. Consider a bipartite graph on $W \cup R$ where there in an edge $w \rightarrow r$ if the website $w$ mentions the restaurant $r$. The websites can be organized into hierarchies: at the top are general websites that are about all restaurants, e.g. *yelp.com*. Then there are websites specific to cities, cuisine types, etc., e.g. *chefseattle.com* and *4italian.com*. Then there are more specialized websites like *seattleitalian.com*. Similarly, every entity has a set of features associated with it corresponding to its location, cuisine, ambiance, etc. Our goal is to learn a joint hierarchy, similar to the one given in Fig. 1, that can be used to organize both the sources and entities based on their containment bipartite graph.

Hierarchies can be useful in understanding, reasoning, storage, and retrieval of objects (Collins and Quillian 1969) in a variety of domains such as cognitive science, computer vision, databases, and information retrieval. Significant progress has been made on learning hierarchies using machine learning and statistical models (Ho, Eisenstein, and Xing 2012), (Roy et al. 2006), (Hofmann and Buhmann 1996), (Blei, Griffiths, and Jordan 2010), (Willett 1988).

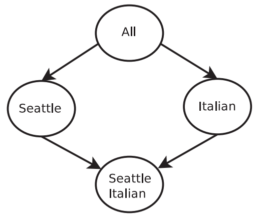


Figure 1: A toy hierarchy for websites and restaurants.

However, most of the existing work have focused on learning tree-based hierarchies. This can be problematic in the cases where the underlying data is generated from a complex structure other than a tree (e.g. Fig. 1). It could lead to a misrepresentation of the underlying structural bindings of the objects. To overcome this limitation, we propose an approach to learn arbitrary hierarchies for a set of sources and entities based on their containment relationship.

Our approach is to represent the hierarchy concepts using sets of binary features. E.g., the hierarchy in Fig. 1 can be represented by labeling node *All* as $\{0, 0\}$, *Seattle* as $\{1, 0\}$, *Italian* as $\{0, 1\}$, and *Seattle* - *Italian* as $\{1, 1\}$. Here 1 means that the feature is important to that node, and 0 means that it is not. Now the two binary features are assigned to the sources and the entities and are placed in the hierarchy at the node with the matching feature. E.g., if *yelp.com* is assigned $\{0, 0\}$ then it is placed at the node *All* in the hierarchy. Intuitively, this makes sense as yelp lists all the restaurants irrespective of their location or cuisine. Now, based on the feature assignments a source is expected to contain all those entities which have atleast all the features that the source cares about. So if source $s_1$ has feature $\{0, 0\}$, $s_2$ has feature $\{1, 0\}$, $s_3$ has feature $\{0, 1\}$, and entity $e$ has feature $\{1, 1\}$, then $s_1, s_2, s_3$ contain $e$.

Based on the binary feature representation, our problem can be formulated as learning binary features that maximize the probability of the observed containment of entities in sources. This problem can be challenging due to *incompleteness* and *errors* in the sources. Thus, the challenge is to model data sources that have a less than perfect precision, and an arbitrarily low recall. Another challenge is that the number of features that characterize sources and entities might not be known a-priori. In order to address these challenges, we propose a generative model that, given an assignment of features to sources and entities, produces the probability of observing a given bipartite graph. The feature assignment to the sources and entities induces a partial order amongst them and this partial order defines the hierarchy. Moreover, instead of fixing the number of features, we consider an infinite set of features. We use the non-parametric *Indian Buffet Process* (Griffiths and Ghahramani 2005) to define a prior over the feature sets.

## Related Work

The problem of latent structure discovery is being tackled in an unsupervised fashion in various domains. The seminal approach in this regard was latent semantic analysis (Deerwester et al. 1990) which projected the document and word vectors to a semantic subspace by using singular value decomposition. Recent approach latent Dirichlet allocation (Blei, Ng, and Jordan 2003) models documents as a collection of topics from which words in the document are generated. Hierarchical LDA (Blei, Griffiths, and Jordan 2010) and Pachinko allocation model (Li and McCallum 2006) extend LDA by organizing topics into hierarchies. These approaches are not applicable to our problem as we aim to learn structure over both sources and entities simultaneously. (Roy et al. 2006) proposed a generative model to learn tree hierarchies for unstructured data. Their approach is not applicable here as we with to learn hierarchies where the objects can lie anywhere and not just at leaves.

Co-clustering based techniques such as correlational clustering (Bansal, Blum, and Chawla 2004), Spectral co-clustering (Dhillon 2001), Bayesian co-clustering (Shan and Banerjee 2008), can be applied to our problem. These techniques cluster sources and entities simultaneously in a way that best explains the co-occurrence information. However, all such approaches are restricted to co-occurrence matrices that exhibit a block diagonal structure. Hence, they do not perform well when applied to problems where both sources and entities follow a hierarchical or directed acyclic graph (DAG) structure. Agglomerative and divisive hierarchical clustering methods (Fernández and Gómez 2008) are another way to organize sources. Such methods, however, do not learn complex hierarchies such as DAGs.

Another class of methods that is related to our work is Non-Negative Matrix Factorization (Paatero and Tapper 1994), (Meeds et al. 2006), which factorizes a matrix into two factor matrices. (Meeds et al. 2006) proposed a matrix factorization of dyadic data in terms of binary feature matrices. They used IBP prior to instantiate the feature matrices. Their model is generic and works for any kind of dyadic data. We use the same prior to instantiate the features for sources and entities. Note that co-clustering or binary matrix factorization approaches cannot be directly applied to our problem because these models ignore the special meaning of the edge between a source and an entity.

## Approach

Let $G = (S \cup E, X)$ be a directed bipartite graph between sources and entities, where $S = \{s_1, \ldots, s_m\}$ is the set of

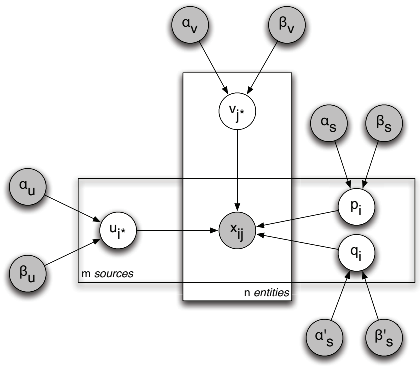


Figure 2: The graphical model structure for the case with finite features.

$m$ sources, $E = \{e_1, \ldots, e_n\}$ is the set of $n$ entities, and $X_{m\times n}$ is the adjacency matrix with entry $x_{ij} = 1$ if the source $s_i$ contains the entity $e_j$, and 0 otherwise. We represent hierarchy using a set of $K$ binary features. Let $F(s_i)$ represent the feature vector of the source $s_i$ and $F(e_j)$ represent the feature vector of the entity $e_j$. Note that both $F(s_i)$ and $F(e_j)$ are length $K$ binary vectors, s.t., if the k$^{th}$ component of $F(s_i) = 1$ it indicates that $s_i$ has feature $k$, otherwise $s_i$ *does not care about the feature* $k$. Using this feature representation, we can model the containment of $e_j$ by $s_i$, i.e. $x_{ij} = 1$, based on the containment relation.

**Definition 1 (Containment relationship)** *A source $s_i$ contains an entity $e_j$ if the entity has at least all those features that are contained in the source.*

$$Contain(s_i, e_j) = \mathbf{1}[F(s_i)^T \cdot \bar{F}(e_j) == 0] \qquad (1)$$

*where $\mathbf{1}[cond]$ is the indicator random variable which returns 1 if the cond is $true$, and 0 otherwise. Note that $\bar{F}(e_j)$ indicates the compliment of the vector $F(e_j)$. Compliment of a vector is flipping 1 to 0 and 0 to 1. So compliment of $\{0\ 1\ 0\ 1\}^T$ is $\{1\ 0\ 1\ 0\}^T$.*

E.g., let a source $s$ have feature vector $\{0, 1, 0, 1\}$ and an entity $e$ have feature vector $\{1, 1, 0, 1\}$. Clearly, $e$ has all the features that $s$ cares about and hence $Contain(s, e) = 1$. So the problem is to learn a function $F : S \cup E \rightarrow 2^K$ that maps each source and entity to the binary feature vectors. Once $F$ has been inferred, we can construct the hierarchy as follows. The image of $F$ is a collection of subsets of $2^K$ binary vectors. Define a partial order $\preceq$ on the image of $F$, where $x \preceq y$ iff

$$Contain(x, y) \wedge \nexists z\, [Contain(x, z) \wedge Contain(z, y)] \qquad (2)$$

where $x, y, z \in Image(F)$. The transitive reduction of the partial order is a directed acyclic graph (DAG) over the image of $F$, where there is an edge $x \rightarrow y$ if $x \preceq y$. We can use this DAG as the graphical representation of the hierarchy of $F$. For instance, given two features $\{seattle, italian\}$, a hierarchy consisting of all possible subsets of these features is precisely the DAG in Fig. 1.

### Generative Model

Let $U_{m\times K}$ and $V_{n\times K}$ be feature matrices for sources and entities, respectively with $K$ features. $F$ is then simply a mapping of sources and entities to appropriate rows in $U$ and $V$. Let $u_{i*}$ be the i$^{th}$ row of $U$ indicating feature vector of $s_i$. Similarly, $x_{i*}$ be the i$^{th}$ row of adjacency matrix $X$. Our generative model for $U, V$ (or $F$) is as follows:

1. Given hyper-parameters $\phi$ generate feature matrices $U, V$ ($\sim$ FeaturePrior($\phi$)). We define the FeaturePrior($\cdot$) in next sub-section for both finite and unknown number of features $K$.
2. For each source $s_i \in S$, draw the edge existence probability $p_i$ and edge error probability $q_i$ as $p_i \sim \text{Beta}(\alpha_s, \beta_s)$ and $q_i \sim \text{Beta}(\alpha'_s, \beta'_s)$, respectively. Let the set of entities in a source $s_i$ be $E_i = \{e : e \in E \wedge Contain(s_i, e)\}$, then entry $x_{ij}$ is generated as:

$$x_{ij} \sim \begin{cases} \text{Bernoulli}(p_i) & \text{if } e_j \in E_i \\ \text{Bernoulli}(q_i) & \text{otherwise} \end{cases} \tag{3}$$

Then the likelihood $P(X, U, V)$ can be factored as

$$P(U, V|\phi) \prod_{i=1}^{m} P(x_{i*}|u_{i*}, V, \alpha_s, \beta_s, \alpha'_s, \beta'_s) \tag{4}$$

For a given source $s_i$, consider following metrics.

$$\begin{aligned} n_i &= |E_i| \\ c_i &= |\{e_j : e_j \in E_i \wedge \mathbf{1}[x_{ij} == 1]\}| \\ \bar{c}_i &= |\{e_j : e_j \notin E_i \wedge \mathbf{1}[x_{ij} == 1]\} \end{aligned}$$

Here $n_i$ represents the number of entities contained in $s_i$, $c_i$ indicates the correctly contained entities (as per the observation $X$) and $\bar{c}_i$ indicates the missing entities that should be contained by $s_i$. Then we can write the likelihood of $x_{i*}$ as

$$\begin{aligned} & P(x_{i*}|u_{i*}, V, \alpha_s, \beta_s, \alpha'_s, \beta'_s) \qquad (5) \\ = & \int P(x_{i*}|u_{i*}, V, p_i, q_i) dP(p_i|\alpha_s, \beta_s) dP(q_i|\alpha'_s, \beta'_s) \\ = & \frac{\text{Beta}(c_i{+}\alpha_s, n_i{-}c_i{+}\beta_s)}{\text{Beta}(\alpha_s, \beta_s)} \times \frac{\text{Beta}(\bar{c}_i{+}\alpha'_s, n{-}n_i{-}\bar{c}_i{+}\beta'_s)}{\text{Beta}(\alpha'_s, \beta'_s)} \end{aligned}$$

where Beta($\cdot, \cdot$) is the Beta function. Next we present our prior $P(U, V|\phi)$ over the feature matrices.

## Prior on Latent Binary Features

In this section, we present two priors for the binary feature matrices $U$ and $V$ in the case of finite (or known) and infinite (or unknown) number of features.

**Finite Features** If the number of features $K$ is known then we use the following generative process:

1. For each feature $k$, generate the feature presence probability for sources ($p_k$) and entities ($q_k$) as $p_k \sim \text{Beta}(\alpha_u, \beta_u)$ and $q_k \sim \text{Beta}(\alpha_v, \beta_v)$, respectively.
2. Each entry $u_{ij}$ in the $U$ matrix and $v_{jk}$ in $V$ matrix is drawn independently as $u_{ik} \sim \text{Bernoulli}(p_k)$ and $v_{jk} \sim \text{Bernoulli}(q_k)$, respectively.

Then the likelihood $P(U, V|\alpha_u, \beta_u, \alpha_v, \beta_v)$ is

$$\prod_{k=1}^{K} \frac{\text{Beta}(m_k{+}\alpha_u, m{-}m_k{+}\beta_u)}{\text{Beta}(\alpha_u, \beta_u)} \frac{\text{Beta}(n_k{+}\alpha_v, n{-}n_k{+}\beta_v)}{\text{Beta}(\alpha_v, \beta_v)} \tag{6}$$

where $m_k$ is the number of sources containing feature $k$ and $n_k$ is the number of entities containing feature $k$. The graphical model structure for the finite feature case is shown in Fig. 2. In our experiments, we set $\alpha_u = \alpha_v = \frac{\alpha}{K}$ and $\beta_u = \beta_v = 1$. Then the conditional probability of a feature $u_{ik}$ is given by:

$$P(u_{ik} = 1|U_{-i,k}) = \frac{m_{-i,k} + \frac{\alpha}{K}}{m + \frac{\alpha}{K}} \tag{7}$$

where $m_{-i,k}$ indicates the number of sources that contain feature $k$ excluding source $s_i$. Similarly we can derive the conditional probability for the entities. Next we introduce the infinite latent feature model which allows the number of features to grow.

**Infinite Features** We use Indian Buffet Process (IBP) to allow the features to grow dynamically. This results in a non-parametric approach to the feature prior. IBP is a generative process which can be described using the following procedure. Assume that there are (potentially) infinite number of dishes in an Indian restaurant. There is a queue of $N$ customers who arrive at the restaurant one after another. The first customer picks the first Poisson($\alpha$) dishes. Then the $i$th customer picks $k$th dish proportional to its popularity $\frac{d_k}{i}$, where $d_k$ is the number of times $k$th dish has been picked previously. In addition, the $i$th customer picks Poisson($\frac{\alpha}{i}$) new dishes. Note that the number of dishes $K$ can grow as more data points are added. It can be shown that the total number of dishes $K \sim \text{Poisson}(N\alpha)$. Further details of IBP can be found in (Griffiths and Ghahramani 2005).

In order to apply IBP in our setting, we combine sources and entities together to get an initial $U, V$. Subsequent updates are then done using the Gibbs sampling. Note that sources and entities are sampled from the same process since we do not want to separate their features. Although, in a real-world scenario, the sources and items might pick features using different distributions, we did not find this assumption to limit our experimental performance. The conditional probability in this case is obtained by taking the limit of Eq. 7 as $K \to \infty$. Hence, $P(u_{ik} = 1|U_{-i,k}) = \frac{m_{-i,k}}{m}$.

## Inference

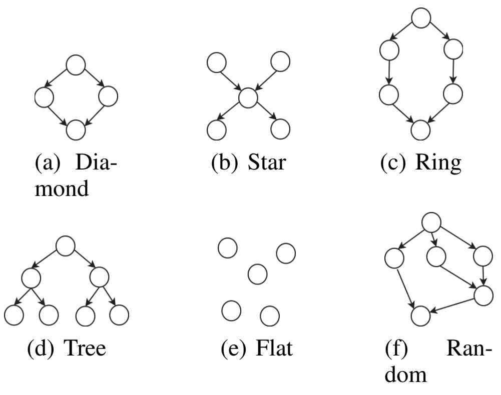
The goal of inference is to determine the best feature matrices $\overline{U}, \overline{V}$ ($\arg\max_{U,V} \{P(X|U, V) \cdot P(U, V)\}$). We use Gibbs sampling to sample $U, V$ from their prior sampled values using the two sampling strategies described below.

**Individual Feature Sampling (IFS)** In this strategy, each entry in the feature matrix is sampled conditioned on the remaining entries in the feature matrices. The update rule can be derived as follows:

$$P(u_{ik}|U_{-i,k}, V, X) \propto P(X|U, V)\, P(u_{ik}|U_{-i,k}) \tag{8}$$

Since $u_{ik}$ takes values from the set $\{0, 1\}$, we consider the ratio $\frac{P(u_{ik}=1|U_{-i,k},V,X)}{P(u_{ik}=0|U_{-i,k},V,X)}$ to update $U$ (similarly $V$).

**Collective Feature Sampling (CFS)** Since, IFS sampling method changes the feature one at a time, it can mix very slowly. Here, we present an algorithm to sample an entire



(a) Diamond (b) Star (c) Ring

(d) Tree (e) Flat (f) Random

Figure 3: Examples of hierarchies.

row in the feature matrix. The update rule for the feature row can be derived as follows,

$$P(u_{i*}|U_{-i*}, V, X) \propto P(X|U,V)\, P(u_{i*}|U_{-i*}) \qquad (9)$$

where the first term $P(X|U,V)$ is evaluated using Eq. 5. The second term can be computed using the IBP prior. For a given configuration of $u_{i*}$, we get

$$P(u_{i*}|U_{-i*}) \propto \prod_{k=1}^{K} m_{-i,k}^{u_{ik}}\, (m - m_{-i,k})^{(1-u_{ik})} \qquad (10)$$

In this case, we need to cycle through $2^K$ possible vectors to pick $u_{i*}$. This sampling strategy will mix faster but can be prohibitive for very large $K$.

**Complexity of Sampling Methods** The IFS method samples individual features and runs in $O(k)$ steps for a given source. As a result one complete IFS iteration runs in $O(mnK^2)$ time. On the other hand the CFS sampling runs in order of $O(mnK2^K)$ time, where the expected number of features $K = O(\alpha(m+n))$.

## Datasets

We consider two datasets for our analysis: (a) *Yahoo! Local Business Listings dataset*, and (b) *Synthetic dataset*. The following subsections contain a brief description of the datasets used to perform our experiments.

### Yahoo! Local Business Listings

We created two datasets, one small and one large, from Yahoo! Local Business Listings (http://local.yahoo.com/).

**YahooBig** We took 700,000 restaurant listings from Yahoo! Local and used the entire Web crawl data to construct a bipartite graph by adding edges between a web page and a restaurant if the web page contains the full 10-digit phone number of the restaurant. The resulting graph had noise because of some false matches of phone numbers with 10 digit numbers on pages. We removed all the web pages with degree less than 5 to reduce some of this noise. The resulting graph had 970,377 nodes and 24,767,397 edges.

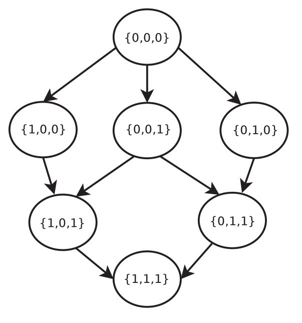


Figure 4: Complex diamond.

**YahooSmall** To analyze the effectiveness of our technique over a moderate size real-world dataset, we selected two business categories *restaurants* and *shopping* and three cities, *Boston*, *Chicago* and *Seattle*. For each city-category pair, we independently sampled 100 listings from the database resulting in a total of 597 entities, as three listings were categorized as both *restaurant* and *shopping*. Next, we constructed the bipartite graph between web pages and the selected entities. The resulting dataset had 21,257 distinct websites containing 597 entities, and 90,774 edges.

### Synthetic Dataset

We also generated synthetic data from several manually constructed hierarchies. The synthetic data allows us to perform a more detailed comparison of our technique in a controlled setting, and to measure its accuracy. Some of the interesting hierarchies that we used are shown in Fig. 3. To generate data from a given hierarchy, we encode it in terms of binary features. For e.g., the Star (see Fig. 3(b)) can be encoded using 4 features, by following labeling: two top layer nodes are labeled with feature $\{1,0,0,0\}$ and $\{0,1,0,0\}$, middle node with feature $\{1,1,0,0\}$, and two bottom nodes with feature $\{1,1,1,0\}$ and $\{1,1,0,1\}$, respectively. Next, we randomly assigned 2000 sources and 2000 entities to a node in the hierarchy. Based on the assignment, sources and entities get the features corresponding to that node. Then bipartite graph $X$ is constructed by using $Contain$ function (see Eq. 1).

To simulate a real-world scenario, we flipped every bit in $X$ from $0 \to 1$ with probability $p_{noise}$ and $1 \to 0$ with probability $p_{inc}$. The probability $p_{noise}$ models the noise in the sources, while $p_{inc}$ models the incompleteness of sources. $\widehat{X}$, that is obtained after bit-flipping in $X$, is given as input to the model with the aim of recovering the true hierarchy.

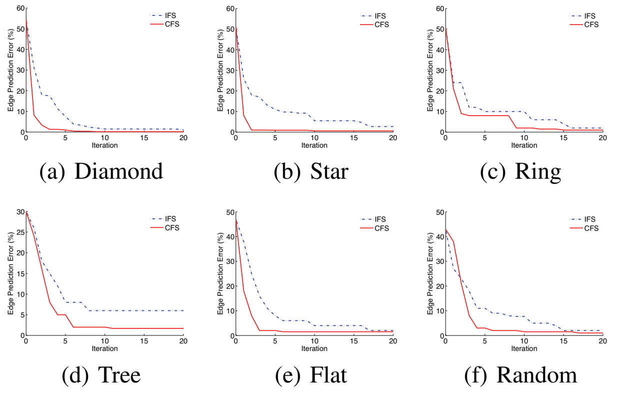
## Experiments: Synthetic Data

### Model Performance

To give a sense of how well our algorithm works, we ran it over the synthetic dataset generated using the hierarchies in Fig. 3. We first set $p_{noise}$ and $p_{inc}$ in the range $[0, 0.1]$. In this case, our model discovered the true hierarchies accurately for fixed as well as unknown number of features, for all synthetic datasets. For higher values of error probabilities, the model learnt slightly complex hierarchies. As an example, the model discovered the hierarchy present in

|  | Our | B1 | B2 | B3 |
|---|---|---|---|---|
| Diamond | **1.5** | 2.6 | 5.2 | 4.7 |
| Star | **1.3** | 3.9 | 7.8 | 8.1 |
| Ring | **1.8** | 5.2 | 5.6 | 6.2 |
| Tree | 2.2 | 2.1 | **1.9** | 2.5 |
| Flat | 1.5 | **1.1** | 1.2 | 1.4 |
| Rand | **3.2** | 4.1 | 4.7 | 5.3 |

Table 1: Edge prediction error (EPE) (%) for our model and the three baseline approaches. b1 = co-clustering, b2 = hierarchical clustering, b3 = traditional clustering.



(a) Diamond (b) Star (c) Ring
(d) Tree (e) Flat (f) Random

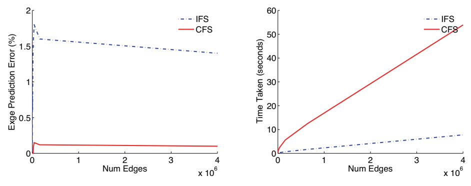
Figure 5: EPE for IFS and CFS over successive iterations.

Fig. 4 when data was generated using hierarchy in Fig 3(a). These complex hierarchies can be simplified using two techniques: (a) eliminating co-related features, and (b) collapsing a hierarchy node if it contains few sources and entities. Indeed in the case of Fig. 4, the hierarchy node $\{0, 0, 1\}$ can be merged with both $\{1, 0, 0\}$ and $\{0, 1, 0\}$ and the nodes $\{1, 0, 1\}$ and $\{0, 1, 1\}$ can be merged with $\{1, 1, 1\}$, leading to the recovery of actual diamond structure. In any case, the model discovered easy-to-interpret hierarchies which gave a clear picture of the structural binding of sources and entities.

## Model Comparison

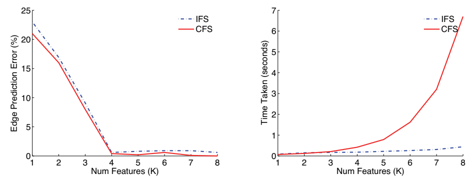
We considered several baseline models for evaluation. Here we report the performance of the best baseline models, which are (b1) Co-clustering (Dhillon 2001), (b2) Hierarchical clustering (Fernández and Gómez 2008), and (b3) Traditional clustering (MacQueen 1967). The output of the baseline algorithm can be used to assign features to source and entities, e.g. if we have $K$ clusters then each source and entity can be assigned a feature vector of size $K$ with the bit corresponding to the cluster in which it lies set to 1. Based on the features, edges between source and entities can be predicted. We used $p_{noise} = p_{inc} = 0.2$ to generate $\widehat{X}$. Finally, we computed the edge prediction error (EPE) by computing the average error in the prediction from $X$. We ran collective feature sampling (CFS) for 20 iterations to generate predictions for our model.

Table 1 shows the EPE for all the models. We see that our model performed better than all the baseline approaches in



(a) EPE (b) Total time taken

Figure 6: Model performance over different *dataset sizes* in learning the $Diamond$ hierarchy.



(a) EPE (b) Total time taken

Figure 7: Model performance over different *number of features* in learning the $Diamond$ hierarchy.

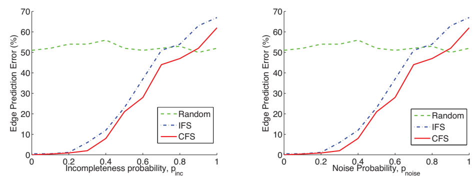
learning complex hierarchies. For the $tree$ and $flat$ hierarchy it has slightly worse performance. This is because the baseline models provide a more parsimonious fit to the data in this case. The result highlights that our approach performs better when data has complex hierarchical structure over approaches that have simpler structural assumptions.

**Effect of Sampling Method** Fig. 5 shows that the CFS converges quickly within 5-10 iterations, whereas IFS takes a lot more iterations and often fails to converge to the solution obtained by CFS. The CFS method is significantly better than IFS for all the hierarchies with $p \sim 0.001$ using paired one-sided $ttest$.

**Effect Of Dataset Size** We varied the number of sources and entities from 10 to 2000 leading to $X$ of size 100 to 4 million, respectively. Fig. 6(a) shows the EPE and Fig. 6(b) shows the total time taken by the algorithm in learning the diamond hierarchy. We observe that CFS has a lower EPE than IFS, even though the time it takes grows rapidly, due to a larger slope, by a factor of $O(2^K/K)$. We also note that more data leads to a better fit over the underlying hierarchy as it reduces EPE.

**Effect of Number of Features** To test the performance over varying number of features, we considered the finite feature model. Fig. 7(a) shows the EPE and Fig. 7(b) shows the total time taken. We note that additional features beyond a certain point do not improve the prediction. A key thing to note is that even though the diamond hierarchy can be represented with 2 features, if we set $K = 3$ then it leads to a poor model fit due to under-learning.

**Effect of Noise** Figs. 8(a) and 8(b) show that the noise and model error shares a positive linear relationship. We note that our model is robust for noise probabilities $\leq 0.3$. If the



(a) EPE (b) Total time taken

Figure 8: Model performance over different noise probabilities in learning the *Diamond* hierarchy.

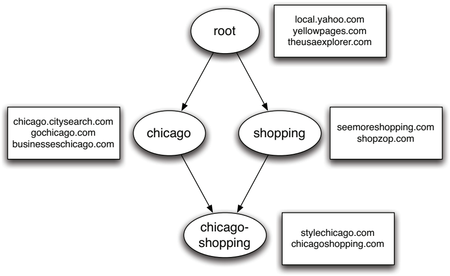


Figure 9: A subset of the hierarchy that is discovered on the real data by our model.

noise is very large, it might be worthwhile to pre-process and remove noisy sources and entities.

## Evaluation: Real Dataset

### Evaluation: YahooBig

The big dataset had 700,000 restaurants and around 10% of them were missing cuisines information. We implemented our approach on the Map-Reduce framework and fixed the number of features to the number of cuisines and initialized the feature vector for each entity based on its cuisine type from the database. Then, we estimated the source features and recomputed entity features using IFS sampling for up to 10 iterations. At the end of our inference, we looked at the final set of vectors for each entity, and assigned the corresponding cuisines to each entity. We were able to assign a cuisine, previously unknown in the database, to *50,201* (around 7%) entities. For instance, we were able to correctly infer that *Bangkok House* (http://local.yahoo.com/info-14034795), which has no cuisine information, as a Thai restaurant. A manual inspection of a random sample showed that the algorithm was accurate 89% of the time. Thus, purely based on the containment relationship between entities and sources, we were able to infer the previously missing cuisines information with a high precision. Note that some of the errors were an artifact of the data construction process: we used phone numbers to match businesses with web pages, and some of the phone numbers we had were incorrect. This result shows how our approach can run for semi-labeled datasets and enrich databases.

| | B | C | S | R | S |
|---|---|---|---|---|---|
| $f_1$ | .95 | .06 | .00 | .49 | .53 |
| $f_2$ | .00 | .96 | .04 | .48 | .53 |
| $f_3$ | .10 | .11 | .88 | .55 | .55 |
| $f_4$ | .31 | .21 | .59 | .75 | .36 |
| $f_5$ | .44 | .39 | .32 | .48 | .69 |

Figure 10: Correlation of features with cities Chicago(C), Boston(B), Seattle(S) and categories Restaurant(R) and Shop(S). The numbers denote the $F$ measure of the given feature in classifying the given attribute.

| Our | B1 | B2 | B3 |
|---|---|---|---|
| **2.7** | 3.6 | 7.2 | 5.1 |

Table 2: Edge prediction error (EPE) (%) for our model and the three baseline approaches for the YahooSmall dataset.

### Evaluation: YahooSmall

**Model Performance** The sample of the hierarchy discovered by our model on this dataset is depicted in Fig. 9. At the top, we see the general websites and a level below are the concepts corresponding to the location and cuisine and so on. Further analysis revealed that several of the entities were shopping malls with coffee shops, hence they intuitively belong to both the categories, as indicated by the discovered hierarchy. These entities were only assigned a single category, indicating that our model was able to supplement the missing information. The resulting hierarchy was more interesting, and justifiably so, than what we had expected a priori from our data. Fig. 10 plots the $F$ measure of the discovered source and entity features with the five dataset attributes. The best baseline had a similar F-measure as our model (no statistically significant difference). However, the baselines do not surface the new concepts that our model discovered, indicating that our model can be used to surface concepts not known a priori.

**Model Comparison** We removed 10% of the edges and computed the EPE for the different models. Table 2 shows that the EPE of our model is lowest in comparison to the baseline, indicating that it is able to discover the underlying hierarchy and supplement the missing information automatically based on the discovered hierarchy.

## Conclusion

This paper proposes a generative model for learning complex hierarchies over dyadic data using latent binary features and a containment relationship over those features. Results on synthetic and real datasets show that our approach provides a parsimonious fit to the hierarchically structured data. It also highlights that such hierarchies do occur in real-datasets and our model can surface much richer concepts that were a-priori unknown, in comparison to hierarchy agnostic approaches. Our model shows that complex hierarchies can be discovered effectively in unsupervised as well as semi-supervised settings for large scale datasets.

## References

Bansal, N.; Blum, A.; and Chawla, S. 2004. Correlation clustering. *Machine Learning* 56:89–113.

Blei, D. M.; Griffiths, T. L.; and Jordan, M. I. 2010. The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies. *J. ACM* 57(2).

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

Collins, A. M., and Quillian, M. R. 1969. Retrieval time from semantic memory. *Journal of Verbal Learning and Verbal Behavior* 8(2):240 – 247.

Deerwester, S. C.; Dumais, S. T.; Landauer, T. K.; Furnas, G. W.; and Harshman, R. A. 1990. Indexing by latent semantic analysis. *JASIS* 41(6):391–407.

Dhillon, I. S. 2001. Co-clustering documents and words using bipartite spectral graph partitioning. In *KDD*, 269–274.

Fernández, A., and Gómez, S. 2008. Solving non-uniqueness in agglomerative hierarchical clustering using multidendrograms. *J. Classification* 25(1):43–65.

Griffiths, T., and Ghahramani, Z. 2005. Infinite latent feature models and the indian buffet process. In *NIPS*.

Ho, Q.; Eisenstein, J.; and Xing, E. P. 2012. Document hierarchies from text and links. In *WWW*, 739–748.

Hofmann, T., and Buhmann, J. M. 1996. Inferring hierarchical clustering structures by deterministic annealing. In *KDD*, 363–366.

Li, W., and McCallum, A. 2006. Pachinko allocation: Dag-structured mixture models of topic correlations. In *ICML*, 577–584.

MacQueen, J. B. 1967. Some methods for classification and analysis of multivariate observations. In *Berkeley Symposium on Mathematical Statistics and Probability*, 281–297.

Meeds, E.; Ghahramani, Z.; Neal, R. M.; and Roweis, S. T. 2006. Modeling dyadic data with binary latent factors. In *NIPS*, 977–984.

Paatero, P., and Tapper, U. 1994. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics* 5(2).

Roy, D. M.; Kemp, C.; Mansinghka, V. K.; and Tenenbaum, J. B. 2006. Learning annotated hierarchies from relational data. In *NIPS*, 1185–1192.

Shan, H., and Banerjee, A. 2008. Bayesian co-clustering. In *ICDM*, 530–539.

Willett, P. 1988. Recent trends in hierarchic document clustering: A critical review. *Information Processing and Management* 24(5):577–597.