Beyond Clustering: Sub-DAG Discovery for Categorising Documents

Ramakrishna B Bairi
IITB-Monash Research Academy
IIT Bombay
Mumbai, India
bairi@cse.iitb.ac.in

Mark Carman
Faculty of IT
Monash University, Victoria
3800, Australia
mark.carman@monash.edu

Ganesh Ramakrishnan
Dept of CSE
IIT Bombay
ganesh@cse.iitb.ac.in

ABSTRACT

We study the problem of generating DAG-structured category hierarchies over a given set of documents associated with "importance" scores. Example application includes automatically generating Wikipedia disambiguation pages for a set of articles having click counts associated with them. Unlike previous works, which focus on clustering the set of documents using the category hierarchy as features, we directly pose the problem as that of finding a DAG structured generative mode that has maximum likelihood of generating the observed "importance" scores for each document where documents are modeled as the leaf nodes in the DAG structure. Desirable properties of the categories in the inferred DAG-structured hierarchy include document coverage and category relevance, each of which, we show, is naturally modeled by our generative model. We propose two different algorithms for estimating the model parameters. One by modeling the DAG as a Bayesian Network and estimating its parameters via Gibbs Sampling; and the other by estimating the path probabilities using the Expectation Maximization algorithm. We empirically evaluate our method on the problem of automatically generating Wikipedia disambiguation pages using human generated clusterings as the ground truth. We find that our framework improves upon the baselines according to the F1 score and Entropy that are used as standard metrics to evaluate the hierarchical clustering.

1. INTRODUCTION

With the exponential growth of digital artifacts (such as texts, images, etc.,) particularly on the Web, hierarchical organization of these artifacts is becoming increasingly important to manage the data. Several real world machine learning applications involve hierarchical categorization of a set of artifacts. Artifacts could be, for example, a set of documents for text classification, a set of genes in functional genomics, or a set of images in computer vision. One can often define a natural category hierarchy to categorize these artifacts. For example, in text and image classification problems, each document or image is assigned a hierarchy of labels — a baseball document would be assigned the labels "baseball", "team sports" and "sports." In many of these applications, the category hierarchy is generated on the entire collection of artifacts [30, 5, 21, 32, 38].

Many techniques have been proposed for category hierarchy creation for organizing digital artifacts. While supervised techniques [13, 36, 16, 11] try to learn the hierarchy from the labeled examples, unsupervised methods [28, 27, 7] try to infer the hierarchy from the data (artifacts) itself, without recourse to human generated labels. Another class of research [3, 2, 31, 39, 18] has been actively looking into adapting a generic "global" category hierarchy (such as Wikipedia/Freebase/DBPedia concept hierarchy) into a specific "local" categorization. These methods are able to associate artifacts (text documents) with closely matching concept nodes from the "global" category hierarchy. The matched concept nodes thus become categories for the artifacts. These methods not only associate curated category names from the "global" category hierarchy, but also bring in rich semantics for the categories from the "global" hierarchy. This motivates us to focus on this class of approaches for category hierarchy generation in our current work. It is easy to find massive (i.e., size in the order of million) DAG structured category hierarchies in practice. Wikipedia's category hierarchy consists of more than 1.5M categories (categories) arranged hierarchically. YAGO [35] and Freebase [9] are other instances of "global" category hierarchies.

It is very common to have some sort of "importance" weights/scores associated with the artifacts in a collection. For example, in a web page collection, each web page may have a click count (the number of times the page has been viewed or accessed) associated with it. In Hierarchical tag visualization [33] applications, leaf tags may have document counts (number of documents assigned to the tag) associated with them. And, in an advertisement collections, each advertisement may have a revenue or cost associated with it. The existing methods for category identification [13, 36, 16, 11, 28, 27, 3, 31, 39, 18] do not focus on the hierarchy generation from the artifact importances. To reflect the artifact importance in the choices of category nodes of the hierarchy, we propose novel approaches to hierarchy generation. Specifically, we propose a generative model for explaining the observed artifact importances based on the hierarchy structure and apply Gibbs sampling and EM methods to estimate the parameter of these models.

Given a DAG-structured category hierarchy and a collection of artifacts with associated importances, we investigate

CIKM '16, October 24–28, 2016, Indianapolis, IN, USA.

DOI: http://dx.doi.org/10.1145/2983323.2983810

the problem of finding a sub-DAG of DAG-structured categories that are induced by the artifacts. This problem arises naturally in several real world applications. For example, consider the problem of identifying appropriate label hierarchies for a collection of articles. Several existing text collection datasets such as the 20 Newsgroups [1] and Reuters-21578 [29] work with a predefined set of categories. We observe that these category names are overly general (too abstract)[1] for the articles categorized under them and fail to induce any reasonable category hierarchy. On the other hand, techniques proposed by systems such as Wikipedia Miner [26] and TAGME [14] generate several labels for each article in the dataset from the Wikipedia (pages and categories) and are highly specific to the article. Collating all labels from all articles to create a label set for the dataset can result in a large number of labels and become unmanageable. We need a hierarchy of these labels to manage the dataset better. Our proposed techniques can discover a suitable label hierarchy (as a sub-DAG) from such large sets of labels using a "global" DAG-structured category hierarchy (such as Wikipedia).

A particularly important application of our work (and the one we use for our evaluations in Section 6) is the problem of *hierarchical* disambiguation page creation in Wikipedia. That is, given a collection of articles spanning different categories but with similar titles, automatically generate a hierarchical disambiguation page for those titles using the Wikipedia category hierarchy[2]. Disambiguation pages[3] on Wikipedia are used to resolve conflicts in article titles that occur when a title is naturally associated with multiple articles on distinct categories. For example, the title Apple[4] can refer to a plant, a company, a film, a television show, a place, a technology, an album, a record label, and a newspaper each of which is associated with its own page on wikipedia. Each disambiguation page organizes articles into several groups, where the articles in each group pertain only to a specific category. Disambiguations may be seen as paths in a category hierarchy leading to different articles that arguably could have the same title. For the "Apple" example, *film*, *television show*, *album*, *record label* are specific types of *Entertainment*. While the *company* and *plant* are types of *Establishments*. The problem then is to organize the articles into multiple hierarchical groups where each leaf group contains articles of similar nature (categories) and has an appropriately discerned group heading. Each article in Wikipedia has an associated click count. Currently, Wikipedia does not consider this while grouping articles on a Disambiguation page since Disambiguation pages are created manually. Our proposed approach is able to leverage this information while generating the hierarchical groups automatically. Figure 1 describes the process of category DAG creation for the disambiguation page for the term "Apple".

All the above mentioned problems can be modeled as the problem of finding the most representative sub-DAG of category nodes from a DAG-Structured category hierarchy. We model this as a two step process. During the first step, we build a generative model that is able to produce the observed importance scores for the artifacts from the DAG-Structured category hierarchy. In this process, certain category nodes in the DAG-Structured category hierarchy become more important than other nodes for generating the observed importance scores for the artifacts. In the second step, we collect those important nodes and the edges that connect them (possibly indirectly) from the DAG-Structured category hierarchy to produce a sub-DAG. The first step is akin to high recall step where the entire DAG-Structured category hierarchy is used to generate the importance scores. Whereas, the second step is akin to the high precision step where a few nodes that contribute maximally to the generation of importance scores are selected.

1.1 Related Work

To the best of our knowledge, the specific problem we consider here is new. Previous works on identifying categories (more generally, topics) can be broadly categorized into one of the following types: a) cluster the artifacts hierarchically and then identify names for each node in the hierarchy; or b) dynamically identify hierarchical topics for a set of artifacts. In the former approach, groups of artifacts form a node in the hierarchy, starting with individual artifacts at the leaves, similar artifacts are groped together to form higher level nodes. In the later approach, a group of high probability words are associated to each identified topic. A name can be assigned to a topic by manually inspecting the words or using additional algorithms like [24, 22].

Hierarchical LDA [7] infers a topic hierarchy by modeling term occurrences in documents using the nested Chinese restaurant process. Non parametric extensions of LDA include the Hierarchical Dirichlet Process [37] mixture model, which allows the number of topics to be unbounded and learnt from the data; In these approaches, unlike our proposed approach, an existing topic hierarchy is not used and the existing artifact-topic information is not leveraged.

The Pachinko Allocation Model (PAM)[20] captures arbitrary, nested, and possibly sparse correlations between topics using a DAG. The leaves of the DAG represent individual words in the vocabulary, while each interior node represents a correlation among its children, which may be words or other interior nodes (topics). PAM learns the probability distributions are words for a topic, subtopics for a super topic, and topics in a document. We cannot, however, generate a subset of topics from a large existing topic DAG that can act as summary topics, using PAM, due to the presence of too many nodes leading to prohibitively long execution time.

HSLDA [27] introduces a hierarchically supervised LDA model to infer hierarchical labels for a document. It assumes an existing label hierarchy in the form of a tree. The model infers one or more labels such that, if a label l is inferred as relevant to a document, then all the labels from l to the root of the tree are also inferred as relevant to the document. Our approach differs from HSLDA since: (1) we use the label hierarchy to infer a set of labels for a *group* of documents; (2) we do not enforce the label hierarchy to be a tree as it can be a DAG; and (3) we do not have labeled data (We use the ground truth from the Wikipedia disambiguation dataset only for evaluating the algorithm, and not for training the model.)

Wei and James [6] present a hierarchical multi-label classification algorithm that can be used on both tree and DAG structured hierarchies. They formulate a search for the op-

[1]Category *Science* is more general than the category *Chemistry* which is more abstract than the category *BioChemistry*
[2]http://en.wikipedia.org/wiki/Help:Categories
[3]http://en.wikipedia.org/wiki/Wikipedia:Disambiguation
[4]http://en.wikipedia.org/wiki/Apple_(disambiguation)

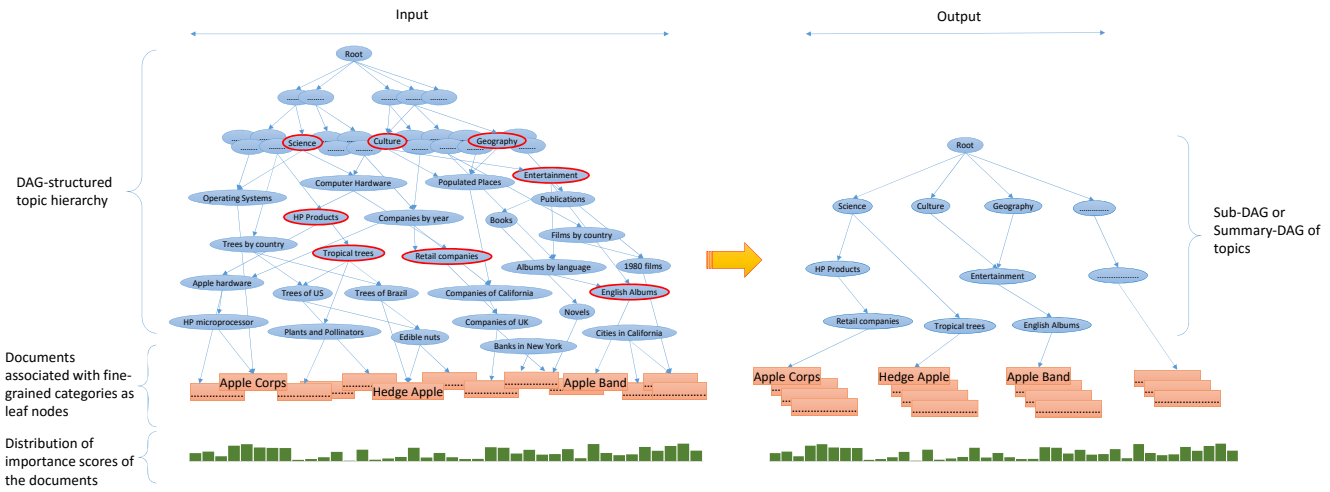

Figure 1: Overview of sub-DAG selection: Documents with the importance scores associated as leaf nodes of DAG-structured category hierarchy is given as input to our method. We generate a sub-DAG from the DAG-structured category hierarchy such that the documents generated from this sub-DAG (using the estimated parameters) will have the distribution of importance scores as close as possible to the observed distribution.

timal consistent multi-label for each example as the problem of finding of the best subgraph in a tree/DAG. In our approach, we assume, individual documents are already associated with one or more topics and we find a consistent label set for a group of documents using the DAG structured topic hierarchy.

Medelyan et al. [23] and Ferragina et al. [14] detect topics for a document using Wikipedia article names and category names as the topic vocabulary. These systems are able to extract signals from a text document and identify Wikipedia articles and/or categories that optimally match the document and assign those article/category names as topics for the document. When run on a large collection of documents, these approaches generate enormous numbers of topics, a problem our proposed approach addresses.

Bairi et. al. [2] summarize a set of topics from Wikipedia that can be viewed as a summary topics for the documents in the collection. However, their method generates a flat set of topics and does not form a hierarchy, whereas our approach can create a hierarchical summary of topics.

1.2 Our Contributions

While most prior works discussed above focus on the underlying set of documents, (e.g., by clustering documents), we focus directly on the categories. In particular, we formulate the problem as sub-DAG selection on the set of categories organized in a DAG structured hierarchy, while simultaneously considering the documents to be categorized. Our approach is based on generative models which have been successfully used in applications such as document modeling [40], but have to the best of our knowledge never been applied to subsetting a category DAG to create a sub-DAG describing the document collection. By modeling the DAG-structured category hierarchy as a Markov network, we introduce a procedure to estimate the marginal probabilities of the nodes in generating the observed grouping of the documents along with their importance scores[5]. Our approach is based on Gibbs sampling with *path* constraints to ensure root-to-leaf path for every document is maintained. Unlike other methods [27, 8, 20, 40] we do not observe the words in the documents. The co-occurrence statistics of the leaf level categories in the documents drive the sampler to sample common ancestors more frequently, thus increasing the marginal probabilities of those nodes. While the path sampling approaches [17, 19] fail in the case of massive DAG-structured hierarchies (such as Wikipedia category hierarchy,) due to extremely large number of paths, our approach is able to do local sampling (within the Markov Blanket). We also present an EM based method to estimate the expected importance of every edge in the DAG-structured category hierarchy based on the importance scores of the documents. Using the expected importance of edges in the DAG we propose a technique for a sub-DAG generating. From an empirical perspective, we introduce and evaluate our approach on a dataset of around 800 disambiguations that was extracted from Wikipedia and subsequently cleaned using the methods described in the experimentation section. We show that our method outperforms other baselines, and is practical enough to be used on large corpora.

[5]if documents don't have importance scores associated with them then their importance is simply set to be the same

2. PROBLEM FORMULATION

Let $G(V; E)$ be the DAG structured category hierarchy with V categories. These categories are observed to have a parent child (isa) relationship forming a DAG. Let D be the set of documents that are associated with one or more of these categories. The left part of Figure 1 depicts a category hierarchy with associated documents. If a document is attached to a category t, we assume that all the ancestor categories of t are also relevant for that document. This assumption has been employed in earlier works [7, 6, 30] as well. Furthermore, we assume that there exists a function associating importance scores with every document node. Examples of scores can be click counts of the documents, number of likes given to the document, etc. Given a budget of K, our objective is to choose a DAG of K categories from $G(V; E)$ that best describes the documents in D. The notion

of best describing categories is characterized through a generative process which can generate the observed importance scores at the document nodes.

It is easy to find massive DAG structured category hierarchies in practice. Wikipedia's category hierarchy consists of more than 1M categories (categories) arranged hierarchically. In fact, they form a cyclic graph [42]. However, we can convert the graph to a DAG by eliminating the cycles. YAGO [35] and Freebase [9] are other instances of massive category hierarchies. The association of the documents with the existing category hierarchy is also well studied. Systems such as WikipediaMiner [26], TAGME [14] and several annotation systems such as [12, 25, 10] attach categories from Wikipedia (and other catalogs) to the documents by establishing links between them.

Our goal is the following: Given,

1. A DAG-structured category hierarchy $G(V;E)$ associated with the categories and documents, over $V = V_{\text{categories}} \cup V_{\text{docs}}$ nodes, which contains $\left|V_{\text{categories}}\right|$ internal nodes (categories) denoted:$c_1, ..., c_{\left|V_{\text{categories}}\right|}$ and $|V_{\text{docs}}|$ leaf nodes (documents) denoted:$d_1, ..., d_{|V_{\text{docs}}|}$
2. A function associating scores with each of the document nodes: $Count(d_i) \in \mathbf{N}_+$. Note, we assume without loss of generality that the scores are *positive integers*. Any set of positive real number scores can be appropriately scaled and rounded to produce integer scores.

Estimate a model that can predict the observed scores, i.e.:

1. Associate a Bayesian Network over binary variables with the structure given by the DAG above, (that is, let X_i be a binary variable corresponding to node c_i and Y_j be a binary variable corresponding to node d_j).
2. Determine the parameters of a Bayesian Network, that is, $P(X_i|parents(X_i))$ and $P(Y_j|parents(Y_j))$ such that the leaf nodes have marginal probability proportional to their scores, that is, $P(Y_j = 1) = Count(d_j)/\sum_k Count(d_k)$.
3. Construct a sub-DAG of K nodes by first selecting K nodes from the Bayesian Network that have maximum marginal probability and entropy over its children and then interconnecting them with the edges from the DAG. These nodes contribute most in predicting the observed score.

In this work we make a simplifying assumption that the leaf variables (i.e., documents) don't co-occur and have frequency given by their counts. That is, we prepare example data for learning the network as unit vectors of the form: - $(Y_1, ..., Y_N) = (0, ..., 0, 1, 0, ...)$ - where $P(Y_i = 1) = Count(d_i)/\sum_j Count(d_j)$

3. GIBBS SAMPLING BASED PARAMETER ESTIMATION

Given a DAG structured category hierarchy $G = (V, E)$ and a set of documents (with importance scores) attached as the leaf nodes in the category DAG, we assume a Bayesian Network (BN) with category nodes. Our goal is to estimate the parameters of this BN such that, a generative process using this BN to assign to the binary leaf nodes (documents) is able to produce the desired marginal distribution given by the observed document scores. With this we mean that, repeated sampling of nodes from this BN (with the estimated parameters) is able to produce the leaf nodes (documents) the number of times proportional to their importance scores. After estimating the parameters of the BN, in the next step, we produce a ranking of the category nodes using their marginal probabilities along with entrop of their children to construct a sub-DAG of K nodes with approximately the same marginal distribution (given by the observed document scores) over the leaves. This step is described in Section 5. In this section, we focus on estimating the parameters of the BN.

Let $X = (X_v)_{v \in V}$ be a set of random variables indexed by the category nodes V. For a BN with respect to G, its joint probability density function (with respect to a product measure) can be written as a product of the individual density functions, conditioned on their parent variables:

$$p(x) = \prod_{v \in V} p\left(x_v \mid x_{\pi(v)}\right) \tag{1}$$

where $\pi(v)$ is the set of parents of v (i.e. those vertices pointing directly to v via a single edge).

Here $p(x)$ is the probability of observing a particular assignment of categories to a document. Specifically,

$$p(X_1 = x_1, \ldots, X_n = x_n) = \prod_{v=1}^{n} p\left(X_v = x_v \mid \forall_{j \in \pi(v)}\ X_j = x_j\right) \tag{2}$$

All the observations x_i are binary, taking values 0 or 1. If $x_i = 1$, the i^{th} category is assigned to the document; otherwise, it is not assigned to the document. Hence, there will be $2^{\pi(i)}$ number of parent configurations for any category X_i.

Once the parameter θ of this BN, i.e. $p\left(x_v \mid x_{\pi(v)}\right)$ for all the categories in the G has been estimated, the likelihood of observing the document collection D from this BN can be computed as:

$$LL(X|\theta) = \sum_{d \in D} \sum_{v \in V} \log\left(p\left(X_v = x_v \mid X_{\pi(v)} = x_{\pi(v)}\right)\right) \tag{3}$$

Given an assignment of valves to the variables in a BN, a Gibbs sampler simply iterates over each latent category X_i (note that there is one BN for each document) sampling a new value for the variable according to its posterior distribution:

$$X_i \sim \text{Bernoulli}(P), \text{ where} \tag{4}$$

$$P = p(X_i = 1 \mid X_{-i}) \tag{5}$$

$$1 - P = p(X_i = 0 \mid X_{-i}) \tag{6}$$

Here X_{-i} denotes all nodes in the BN except for X_i.

The conditional independence property of a BN states that any two nodes (u, v) are conditionally independent given a set of variables z which d-separates them:

$$X_u \perp\!\!\!\perp X_v \mid X_Z \tag{7}$$

The Markov Blanket of node v, denoted $\mathcal{MB}(v)$, is the minimal set of nodes which d-separates node v from all other nodes. Using this property, Equation 5 and 6 simplify to:

$$P = p\left(X_i = 1 \mid X_{\mathcal{MB}(i)}\right) \tag{8}$$

$$1 - P = p\left(X_i = 0 \mid X_{\mathcal{MB}(i)}\right) \tag{9}$$

Furthermore, the conditional distribution of one variable given all others is proportional to the joint distribution:

$$p\left(x_j | x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n\right) \propto p\left(x_1, \ldots, x_n\right)$$

"Proportional to" in this case means that the denominator is not a function of x_j and thus is the same for all values of x_j; it forms part of the normalization constant for the distribution over x_j.

Applying this principle to Equation 8, we have

$$p\left(X_i = 1 \mid X_{-i}\right) \propto p\left(X_i, X_{\mathcal{MB}(i)}\right) \tag{10}$$

Using the factorization stated in Equation 1 and 2, we have

$$\begin{aligned} p\left(X_i{=}1 \mid X_{-i}\right) &\propto p\left(X_i{=}1, X_{\mathcal{MB}(i)}\right) \\ &\propto p\left(X_i{=}1 \,|\, X_{\pi(i)}\right) \prod_{k \in \text{children}(i)} p\left(X_k \mid X_{\pi(k)}\right) \end{aligned} \tag{11}$$

Similarly, $p\left(X_i = 0 \mid X_{-i}\right)$ is computed. Note that, the normalization constant is computed by simply summing the right side of Equation 11 for $X_i = 1$ and $X_i = 0$.

The Gibbs Sampler samples repeatedly from the posterior in Equation 11. When the Gibbs Sampler reaches a steady state, we would have estimated the parameters of the BN, i.e. $p\left(x_v \,\middle|\, x_{\pi(v)}\right)$ for all $v \in V$.

The Algorithm 1 describes the Gibbs Sampling. The algorithm iteratively samples the categories for the documents as per the posterior distribution in Equation 11. In order to reflect the importance scores of the documents in the posterior estimation, we create $M = \sum_d \text{count}(d)$ instances of the BN one for each of the training examples. Similar to the collapsed Gibbs sampler for the LDA [34] we create counters to hold the number of times a category is sampled for a document (lines 7-13). Repeatedly sampling the latent categories for the documents as per the estimated posterior (from the counter values accumulated so far) and then updating the counters based on the samples observed, makes the estimated posterior converge to its true posterior (lines 15-32.) To ensure root to leaf path in the categories sampled for a document, we enforce two constraints during the sampling (line 28):

- Set $X_{di} = 1$ if there exist a child X_k of X_i such that $X_{dk} = 1$, but all of its parents are set to 0, i.e. $X_{d\pi(k)} = 0$

  This constraint ensures that the parent category is set to 1 if there is a child whose all parents are set to 0.

- Set $X_{di} = 0$ if there does not exist a child X_k of X_i such that $X_{dk} = 1$

  This constraint ensures that, parent is set to 0, if none of its children are set to 1

The process of sampling and updating counters is done until we observe stability in the log likelihood [34]. At the termination the algorithm computes the posterior probability distribution, which reflects the BN parameters.

Algorithm 1 Gibbs sampling for modeling click counts

1: **Input** : DAG structured category hierarchy $G(V, E)$
2: observed categories for documents $\{C_1, ..., C_n\}$
3: importance scores for documents $\{\eta_1, ..., \eta_n\}$
4: **Output** : Parameters of BN
5: Create training instances by repeating documents:
6: $D = \{d_{1,1}, \ldots, d_{1,\eta_1}, d_{2,1}, \ldots, d_{2,\eta_2}, ...\}$
7: Let $X_{di} \in \{0, 1\}$ denote the current assignment of category random variable X_i for document $d \in D$
8: Let $X_{d\pi(i)} \in \{0, ..., 2^{|\pi(i)|} - 1\}$ be a variable representing the configuration of parents of X_i in document d.
9: ▷ Set observed categories and all ancestors to true:
10: Set $X_{di} = 1$ if $X_i \in C_d \vee X_i \in \pi^*(C_d)$
11: Initialize $X_{di} \sim \text{Uniform}(\{0, 1\})$ for all $d \in D$ and $i \in V$
12: Initialize counts:
$N_{iJ} = \sum_d 1(X_{di} = 1 \wedge X_{d\pi(i)} = J)$
$N_J = \sum_d 1(X_{d\pi(i)} = J)$
13: **for** $d \in D$ **do**
14: **for** $i \in V$ **do**
15: **if** $X_i \notin latentVariableSet(d)$ **then**
16: **continue**
17: **end if**
18: ▷ Remove current assignment to X_{di} from N_{iJ}
19: $J = X_{d\pi(i)}$
20: **if** $X_{di} = 1$ **then**
21: $N_{iJ} = N_{iJ} - 1$
22: **end if**
23: $N_J = N_J - 1$
24: ▷ Re-sample X_{di}
25: $X_{di} \sim \text{Bernoulli}\,(P)$, where

$$P \propto \beta_{iJ} \prod_{k \in \text{children}(i)} \beta_{k\pi(k)}^{\mathbb{1}(X_{dk}=1)} \left(1 - \beta_{k\pi(k)}\right)^{\mathbb{1}(X_{dk}=0)}$$

and where $\beta_{iJ} = \frac{N_{iJ} + \alpha_0}{N_J + \alpha_1}$
26: Constrain X_{di} in the following cases:

- Set $X_{di} = 1$ if $\exists_{k \in \text{children}(i)}\ X_{dk} = 1 \wedge \forall_{l \in \pi(k)}\ X_{dl} = 0$
- Set $X_{di} = 0$ if $\neg\exists_{k \in \text{children}(i)}\ X_{dk} = 1$

▷ Add new assignment of X_{di} to N_{iJ}
27: **if** $X_{di} = 1$ **then**
28: $N_{iJ} = N_{iJ} + 1$
29: **end if**
30: $N_J = N_J + 1$
31: **end for**
32: **end for**
33: If not converged, goto 13

From the Gibbs Sampler's samples (or from the posterior distribution,) we further compute the marginal probabilities of every category node in the $G\left(V; E\right)$ as

$$p\left(X_i\right) = \frac{\sum_J N_{iJ}}{\sum_{i,J} N_{iJ}} = \frac{\text{Number of documents with } X_i = 1}{\text{Total number of documents}} \tag{12}$$

This probability estimate reflects the importance of a category node and will be used to determine the appropriate sub DAG using an algorithm outlined in section 5. Before

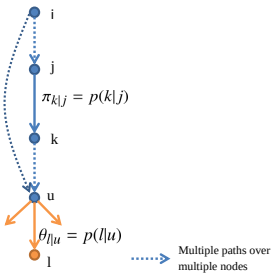

Figure 2: Illustration of path and edge parameters

presenting the sub DAG selection algorithm, we present another approach based on EM to estimate the importance scores of the category nodes in the next Section.

4. EM BASED PARAMETER ESTIMATION

Unlike the previous approach where we treated the DAG as a BN, in this approach we focus on the paths in the DAG from the root to the leaves. We assume that the probability of generating a document at the leaf node is proportional to the product of edge probabilities on the path. Given the category DAG with documents having importance scores at the leaf level, our goal is to estimate the edge probabilities. The leaf nodes' (documents') importance scores are observed. The probabilities of paths (categories) leading to the leaf nodes from the root are hidden. We want to estimate the probability of a path given the observations. In the Figure 2 we illustrate edge parameters which are the conditional probabilities. Given an edge going from node j to k, the probability of the edge is the probability of reaching node k given the node j. Similarly, the probability of reaching a leaf document node l given the category node u is probability of edge going from u to l. We segregate these two probabilities in order to model the category-category and document-category assignments. We estimate these probabilities (parameters of the model) by applying the EM algorithm. Before describing E and M steps, we list the notations that we use:

$\mathcal{P}$: A path from the root to a category having only documents. Note, a path consists of only category nodes.

l : A leaf node, which is a document.

(j,k) : An edge between the categories j and k. The category j is the parent of category k.

$p(\mathcal{P}|l)$: Probability of a path $\mathcal{P}$ given a leaf node l

$\pi_{k|j} = p(k|j)$: Probability of edge (j,k). In other words, probability of reaching child k from the parent j.

$\Theta_{l|k} = p(l|k)$: Probability of generating document l from a given category k.

$\Theta_{l|\mathcal{P}} = p(l|\mathcal{P}) = p(l|\text{end}(\mathcal{P}))$: Probability of generating document l for a given path $\mathcal{P}$. It is same as the probability of generating l from the last category 'end($\mathcal{P}$)' of path $\mathcal{P}$.

N_l : The importance score of the leaf (document) node l.

4.1 E-Step:

In E step, we estimate $p(\mathcal{P}|l)$, the probability of path $\mathcal{P}$ to a given leaf node (document) l. Using Bayes rule, this can be written as:

$$p(\mathcal{P}|l) = \frac{p(l|\mathcal{P})\,p(\mathcal{P})}{\sum_{\mathcal{P}'} p(l|\mathcal{P}')\,p(\mathcal{P}')} \tag{13}$$

The probability of a path can be decomposed as the product of probabilities of edges on the path. Applying this to the above equation gives us the following quantity.

$$p(\mathcal{P}|l) = \frac{\Theta_{l|\mathcal{P}} \prod_{i=1}^{m(\mathcal{P})} \pi_{v(i)|v(i-1)}}{\sum_{\mathcal{P}'} \Theta_{l|\mathcal{P}'} \prod_{i=1}^{m(\mathcal{P}')} \pi_{v(i)|v(i-1)}} \tag{14}$$

While estimating the E step (that is, $p(\mathcal{P}|l)$), the edge probabilities $\pi_{k|j}$ and $\Theta_{l|\mathcal{P}}$ are held at the values estimated from M step.

4.2 M-Step:

In M step, we estimate the edge probabilities $\pi_{k|j}$ and $\Theta_{l|\mathcal{P}}$ by holding the path probabilities $p(\mathcal{P}|l)$ at the values estimated from E step. That is,

$$\pi_{k|j} = \frac{\sum_l N_l \sum_{\mathcal{P}:(j,k)\in\mathcal{P}} p(\mathcal{P}|l)}{\sum_l N_l \sum_{\mathcal{P}:j\in\mathcal{P}} p(\mathcal{P}|l)} \tag{15}$$

$$\Theta_{l|\mathcal{P}} = \frac{N_l \sum_{\mathcal{P}':\text{end}(\mathcal{P}')=\text{end}(\mathcal{P})} p(\mathcal{P}'|l)}{\sum_{l'} N_{l'} \sum_{\mathcal{P}':\text{end}(\mathcal{P}')=\text{end}(\mathcal{P})} p(\mathcal{P}'|l')} \tag{16}$$

The E and M steps are repeated until the convergence, at which the model parameters would have been estimated.

4.3 Downward-Upward Algorithm

Estimating model parameters $\pi_{k|j}$ and $\Theta_{l|\mathcal{P}}$ in EM algorithm (in equations 14,15,16) requires enumerating all the paths in the DAG. In a moderately big DAG, the number of paths can be prohibitively large to process on a computer. For for example, the category DAG in Wikipedia covering the documents in 'Ambient' disambiguation page has about 20 billion paths. Allocating memory to store all the path probabilities and iterating through all the paths for computing various quantities on a moderate capacity computer (16 core, 32 GB RAM) proves to be impossible. This leads us to discover a 'Downward-Upward' algorithm (inspired by Inside-Outside algorithm [4]) to efficiently compute the edge probabilities $\pi_{k|j}$ and $\Theta_{l|\mathcal{P}}$, without enumerating all the paths. At each EM step, the 'Downward-Upward' algorithm first traverses the DAG from root to leaves and then from leaves to the root, by propagating aggregated quantities downwards and then upwards.

By substituting Equations 15 and 16 into Equation 14 we get

$$p^{(t)}(k|j) = \frac{\sum_l N_l \frac{1}{Z_l} \sum_{\mathcal{P}:(j,k)\in\mathcal{P}} \left(p^{(t-1)}(l|\text{end}(\mathcal{P})) \prod_{(r,s)\in\mathcal{P}} p^{(t-1)}(s|r) \right)}{\sum_l N_l \frac{1}{Z_l} \sum_{\mathcal{P}:j\in\mathcal{P}} \left(p^{(t-1)}(l|\text{end}(\mathcal{P})) \prod_{(r,s)\in\mathcal{P}} p^{(t-1)}(s|r) \right)} \tag{17}$$

where $z_l = \sum_{\mathcal{P}'} p^{(t-1)}(l|\text{end}(\mathcal{P}')) \prod_{(r,s)\in\mathcal{P}'} p^{(t-1)}(s|r)$ and t is the iteration index.

We define two functions $\alpha(j)$ and $\beta(j)$ as follows:

$\alpha(j)$ is the total probability of reaching node j from the root node through all the paths from root to j. It can be recursively defined as

$$\alpha(j) = \begin{cases} \sum_{a \in \text{parents}(j)} p(j|a)\,\alpha(a) & \text{if } j \text{ is not the root node} \\ 1 & \text{if } j \text{ is the root node} \end{cases} \tag{18}$$

This can be calculated efficiently by making one pass from the root to the leaves, accumulating the values at each node.

$\beta(j)$ is the fraction of value flowing to node j from the leaf node l through all the paths between j and l. It can be recursively defined as

$$\beta(j) = \begin{cases} \sum_{b \in \text{children}(j)} p(b|j)\,\beta(j) & \text{if } j \text{ is not a leaf node} \\ \frac{N_j}{\alpha(j)} & \text{if } j \text{ is a leaf node} \end{cases} \tag{19}$$

This can be calculated efficiently by making one pass from the leaves to the root, accumulating the values at each node.

The equation 17 can be rewritten in terms of α and β as follows:

$$\begin{aligned} p^{(t)}(k|j) =& \frac{\alpha(j)\,p^{(t-1)}(k|j)\,\beta(k)}{\alpha(j)\,\beta(j)} \\ =& p^{(t-1)}\frac{\beta(k)}{\beta(j)} \end{aligned} \tag{20}$$

The proof is omitted due to space constraints.

Note that, $\pi_{k|j} = \lim_{t\to\infty} p^{(t)}(k|j)$ and $\Theta_{l|\mathcal{P}} = \lim_{t\to\infty} p^{(t)}(l|\text{end}(\mathcal{P}))$

Algorithm2 outlines the Downward-Upward algorithm.

Algorithm 2 Downward-Upward Algorithm

1: Initialize edge probabilities uniformly
2: **while** not converged **do**
3: Downward Propagation: Compute α for each leaf node as in Equation 18
4: Upward Propagation: Compute β for each node as in Equation 19
5: Update edge probabilities as in Equation 20
6: **end while**

From the estimated edge probabilities we further compute the marginal probabilities of every category node X_i in the $G(V;E)$ as

$$p(X_i) = \alpha(i) \tag{21}$$

This probability score reflects the importance of a category node. Combining this score with the entropy of the children, we rank and chose top K category nodes as explained in next section.

5. CONSTRUCTING SUB-DAGS

Once the marginal probabilities $\{p(X_i)\}_{i=1}^{|V|}$ of category nodes $\{X_i\}_{i=1}^{|V|}$ of the DAG $G(V;E)$ have been estimated via the Gibbs Sampling (Equation 12) or EM (Equation 21) algorithms, we determine the importance of a category node X_i through its marginal probability and computing how important its children are. If a node has high marginal probability and all its children too have high marginal probabilities, then we score such a node high. Naturally, entropy of childrens' marginal probabilities is an indication of how informative are the children of a node. Formally we define the entropy of a node X_i's children as follows:

$$H(X_i) = \sum_{k \in \text{children}(X_i)} \bar{p}(X_k) \log\left(\bar{p}(X_k)\right)$$

$\bar{p}(X_k)$ is the normalized marginal probabilities of children (X_i), i.e., $\bar{p}(X_k) = \frac{p(X_k)}{\sum_{j \in \text{children}(X_i)} p(X_j)}$

The rank of a node X_i is defined as follows:

$$r(X_i) = p(X_i) \times H(X_i)$$

Given a budget K, we choose top K ranked nodes and create edges between the nodes X_i and X_j if X_i is ancestor of X_j in DAG $G(V;E)$. This produces a sub-DAG of K nodes that is compact and representative of the of the document collection. In the experimentation section we give a heuristic to estimate the value of K from the training set.

6. EXPERIMENTAL RESULTS

To validate our approach, we make use of the Wikipedia category structure as a category DAG and apply our technique to the task of automatic generation of Wikipedia disambiguation pages. We pre-processed the category graph to eliminate the cycles in order to make it a DAG. Each Wikipedia disambiguation page is manually created by Wikipedia editors by grouping a collection of Wikipedia articles into several groups. Each group is then assigned a name, which serves as a category for the group. Typically, a disambiguation page divides around 20-30 articles into 5-6 groups. Our goal is to measure how accurately we can recreate the groups for a disambiguation page and label them, given only the collection of articles mentioned in that disambiguation page (when actual groupings and labels are hidden.)

6.1 Datasets

We parsed the contents of Wikipedia disambiguation pages and extracted disambiguation page names, article groups and group names. For each article, we extracted click count information from Wikipeida's click count logs. We collected about 800 disambiguation pages that had at least four groups on them. Wikipedia category structure is used as the category DAG. We eliminated few administrative categories such as "Hidden Categories", "Articles needing cleanup" , and the like. The final DAG had about 1M categories and 3M links.

Using disambiguation page title as a keyword query, we retrieved all the Wikipedia articles having those keywords in their title. For each of these articles, we also extracted click counts from the click-logs published by Wikimedia. We eliminated about 30% of the articles having low click counts. Remaining articles are then added to the disambiguation page. Note that, for these added articles, we do not know the actual group in the disambiguation page they belong to. Hence, we do not use these articles while computing the metrics (described in next section.) We only use the articles that are grouped under a disambiguation page by the human editors for metric computation. However, adding queried articles are important because: (i) more data makes our Gibbs sampling and EM based algorithms produce better results, and (ii) in practice (during inference time) we

are only given a disambiguation keyword and our task is to generate the disambiguation page for it. We then have to query the Wikipedia articles using the the disambiguation keyword.

6.2 Evaluation Metrics

While the Wikipedia disambiguation page dataset provides us a large collection of human labeled data, it poses two challenges for the evaluation of our methods; (i) Every group of articles on the Wikipedia disambiguation page is assigned a name by the editors. Unfortunately, these names may not correspond exactly to the Wikipedia category names. For example, one of the groups on the "Matrix" disambiguation page has a name "Business and government" and there is no Wikipedia category by that name. However, the group names generated by the automated methods are those of the Wikipedia categories. (ii) While the Wikipedia disambiguation page group names (in most of the cases) form a single level hierarchy, our methods create a DAG structured hierarchical group names. Hence we cannot evaluate the accuracy of the automated methods just by matching the generated group names to those on the disambiguation page. To alleviate this problem, we adopt cluster-based evaluation metrics. We treat every category node of the sub DAG generated by our algorithm as a cluster of articles. All articles in this cluster are reachable from a path originating form the category node. These are considered as inferred clusters for a disambiguation page. We compare them against the actual grouping of articles on the Wikipedia disambiguation page by treating those groups as true clusters. We can now adopt hierarchical cluster evaluation metrics - FScore measure and Entropy - from [43]. For each disambiguation page in the dataset, we compute every metric score and then average it over all the disambiguation pages.

6.2.1 FScore metric

FScore measure identifies for each class of documents the node in the hierarchical DAG that best represents it and then measures the overall quality of the DAG by evaluating this subset of clusters. Note that in our setup the class of a document is the group name under which the document is listed on the Wikipedia disambiguation page. In determining how well a cluster represents a particular class, the FScore measure treats each cluster as if it was the result of a query for which all the documents of the class were the desired set of relevant documents. Given such a view, then the suitability of the cluster to the class is measured using the F_1 value that combines the standard precision and recall functions used in information retrieval. Specifically, given a particular class L_r of size n_r and a particular cluster S_i of size n_i , suppose n_r^i documents in the cluster S_i belong to L_r, then the F_1 value of this class and cluster is defined to be

$$F_1(L_r, S_i) = 2{*}R(L_r, S_i){*}P(L_r, S_i)/\left(R(L_r, S_i) + P(L_r, S_i)\right)$$

where $R(L_r, S_i) = n_r^i/n_r$ is the recall value and $P(L_r, S_i) = n_r^i/n_i$ is the precision value defined for the class L_r and the cluster S_i . The FScore of class L_r is the maximum F value attained at any node in the DAG G . That is,

$$\text{FScore}(L_r) = \max_{S_i \in G} \; F(L_r, S_i) \tag{22}$$

The FScore of the entire DAG is defined to be the sum of the individual class specific FScores weighted according to the class size. That is,

$$\text{FScore} = \sum_{r=1}^{c} \frac{n_r}{n} \text{FScore}(L_r)$$

where c is the total number of classes. A perfect clustering solution will be the one in which every class has a corresponding cluster containing the same set of documents in the resulting hierarchical DAG, in which case the FScore will be one. In general, the higher the FScore values, the better the clustering solution is.

Note that, this metric does not consider the DAG size or structure while computing FScore. Hence it is trivially possible to maximize FScore in our algorithm just by outputting entire DAG-structured hierarchy, instead of finding optimal sub-DAG. To overcome this, we also evaluate on another metric -an Entropy based measure.

6.2.2 Entropy metric

Unlike the FScore, which evaluates the overall quality of a hierarchical tree using only a small subset of its nodes the socalled, Entropy measure takes into account the distribution of the documents in all the nodes of the tree. Given a particular node S_r of size n_r, the entropy of this node is defined to be

$$E(S_r) = -\frac{1}{\log q} \sum_{i=1}^{q} \frac{n_r^i}{n_r} \log \frac{n_r^i}{n_r}$$

where q is the number of classes in the dataset and n_r^i is the number of documents of the i^{th} class that were assigned to the r^{th} node. Then, the entropy of the entire DAG G is defined to be

$$E(G) = \sum_{r=1}^{p} \frac{1}{p} E(S_r) \tag{23}$$

where p is the number of non-leaf nodes of the DAG G. In general, the lower the entropy values the better the clustering solution is.

6.3 Methods Compared

Closest to our technique is message passing technique [15, 41] where each node passes scores to its parent nodes by equally dividing the message among its parents. Equal division is one of the schemes where every node thinks all its parents are equally likely (akin to uniform prior), which is acceptable in the absence of any other information to differently weight the parents. Note that in our work we do not look into the text of documents/category nodes, hence we do not have any other information on the likelihood of a parent. The scores originate from the leaf (document) nodes and propagate upwards towards the root. The leaf nodes are initialized with the message values equal to their importance scores. After the message passing algorithm stabilizes, the marginal probability of a node is proportional to the total number of messages passed through that node. Since the messages originate from the document nodes and are initialized to the importance scores of the documents, the marginal probabilities of the nodes reflect the importance scores of the documents. We then apply our ranking method to rank and build a sub-graph as explained in Section 5. We call this technique as "Equal-Weighting."

We call our method that we described in Section 3, which is based on treating the DAG structured hierarchy as a

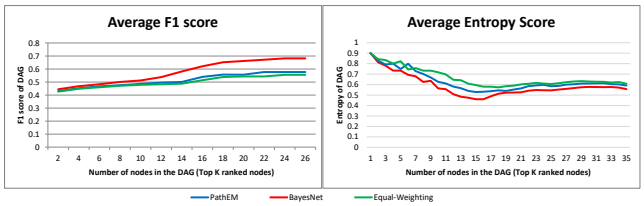

Figure 3: Average F1 and Entropy scores vs DAG size (higher the F1 better; lower the Entropy better)

Bayesian Network and estimating its parameters via Gibbs sampling as "BayesNet". And, the method described is Section 4, which is based on sampling paths and estimating parameters based on EM as "PathEM"

6.4 Evaluation Results

Our first evaluation is based on the F1 metric. Figure 3 first half shows the F1 metric for various values of top K ranked nodes, i.e., sub-DAG of size K nodes. This is the average F1 score from 800 disambiguation pages. In Figure 4 we show the F1 scores for the DAGs of 4 disambiguation pages. By construction, the F1 score monotonically increases with the size of the DAG. Hence, more the nodes we add to the DAG, better the F1 score becomes. Therefore, we should not compare the maximum F1 scores of different algorithms, which obviously will all be equal (a DAG of maximum size.) The F1 scores initially increase rapidly due to the clustering formation that takes place with the addition of each node to the DAG, improving the F1 score. After a certain size (around $K = 15$) the addition of new nodes do not change the clusters and F1 measures that rapidly due to the max F1 in Equation 22 has already found a good node that clusters the documents close to the true clusters. Hence F1 scores start plateauing. Therefore, we recommend to use this heuristic to decide the value of K. Thus, we compare the techniques for the range $K = 15$ to 20.

The BayesNet method performs better than PathEM, which in turn performs better than Equal-Weighting. Since the Gibbs sampling employed in BayesNet is able to come out of local minimum due to the sampling nature, it performs better than PathEM, which often gets stuck into local minimum. Equal-Weighting performs poorly due to the assumption it makes in dividing the messages equally among the parents. This assumption does not seem to perform well because, often a particular parent is more important than other parents. For e.g., the category "Sports" is more relevant parent to the category "Soccer" than the parent category "21st Century Players."

Next we evaluate techniques using the Entropy based measure. The right part of the Figure 3 shows the Entropy metric for various values of top K nodes, i.e., sub-DAG of size K nodes. This is the average Entropy score from 800 disambiguation pages. In Figure 5 we show the Entropy scores for the DAGs of 4 disambiguation pages. Entropy score initially decreases (smaller the Entropy score, better) up to K around 15 and then starts to increase. Due to the improvement in clustering formation with the addition of each node, initially the Entropy score starts to decline. However, when more nodes are added (i.e., as K increases beyond 20) the later nodes do not induce as good clusters as initial nodes, making the overall Entropy score in Equation 23 to increase. However, with large K (greater than 40 or 50) we observe decrease in Entropy once again due to the the addition of "fine-grained" (close to the leaf) categories, which are often associated with single document, resulting in very low Entropy, thus reducing the overall Entropy in Equation 23. Therefore, we recommend to compare different techniques around the first minima that happens around $K = 15$. At this K value, we see that BayesNet performs better than PathEM, which performs better than Equal-Weighting, the reason being the same as explained earlier in F1 measure case.

Figure 6 shows an output (sub-DAG) of BayesNet algorithm run on the "Ambient" disambiguation page.

7. CONCLUSION

We investigated a problem of generating a sub-DAG of categories over a massive DAG-structured category hierarchy such that the sub-DAG produced represents the importance scores of the documents. This representation is characterized through a generative model that learns its parameters such that, repeated sampling of a path ending in a document from the sub-DAG is able to generate the distribution of observed importance scores. Currently we assume that the number of categories K for the sub-DAG is chosen by manual inspection and is given as an input to our algorithm. It would be an interesting future problem to estimate the value of K automatically in our setting. As future work, we also plan to extend our techniques to consume the document text/word along with the importance scores to generate a hierarchy that is also biased towards the textual contents of the documents.

8. REFERENCES

[1] 20Newsgroups. http://qwone.com/ jason/20newsgroups/.

[2] Ramakrishna B Bairi, Rishabh Iyer, Ganesh Ramakrishnan, and Jeff Bilmes. Summarization of multi-document topic hierarchies using submodular mixtures.

[3] Ramakrishna B Bairi, Ganesh Ramakrishnan, and Vikas Sindhwani. Personalized classifiers: evolving a classifier from a large reference knowledge graph. In *IDEAS*, 2014.

[4] James K Baker. Trainable grammars for speech recognition. 1979.

[5] Zafer Barutcuoglu, Robert E. Schapire, and Olga G. Troyanskaya. Hierarchical multi-label prediction of gene function. 2006.

[6] W. Bi and J. T. Kwok. Multi-label classification on tree-and DAG-structured hierarchies. ICML, 2011.

[7] David M. Blei, Thomas L. Griffiths, Michael I. Jordan, and Joshua B. Tenenbaum. Hierarchical topic models and the nested chinese restaurant process. NIPS, 2004.

[8] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *JMLR*, 2003.

[9] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. SIGMOD, 2008.

[10] Razvan Bunescu and Marius Pasca. Using encyclopedic knowledge for named entity disambiguation. ACL, 2006.

[11] Ofer Dekel, Joseph Keshet, and Yoram Singer. Large margin hierarchical classification. ICML, 2004.

[12] Stephen Dill, Nadav Eiron, David Gibson, Daniel Gruhl, R. Guha, Anant Jhingran, Tapas Kanungo, Sridhar Rajagopalan, Andrew Tomkins, John A. Tomlin, and Jason Y. Zien. Semtag and seeker: Bootstrapping the semantic web via automated semantic annotation. WWW, 2003.

[13] Susan Dumais and Hao Chen. Hierarchical classification of web content. IR, 2000.

[14] Paolo Ferragina and Ugo Scaiella. Tagme: On-the-fly annotation of short text fragments (by wikipedia entities). CIKM '10, 2010.

[15] Brendan J Frey and Delbert Dueck. Clustering by passing messages between data points. *science*, 2007.

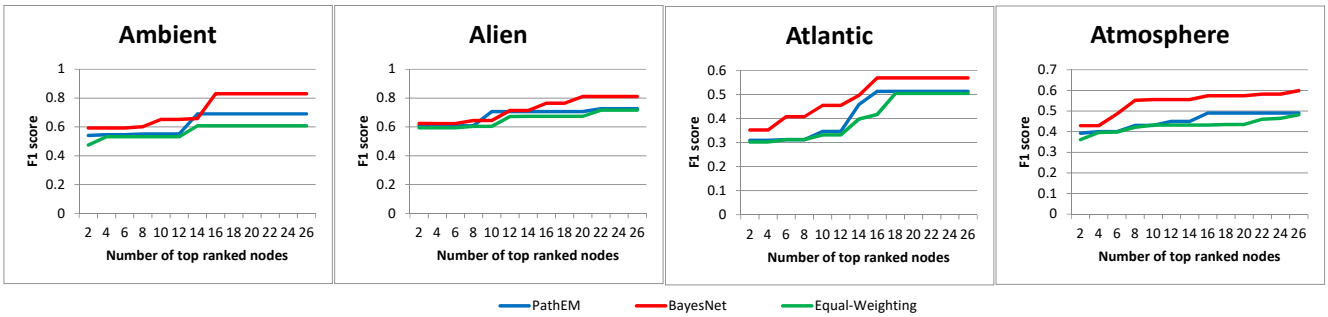

Figure 4: F1 scores of DAGs of 4 Disambiguation pages at various DAG sizes (higher score is better)

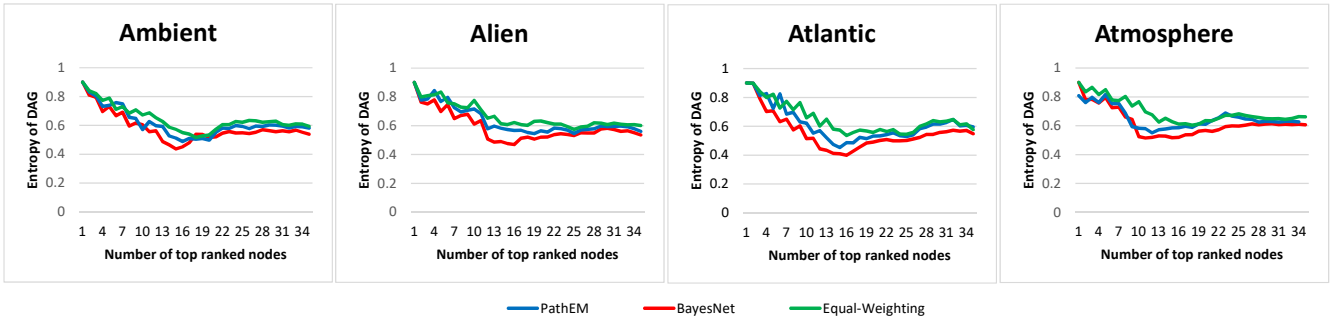

Figure 5: Entropy scores of DAGs of 4 Disambiguation pages at various DAG sizes (lower score is better)

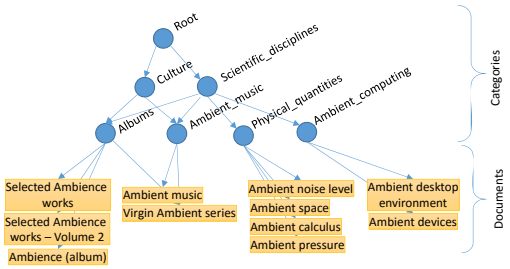

Figure 6: Sub-DAG (of Wikipedia categories) generated for "Ambient" disambiguation page by BayesNet algorithm.

[16] Allan D Gordon. A review of hierarchical classification. *Journal of the Royal Statistical Society. Series A (General)*, 1987.

[17] DMBTL Griffiths and MIJJB Tenenbaum. Hierarchical topic models and the nested chinese restaurant process. *NIPS*, 2004.

[18] Ioana Hulpus, Conor Hayes, Marcel Karnstedt, and Derek Greene. Unsupervised graph-based topic labelling using dbpedia. WSDM, 2013.

[19] Saurabh S Kataria, Krishnan S Kumar, Rajeev R Rastogi, Prithviraj Sen, and Srinivasan H Sengamedu. Entity disambiguation with hierarchical topic models. KDD, 2011.

[20] Wei Li and Andrew McCallum. Pachinko allocation: Dag-structured mixture models of topic correlations. ICML, 2006.

[21] Min ling Zhang and Zhi hua Zhou. Ml-knn: A lazy learning approach to multi-label learning. 2007.

[22] Arun S. Maiya, John P. Thompson, Francisco Loaiza-Lemos, and Robert M. Rolfe. Exploratory analysis of highly heterogeneous document collections. KDD, 2013.

[23] Olena Medelyan, Ian H. Witten, and David Milne. Topic indexing with Wikipedia. AAAI, 2008.

[24] Qiaozhu Mei, Xuehua Shen, and ChengXiang Zhai. Automatic labeling of multinomial topic models. KDD, 2007.

[25] Rada Mihalcea and Andras Csomai. Wikify!: Linking documents to encyclopedic knowledge. CIKM, 2007.

[26] David Milne. An open-source toolkit for mining wikipedia. In *In Proc. New Zealand Computer Science Research Student Conf*, 2009.

[27] Adler J. Perotte, Frank Wood, Noemie Elhadad, and Nicholas Bartlett. Hierarchically supervised latent dirichlet allocation. NIPS, 2011.

[28] Daniel Ramage, David Hall, Ramesh Nallapati, and Christopher D Manning. Labeled lda: A supervised topic model for credit attribution in multi-labeled corpora. EMNLP, 2009.

[29] Reuters-21578. http://www.daviddlewis.com/resources/testcollections/reuters21578/.

[30] Juho Rousu, Craig Saunders, SÃ¡ndor SzedmÃ¡k, and John Shawe-Taylor. Kernel-based learning of hierarchical multilabel classification models. 2006.

[31] Peter Schönhofen. Identifying document topics using the wikipedia category network. 2009.

[32] Jr. Silla, CarlosN. and AlexA. Freitas. A survey of hierarchical classification across different application domains. 2011.

[33] Yang Song, Baojun Qiu, and Umer Farooq. Hierarchical tag visualization and application for tag recommendations. KM, 2011.

[34] Mark Steyvers and Tom Griffiths. Probabilistic topic models. 2007.

[35] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A core of semantic knowledge. WWW, 2007.

[36] Aixin Sun and Ee-Peng Lim. Hierarchical text classification and evaluation. DM, 2001.

[37] Yee Whye Teh, Michael I. Jordan, Matthew J. Beal, and David M. Blei. Hierarchical dirichlet processes. 2006.

[38] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Mining multi-label data. 2010.

[39] Pu Wang, Jian Hu, Hua-Jun Zeng, and Zheng Chen. Using wikipedia knowledge to improve text classification. *Knowledge and Information Systems*, 2009.

[40] Xing Wei and W Bruce Croft. Lda-based document models for ad-hoc retrieval. In *IR*, 2006.

[41] Jonathan S Yedidia, William T Freeman, and Yair Weiss. Understanding belief propagation and its generalizations. 2003.

[42] Torsten Zesch and Iryna Gurevych. Analysis of the wikipedia category graph for nlp applications. NAACL-HL, 2007.

[43] Ying Zhao, George Karypis, and Usama Fayyad. Hierarchical clustering algorithms for document datasets. 2005.