Towards Ontology Learning from Folksonomies *

Jie Tang[⋆] **Ho-fung Leung**[†] **Qiong Luo**[‡] **Dewei Chen**[⋆] **Jibin Gong**[⋆]
[⋆] Department of Computer Science and Technology, Tsinghua University
jietang@tsinghua.edu.cn, chendw@keg.cs.tsinghua.edu.cn
[†] Department of Computer Science and Engineering, The Chinese University of Hong Kong
[‡] Department of Computer Science, Hong Kong University of Science and Technology

Abstract

A folksonomy refers to a collection of user-defined tags with which users describe contents published on the Web. With the flourish of Web 2.0, folksonomies have become an important mean to develop the Semantic Web. Because tags in folksonomies are authored freely, there is a need to understand the structure and semantics of these tags in various applications. In this paper, we propose a learning approach to create an ontology that captures the hierarchical semantic structure of folksonomies. Our experimental results on two different genres of real world data sets show that our method can effectively learn the ontology structure from the folksonomies.

1 Introduction

The Semantic Web aims to provide a Web environment in which each Web page or document is annotated with machine-readable metadata to describe its content. Such metadata can improve the precision of Web search. However, in reality people seldom annotate the Web pages using a consistent ontology such as the Dublin Core ontology.

On the other hand, folksonomy, or user-generated "ontology", has become popular on the Web with the flourish of social networking applications such as social bookmark and blogs. The large amount of user-generated folksonomies provide a promising way to develop the Semantic Web; however it also poses a big challenge in reliability and consistency due to the lack of terminological control. In particular, as the tags are usually freely authored by users, synonymy (multiple tags expressing the same concept), homonymy (a single tag used with different meanings), and polysemy (a single tag used with multiple related meanings) are common, which must be considered in effective content indexing and search.

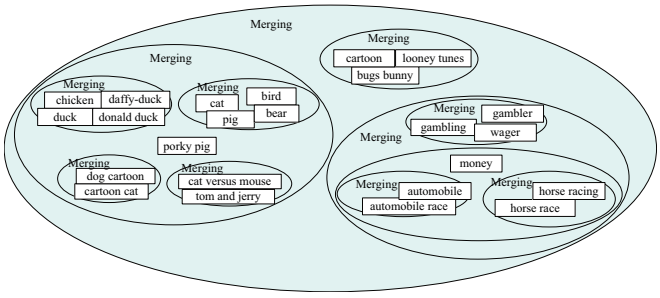
*The work is supported by the Natural Science Foundation of China (60703059), Chinese National Key Foundation Research (2007CB310803), National High-tech R&D Program (2009AA01Z138), Chinese Young Faculty Research Fund (20070003093), and a research award from Google, Inc. The work was also partially supported by a CUHK Research Committee Direct Grant for Research.

Therefore, we explore whether we can (semi-) automatically learn an ontology with hierarchical classifications from folksonomies. The learned ontology is a compromise between a formal ontology, which people seldom use, and the freely created folksonomies [Gruber, 2007]. Such a learned ontology has many immediate applications, for example, collaborative tagging, tag aided search, and tag recommendation.

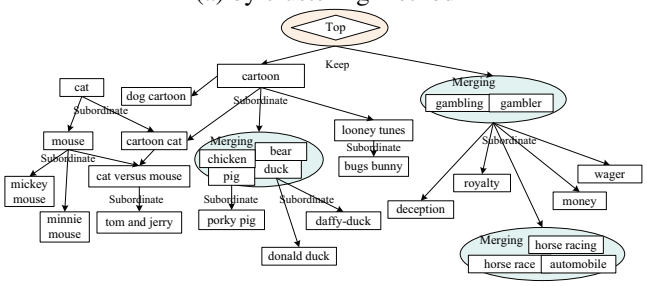
There has been a considerable amount of prior research on ontology learning and extraction. However, ontology learning from folksonomies poses unique technical challenges due to the following reasons:

1. Most of the existing methods focus on ontology learning from text of well-defined terms. However, tags in folksonomies are authored freely by individual users, which may or may not appear in a standard dictionary.
2. Prior methods usually treat the tagging space as a flat schema and do not consider the hierarchical relations between tags. However, tags may describe a resource at different levels of abstraction [Golder and Huberman, 2006].
3. Recently, although a few algorithms have been proposed to learn the hierarchical structures from folksonomies, they mainly focus on clustering similar tags together, but ignore the other relations between tags, e.g., hypernyms and associative relations.

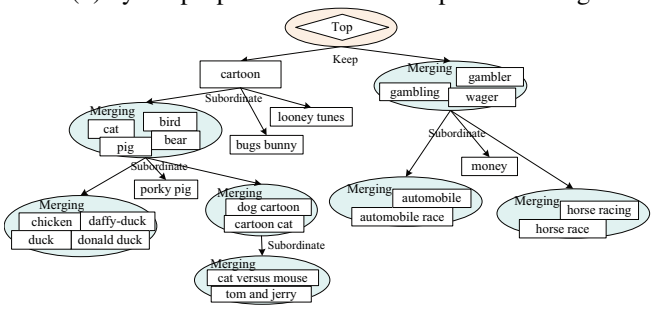
In this paper, we formally define the problem of ontology learning from folksonomies and propose a three-stage approach to solving the problem. Specifically, we present a generative probabilistic model to model tags and their annotated documents. We define four divergence measures to quantitatively differentiate the relations between tags based on the modeling result. We further define an objective function for the hierarchical structure construction and propose an efficient algorithm to realize the objective function. Experimental results on two different genres of real world data sets show that our method can effectively learn the ontology from the folksonomies. Figure 1 is an example comparison of the learned ontologies by our proposed method and the traditional clustering-based method. We see the clustering based method iteratively clusters similar tags together, but cannot identify the other relations (e.g., hypernym) between tags.



(a) by clustering method



(b) by the proposed method w/o topic smoothing



(c) by the proposed method with topic smoothing

Figure 1: Example comparison of learned ontologies from MOVIE data.

Our proposed method can not only find the synonym relation, but also discover the hypernym and some other relations between tags.

Our proposed approach is quite general, as it requires no prior knowledge about a specific domain, and can learn an ontology hierarchy from any social tagging applications. This generality makes our approach applicable to many different domains.

2 Problem Formalization

We first define notations used in this paper. Assuming that a tag $t_i, i \in [1, T]$ can be used to annotate multiple documents, while each document can be also labeled with multiple tags. Further consider a document d contains a vector w_d of N_d words, in which each w_{di} is chosen from a vocabulary of size V. Then a set of tags with the annotated documents can be represented as $\mathbf{D} = \{(t_1, \{\mathbf{w}_{11}, \ldots, \mathbf{w}_{1D_1}\}), \ldots, (t_T, \{\mathbf{w}_{T1}, \ldots, \mathbf{w}_{1D_T}\})\}$, equivalently $\mathbf{D} = \{(\mathbf{w}_1, \mathbf{t}_1), \ldots, (\mathbf{w}_D, \mathbf{t}_D)\}$, where $\mathbf{t}_{d_i}$ denotes a set of tags used to annotate document d_i.

Our objective of ontology learning from folksonomies is to construct a hierarchical ontology based on the tags and their annotated documents. As for the definition of ontology, we adopt it from [Dean *et al.*, 2004], where the main components of an ontology are concepts C, relations R, instances I, and axioms A^O, formally: $O = \{C, R, I, A^O\}$.

For ontology learning from folksonomies, we aim to learn concepts and hierarchical relations between tags. We consider a many-to-one correspondence between tags in the folksonomy and concepts in the ontology. Learning the many-to-many correspondence between tags and concepts is our on-going work and will be reported elsewhere. Finally, the tagged documents are viewed as instances of concepts (tags).

For any two tags, the relation between them might be Hypernym (the meaning of one tag subsumes that of the other tag), Synonym (the meanings of two tags are similar to each other), or Others (two tags are closely related but with different meanings). Correspondingly, we define the possible operations on pairs of tags as: **Subordinate**, **Merge**, and **Keep**. **Subordinate** means that we create a hypernym-hyponym relation between the two tags; **Merge** means that we consider the two tags to be synonyms and merge them into one concept; **Keep** means that we keep the two tags as two different concepts in the ontology. We note that previous cluster-based methods mainly discover the similar relation between tags. The similarity relation actually corresponds to the **Merge** operation defined here. We further introduce the notion of virtual concept (tag). We call the merged tags as a virtual concept. We also call the created concept for two **Kept** tags as a virtual concept.

Here, we only consider tags and their annotated documents. The problem setting can be easily generalized to other resources such as picture and music. Further we can consider the setting with only tag information. The proposed approach is very general, the only thing needed to do for adapting to different settings is to change the modeling approach of the first stage in our approach.

3 Our Approach

At a high level, our approach primarily consists of three stages. In the first stage, we use generative probabilistic models to model correspondences between tags and documents. The basic idea is that assuming each tag has multiple sub-meanings (K topics), a tag having similar high distributions on multiple topics indicates a high likelihood of the tag being a general one; while a tag having a high distribution on only one specific topic indicates that the tag possibly has a specific meaning. In the second stage, we estimate the possible relations between tags. Specifically, we define four divergence measures between tags based on the modeling results from the previous stage. In the third stage, we determine the relation between tags and construct a hierarchical structure.

3.1 Modeling Folksonomy

The purpose of the first stage is to model the tags and their annotated documents in the collection **D**. This can be done in many different ways, for example, a standard vector space modeling approach or the language modeling approach [Salton and McGill, 1986]. However, such approaches cannot

capture the intrinsic relations between documents and tags, e.g., the synonym relation between "knowledge discovery" and "data mining". Our main idea in this work is to use probabilistic topic models to model the documents and the tags in a unified way.

Tag-Topic (TT) Model

Intuitively, a document is "annotated" by following a stochastic process: first, each word w_{di} in the document is associated with a selected topic z_{di}; the tag's author would think about a specific tag t_{di} to represent the topic z_{di} according to $P(\theta|t_{di})$; finally tags $\mathbf{t}_{d_i}$ with the highest posterior probabilities would be used to annotate this document.

In this model, the parameters to be estimated are: (1) the distribution θ of T tag-topics and the distribution ϕ of K topic-words; and (2) the chosen topic z_{di} and tag t_{di} for each word w_{di}. Generally, a variational EM methods [Blei *et al.*, 2003] or a Gibbs sampling algorithm [Griffiths and Steyvers, 2004] is applied to estimate the topic model. We choose Gibbs sampling for its ease of implementation. Instead of estimating the model parameters directly, we evaluate the posterior distribution on just t and z and then use the results to infer θ and ϕ. The posterior probability is defined as:

$$P(z_{di}, t_{di}|\mathbf{z}_{-di}, \mathbf{t}_{-di}, \mathbf{w}, \alpha, \beta) = \frac{P(\mathbf{z}, \mathbf{t}, \mathbf{w}|\alpha, \beta)}{P(\mathbf{z}_{-di}, \mathbf{t}_{-di}, \mathbf{w}|\alpha, \beta)} \propto \frac{m_{tz}^{-di} + \alpha_z}{\sum_z (m_{tz}^{-di} + \alpha_z)} \frac{n_{zv}^{-di} + \beta_v}{\sum_v (n_{zv}^{-di} + \beta_v)} \quad (1)$$

where m_{tz} is the number of times that topic z is associated with the chosen tag t; n_{zv} is the number of times that the word w_v is generated by the topic z; the notation with the superscript $-di$ denote a quantity, excluding the current instance (the i-th word in the document d); $\{\alpha_z\}$ and $\{\beta_v\}$ are respectively topic-specific and word-specific hyperparameters in the topic model.

After Gibbs sampling, we can use the sampled topics for words to estimate the probability of a topic given a tag θ_{tz} and the probability of a word given a topic ϕ_{zv}:

$$\theta_{tz} = \frac{m_{tz} + \alpha_z}{\sum_z (m_{tz} + \alpha_z)} \qquad \phi_{zv} = \frac{n_{zv} + \beta_v}{\sum_v (n_{zv} + \beta_v)} \quad (2)$$

Model Limitations: Intuitively, frequent co-used tags would have a similar meaning, thus have a similar topic distribution. However, the learned topic distributions of the co-tags are not always smooth. This leads us to think about how to smooth the topic distribution between *co-tags*.

Topic Smoothing

The main idea for topic smoothing is to utilize the tag network to regularize (smooth) the topic distribution between tags. In the tag network G, each node represents a tag and each edge represents a co-tag relation. That is if two tags t_i and t_j are used to annotate a same document, we create an edge (t_i, t_j) in the tag network. Thus, we can define the regularized data likelihood of the TT model as:

$$O_\epsilon(D, G) = -\epsilon L(D) + (1 - \epsilon) R(D, G) \quad (3)$$

where $L(D)$ is the (log-)likelihood of the collection D to be generated by the TT model, $R(D, G)$ is a harmonic regularizer defined on the tag network G, and ϵ is a coefficient that controls the relative strength of the two terms. The harmonic regularizer $R(D, G)$ can be further defined as:

$$R(D, G) = \frac{1}{2} \sum_{(t_i, t_j) \in G} w(t_i, t_j) \sum_{k=1}^{K} (\theta_{t_i z_k} - \theta_{t_j z_k})^2 \quad (4)$$

where $w(t_i, t_j)$ denotes the weight of the edge (t_i, t_j). We, for simplicity, define $w(t_i, t_j)$ as $\frac{D_{(t_i, t_j)}}{D_{t_i}}$, where $D_{(t_i, t_j)}$ is the number of documents co-tagged by t_i and t_j, D_{t_i} is the number of documents tagged by t_i.

By minimizing Eq. 3, we attempt not only to find a probabilistic model that best fits the data collection **D**, but also to smooth the topic distribution between tags. For learning all parameters in Eq. 3 together, we use a two-step algorithm. In the first step, we train the model parameters (θ, ϕ) using the objective function $O_1(D, G) = -L(D)$ with the Gibbs sampling algorithm by setting the Dirichlet prior for each tag t_i as $\alpha_{t_i k} = \epsilon\alpha + (1-\epsilon)\frac{K}{|G_{t_i}|} \sum_{(t_i, t_j) \in G} \theta_{t_j k}$, where $|G_t|$ is the number of neighbors of tag t in the tag network G. In the second step, we fix ϕ, and re-estimate the multinomial θ to minimize O_ϵ by running an iterative process to obtain the new θ for each tag t, $\theta_{tik}^{(n+1)} = \gamma\theta_{tik}^{(n)} + (1-\gamma)\frac{\sum_{(t_i, t_j) \in G} w(t_i, t_j)\theta_{tjk}^{(n)}}{\sum_{(t_i, t_j) \in G} w(t_i, t_j)}$, where γ is a coefficient to smooth the topic distribution. The learning algorithm has also been used previously for semi-supervised learning [Zhu and Lafferty, 2005] and for training pLSI model [Mei *et al.*, 2008].

3.2 Divergence Estimation

The purpose of the second stage is to quantitatively characterize the possible relations between tags. Based on the modeling results, we define four divergence measures to characterize the tag relations.

Tag divergence. Tag divergence is the measure of the difference/dissimilarity between two topic distributions of tag t_i and tag t_j. We use the notation $diverg(t_i, t_j)$ to represent the tag divergence from t_i to t_j. It is an asymmetric metric, i.e. $diverg(t_i, t_j) \neq diverg(t_j, t_i)$. Based on the modeling result, we use KL-divergence, a standard measure of the difference between two probability distributions, to define $diverg(t_i, t_j)$ as:

$$diverg(t_i, t_j) = D_{KL}(\theta_{t_i} \| \theta_{t_j}) = \sum_{z=1}^{K} \theta_{t_i z} log \frac{\theta_{t_i z}}{\theta_{t_j z}} \quad (5)$$

Hypernym-divergence. Hypernym-divergence is a measure of the likelihood of a tag t_i's being the hypernym of the other tag t_j, denoted as $hyper - diverg(t_i, t_j)$. High degree of likelihood is indicated by a lower hypernym-divergence.

Still based on the modeling result, we define $hyper - diverg(t_i, t_j)$ as:

$$hyper - diverg(t_i, t_j) = \sum_{z=1}^{K} \frac{P(t_i|z_k) - P(t_j|z_k)}{P(t_i|z_k)} \quad (6)$$

where $P(t|z)$ is obtained by using the Bayesian rule, i.e., $P(t|z) = (P(z|t)P(t))/P(z)$.

The definition is derived from the observation: if tag t_i has higher posterior probability than tag t_j on each topic z_k, then it is very likely that t_i is the hypernym of t_j.

Merging-divergence. Merging-divergence is a measure of the likelihood of two tags t_i and t_j describing the same concept, denoted as $merg-diverg(t_i, t_j)$. This metric is a symmetric measure.

$$merg - diverg(t_i, t_j) = \frac{1}{2}(diverg(t_i, t_j) + diverg(t_j, t_i)) \quad (7)$$

The definition has a straightforward explanation: if two tags are similar to each other, then it is likely that the two tags should be merged together.

Keep-divergence. Keep-divergence is a measure of the likelihood of two tags t_i and t_j describing different concepts. It is denoted as $keep - diverg(t_i, t_j)$. This metric is also a symmetric measure.

$$keep - diverg(t_i, t_j) = \frac{1}{2}(merg - diverg(t_i, t_j) + (max_k(diverg(t_i, t_k) - diverg(t_i, t_k)))) \quad (8)$$

where $max_k(diverg(t_i, t_k))$ is the maximum tag divergence from tag t_i to any other tags.

The definition is inspired by how people create a tag hierarchy: when people create a hypernym for two tags, the two tags must be relevant (we use merging-divergence to quantify it), but are not very similar (otherwise should be merged; we use $max_k(diverg(t_i, t_k)) - diverg(t_i, t_k)$ to quantify it).

3.3 Hierarchical Structure Construction

The purpose of this stage is to construct the hierarchical structure between tags. The challenges here are that, given any two tags, which operation we should assign to them? and how to obtain a global operation assignment for all tags? Our main idea is to define an objective function, which quantifies the loss caused by the operations on a tag set. Then, we can find the operations that can minimize the objective function.

The objective function is defined based on the estimated divergences. Intuitively, in the learned ontology, we want to have a high abstraction level of the tags by merging similar tags or creating the hypernym relation between two tags (if any). Let N be the number of concepts in the learned ontology. Formally, we can minimize the number of learned concepts, i.e. min $log(N)$. On the other hand, by merging or creating hypernym relation between two tags, we inevitably have some loss of information. Theoretically, we need a quantitative measure to estimate the information loss. We use the divergences defined in the previous section to define the loss, e.g. $hyper-diverg(t_i, t_j)$ means the loss when tag t_i is taken as the hypernym of t_j. A large divergence means a high loss. Finally, the objective function is defined as:

$$\sum_{(t_i,t_j)\in HS} hyper - diverg(t_i, t_j) + \sum_{(t_k,t_l)\in MS} merg - diverg(t_k, t_l) + \sum_{(t_r,t_s)\in KS} keep - diverg(t_r, t_s) + \lambda log(N) \quad (9)$$

Table 1: The algorithm of hierarchical structure construction.

1. initialize the tree **O** by taking all tags as the leaf nodes;
2. initialize a concept set **A** with all tags;
3. do{
   (a) find a pair of tags (or virtual tag/concepts) (t_i, t_j) with an operation from A that minimizes the objective function (10);
   (b) execute the operation:
      i. for Subordinate, assign t_i as the hypernym of t_j, remove t_j from **A**, and move ti as the parent node of t_j in **O**;
      ii. for Merge, create a virtual concept c_{ij} by combining t_i and t_j, add it to **A** as well to **O** as a leaf node, remove t_i and t_j from both **A** and **O**, and calculate the divergences of the new concept c_{ij} with all others tags in **A**.
      iii. for Keep, create a virtual concept c_{ij} as the hypernym of both t_i and t_j, add c_{ij} as the parent node of t_i and t_j in **O**, add c_{ij} to **A**, remove t_i and t_j from **A**, and calculate the divergences of the new concept c_{ij} with all others tags in **A**.
4. }until $|\mathbf{A}| \leq 1$; //$|\mathbf{A}|$ is the number of elements in **A**.
5. return the hierarchical tree **O**.

where HS indicates a set of tag pairs on which we use the Subordinate operations; similarly, MS and KS respectively indicates a set of tag pairs on which we use the Merge and Keep operations; λ is a parameter that controls the trade-off between the minimum information loss and the minimum number of concepts in the learned ontology.

To find the exact solution for the objective function is an NP-hard problem. We propose an algorithm to find the approximate solution to this problem. Specifically, we define a local minimization form of the objective function as:

$$argmin_{i\neq j} Operation(t_i, t_j) + \lambda log(N) \quad (10)$$

Here the operation is one of the three types of operations we define. Table 1 summarizes the algorithm. In the algorithm, we introduce a concept set **A** to track the concepts/tags we need to compare. When we find a tag pair with an operation that minimizes Eq. 10, we execute the corresponding operation on the two concepts (tags).

4 Experimental Results

4.1 Data Sets and Experimental Setting

Data sets We conducted experiments on two genres of real-world data: academic papers (PAPER) from http://www.citeulike.com and movie reviews (MOVIE) from http://www.imdb.com. The PAPER data set consists of $D = 4,841$ papers (only titles and abstracts) crawled from the website. There are $T = 8,071$ unique tags and a total of $37,010$ tags assigned to the papers. For the MOVIE data set, we randomly chose $D = 4,009$ movie reviews from the website. The reviews were annotated with $T = 18,559$ unique tags and a total of $142,498$ tags. We preprocessed each data set by (a) removing stopwords and numbers; (b) removing words that appear less than three times in the corpus; and (c) downcasing the obtained words. Finally, we obtained $V = 7,518$ unique words and a total of $408,270$ words in the PAPER data set and $V = 9,777$ unique words and a total of $225,049$ words in the MOVIE data set.

Table 2: Performance of ontologies learning (%).

Data Set	Method	Human Evaluation	ODP Comparison
		accuracy	accuracy
PAPER	Baseline	46.71	34.28
	w/o smoothing	65.43	60.83
	with smoothing	71.24	65.52
MOVIE	Baseline	51.62	47.38
	w/o smoothing	62.46	58.24
	with smoothing	67.89	61.75

Parameter setting In our experiments of topic model (stage 1), the number of topics was empirically set as $T = 80$ and the hyperparameters α and β were set with $\alpha = 50/K$ and $\beta = 0.01$ respectively. In addition, we tentatively set both coefficients ϵ and γ for topic smoothing as 0.5. As for the parameter λ (stage 3), we empirically set it to 1. We ran 5 independent Gibbs sampling chains for 2,000 iterations each. It took three hours on PAPER and 2.5 hours on MOVIE to train the TT model on one machine.

Evaluation Measures and Baseline Methods For quantitatively evaluating the proposed approach, we used two evaluation methods. The first is to compare the generated ontology by our approach with the category structure from Open Directory Project(ODP) (http://www.dmoz.org/). From ODP, we extracted a sub-directory related to our data set. For example, as most papers we crawled from citeulike.com are related to computer science, we extracted the "Computers" subdirectory of ODP. For MOVIE, we extracted the sub-directory "movie" from ODP. We compared the discovered relations with the defined category relations in ODP. For easy evaluation, we only compared the hypernym relation and the sibling relation (two tags have a same super-concept). The other method is to ask ontology authors to directly evaluate the generated ontology. Specifically, we randomly selected snippets (each containing about 100 tags, as the example in Figure 1) from the learned ontology and asked three ontology authors from schemaweb.info if they were satisfied with the discovered relations between tags. After receiving feedbacks from all authors, we combined the evaluation results and counted the satisfactory number S and the unsatisfactory number U. The accuracy is then calculated by $S/(S+U)$.

We defined a baseline method based on the hierarchical clustering. Specifically, we used vector space model to represent each tag and used the cosine similarity to measure the similarity between tags. Then a hierarchical clustering algorithm was applied to iteratively merge the most similar tags (tag clusters) together.

4.2 Quantitative Evaluation

Table 2 shows the performance of ontology learning by our proposed approach and the baseline method. We evaluated the generated ontologies by our approach with and without (w/o) topic smoothing. In Table 2, the third column denotes the accuracy evaluated by the three ontology authors while the fourth column denotes the accuracy by comparing with the ODP directories.

We see from Table 2 that our approach significantly outperforms the baseline method. We can also see that topic smoothing indeed improves the quality of the learned ontologies. Table 3 shows the statistics of the learned ontologies from the two data sets. Columns 4-7 list the number of created virtual concepts, operation numbers of Subordinate, Merge, and Keep respectively.

Table 3: Statistics of the learned ontologies.

Data Set	Method	#Concept	#Operations			
		#Tag	#Virtual concept	#Subordinate	#Merge	#Keep
PAPER	w/o smoothing	3032	2021	4945	1448	549
	with smoothing	2463	1916	5714	2104	597
MOVIE	w/o smoothing	9456	3291	15123	4124	1374
	with smoothing	4590	5744	9208	9369	1246

4.3 Example Analysis

Figure 1 shows the example snippet of the learned ontology from the MOVIE data set with and without using the topic smoothing method. We can see an interesting pattern from the figures. First, with topic smoothing, our method tends to result in merged concepts. For example, without topic smoothing (Figure 1 (b)), "cat versus mouse" and "tom and jerry" are separated as two hyponyms; while with topic smoothing (Figure 1 (c)), they are merged into one concept. This is because with topic smoothing, we can obtain a more similar topic distribution for the frequently co-used tags, which makes them have a lower merging divergence.

Secondly, the learned ontology unveils some hot topics on the internet. For example, in the MOVIE data, we have found that "cat" seems to be the most popular creature in the cartoon movies. Thus it is created as the hypernym of the other creatures. Among the 170 movies tagged with "cartoon" in our data, 47.1% of them are tagged with "cat". The other creatures seem not so popular, e.g., "mouse" has a percentage of 20.0% and "pig" has only 5.9%.

Thirdly, we see the learned hierarchical relations are not necessarily superconcept-subconcept relations. We analyzed the learned ontology and found that the learned relations can be mainly categorized into three types: (1) one tag is superordinated to the other tag, e.g., "cartoon" and "dog cartoon"; (2) one tag is related to the other tag, e.g., "gambling" and "money"; (3) one tag is the a part of the other tag, e.g., "horse race" and "gambling". Statistics further show that in the PAPER data, 24.2% of the relations are the first type, 57.6% of the relations belong to the second, and 18.2% of the relations are of the third type; in the MOVIE data, 58% of the relations belong to the first type, 37% of the relations are the second type, and only 5% of the relations are the third type.

Finally, we need note that there are also some unsatisfactory results in the learned ontology. For example, in Figure 1, "automobile" is merged with "automobile race". It might be more reasonable to create a hypernym-hyponym relation between them. We conducted error analysis on the results and found that the major errors are due to: (1) the algorithm for discovering the structure is still a suboptimal solution; (2) there is much noise in the tag assignments of the documents on the Web. The quality of tags by different users varies largely depending on the users' expertise. This might be alleviated by further incorporating the user authority information into the current method.

5 Related Work

Ontology learning is an important area in the Semantic Web. Many research efforts have been made so far. However, much of the previous work focuses on extracting concepts and relations from text. Recently, several research papers have been conducted to integrate the user created folksonomies with the Semantic Web. However, no previous work has been done for dealing with all the tasks (defined in this paper) for ontology learning from folksonomies, to the best of our knowledge.

Ontology learning (also called ontology extraction) from text aims at extracting ontological concepts and relations from plain text or Web pages. For example, [Maedche and Staab, 2001] propose an ontology learning framework for ontology extraction, merging, and management. They employ data mining approaches (such as hierarchical cluster and association rule) and some background knowledge to learn the concepts, hierarchical relations, and associative relations from text. [Han and Elmasri, 2003] have developed a system called WebOntEx, trying to extract ontology from Web pages based on HTML tags, lemmatization tags, and conceptual tags. [Buitelaar *et al.*, 1999] have implemented a plug-in in Protégé to support ontology extraction from text. [Sleeman *et al.*, 2003] discuss a method to identify ontological knowledge implicit in a knowledge base.

As for folksonomy integration, a few algorithms have been proposed for learning the synonym and hypernym relations between tags, e.g. [Li *et al.*, 2007]. Some other efforts try to generate clusters of highly related tags and associate each cluster to a concept of the existing ontology, for example [Specia and Motta, 2007]. Zhou et al. propose an unsupervised method based on deterministic annealing for exploring the hierarchical relations between tags [Zhou *et al.*, 2007]. However, it does not consider the different types of relations (e.g., Hypernym, Synonymy, and Others) between tags.

6 Conclusion

In this paper, we investigate the problem of ontology learning from folksonomies. We formalize the major problems of ontology learning from folksonomies and propose our solution to the task. We exploit a probabilistic topic model to model the tags and their annotated documents and define four divergence measures to characterize the relations between tags. We propose an algorithm to construct the hierarchical structure between tags. Experimental results on two different types of real-world data sets show that our method can effectively learn the ontological hierarchy from social tags.

There are many potential future directions of this work. It would be interesting to use a global optimization algorithm to solve the Eq. 9. It would also be interesting to investigate how to discover the associative relation between tags.

References

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

[Buitelaar *et al.*, 1999] Paul Buitelaar, Daniel Olejnik, and Michael Sintek. A protege plug-in for ontology extraction from text based on linguistic analysis. In *Proceedings of ESWC'04*, pages 91–96, 1999.

[Dean *et al.*, 2004] Mike Dean, Guus Schreiber, Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. Owl web ontology language reference. w3c recommendation., Feb. 2004.

[Golder and Huberman, 2006] Scott A. Golder and Bernardo A. Huberman. Usage patterns of collaborative tagging systems. *Journal of Information Science*, 32(2):198–208, 2006.

[Griffiths and Steyvers, 2004] Thomas L. Griffiths and Mark Steyvers. Finding scientific topics. In *Proceedings of the National Academy of Sciences*, pages 5228–5235, 2004.

[Gruber, 2007] Thomas Gruber. Ontology of folksonomy: A mash-up of apples and oranges. *International Journal on Semantic Web & Information Systems*, 3(2):1–11, 2007.

[Han and Elmasri, 2003] Hyoil Han and Ramez Elmasri. Ontology extraction and conceptual modeling for web information. pages 174–188, 2003.

[Li *et al.*, 2007] Rui Li, Shenghua Bao, Yong Yu, Ben Fei, and Zhong Su. Towards effective browsing of large scale social annotations. In *Proceedings of WWW'07*, pages 943–952, 2007.

[Maedche and Staab, 2001] Alexander Maedche and Steffen Staab. Ontology learning for the semantic web. *IEEE Intelligent Systems*, 16(2):72–79, 2001.

[Mei *et al.*, 2008] Qiaozhu Mei, Deng Cai, Duo Zhang, and ChengXiang Zhai. Topic modeling with network regularization. In *Proceedings of WWW'08*, pages 101–110, 2008.

[Salton and McGill, 1986] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.

[Sleeman *et al.*, 2003] Derek Sleeman, Stephen Potter, Dave Robertson, and Marco Schorlemmer. Ontology extraction for distributed environments. pages 80–91, 2003.

[Specia and Motta, 2007] Lucia Specia and Enrico Motta. Integrating folksonomies with the semantic web. pages 624–639. 2007.

[Zhou *et al.*, 2007] Mianwei Zhou, Shenghua Bao, Xian Wu, and Yong Yu. An unsupervised model for exploring hierarchical semantics from social annotations. In *Proceedings of ISWC'07*, 2007.

[Zhu and Lafferty, 2005] Xiaojin Zhu and John Lafferty. Harmonic mixtures: Combining mixture models and graph-based methods for inductive and scalable semi-supervised learning. In *Proceedings of ICML'05*, pages 1052–1059, 2005.