Prototype Hierarchy Based Clustering for the Categorization and Navigation of Web Collections

Zhao-Yan Ming[1,2], Kai Wang[2] and Tat-Seng Chua[2]
[1]NUS Graduate School for Integrative Sciences and Engineering
[2]Department of Computer Science, School of Computing
National University of Singapore
{mingzy,kwang,chuats}@comp.nus.edu.sg

ABSTRACT

This paper presents a novel prototype hierarchy based clustering (PHC) framework for the organization of web collections. It solves simultaneously the problem of categorizing web collections and interpreting the clustering results for navigation. By utilizing prototype hierarchies and the underlying topic structures of the collections, PHC is modeled as a multi-criterion optimization problem based on minimizing the hierarchy evolution, maximizing category cohesiveness and inter-hierarchy structural and semantic resemblance. The flexible design of metrics enables PHC to be a general framework for applications in various domains. In the experiments on categorizing 4 collections of distinct domains, PHC achieves 30% improvement in μF_1 over the state-of-the-art techniques. Further experiments provide insights on performance variations with abstract and concrete domains, completeness of the prototype hierarchy, and effects of different combinations of optimization criteria.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*clustering*

General Terms

Algorithms, Performance, Experimentation.

Keywords

Hierarchical Clustering, Prototype Hierarchy, Hierarchy Induction, Criterion Function

1. INTRODUCTION

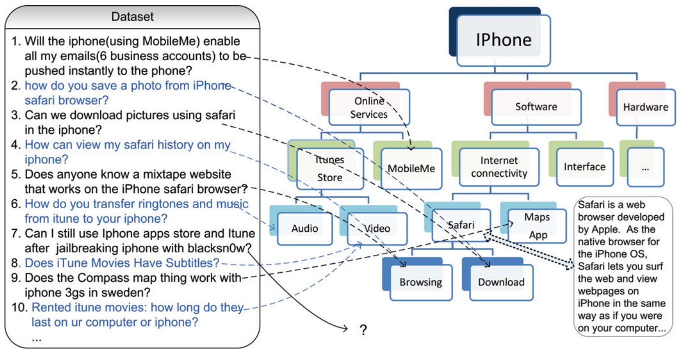
With the flourishing of user contributed services like Yahoo! Answers, discovering the utility of user-generated-contents becomes a research topic of interest to many researchers. The utility of user-generated-contents comes in two major aspects, the quality and accessibility. Efforts have been put to distinguish the good and bad quality content [1]. To make contents more accessible, state-of-the-art retrieval models like translation based language model [18] and syntactic tree matching [15] have achieved promising performance. Organizing the huge collections of data for information navigation is another important direction in exploring web collections. Categorization, especially hierarchical clustering with labels and descriptions of clusters, enables browsing style of information access. Users can navigate through the hierarchy driven by their information needs [11, 17].

Currently, web services rely on users to construct topic hierarchies and assign objects into their nodes. Open Directory Project (ODP) and Wikipedia are both examples of hierarchically organized web collections formed by community of editors. Yahoo! Answers (YA) is organized in a hierarchical tree containing 728 nodes with 26 top-level categories, relying on users to select a category for their postings. Besides the reliance on manual assignment, a hierarchy as large as YA's directory is too coarse to contain a category like *IPod* (it is in *Music & Music players*) whose subtopics might be of interest to many users. These suggest the necessity of automatic fine-grained hierarchical categorization.

Toward automatic categorization of web collections into hierarchies, supervised techniques that require manually-labeled corpora are not appropriate for dynamic Web information services [9]. Existing unsupervised techniques generally focus either on clustering the collections into smaller groups [5, 17], or extracting labels for clustered groups [4]. SnakeT [6] is a successful hierarchical clustering engine that performs sequential clustering and labeling on snippets returned by search engines. However, the resulting clusters and labels may not be consistent and systematic because of its data-driven nature. LiveClassifier [9] addresses the categorization and navigation in one go by utilizing predefined topic hierarchies and searching the training instances to feed into a supervised learner. This approach, however, ignores the underlying topic structure of the target collection; and the result is confined to the predefined hierarchy which may not be a perfect match to the collection.

In this paper, we propose an unsupervised approach called **Prototype Hierarchy based Clustering** (PHC) to tackle the problem of web collection categorization and navigation. PHC utilizes the world knowledge in the form of prototype hierarchies, while adapts to the underlying topic structures of the collections. By following the structure of the prototype hierarchy, PHC eliminates the problem of determining the number of clusters and assigning initial clusters.

Moreover, the PHC results are interpretable, comprehen-


Dataset

1. Will the iphone(using MobileMe) enable all my emails(6 business accounts) to be pushed instantly to the phone?
2. how do you save a photo from iPhone safari browser?
3. Can we download pictures using safari in the iphone?
4. How can view my safari history on my iphone?
5. Does anyone know a mixtape website that works on the iPhone safari browser?
6. How do you transfer ringtones and music from itune to your iphone?
7. Can I still use Iphone apps store and Itune after jailbreaking iphone with blacksn0w?
8. Does iTune Movies Have Subtitles?
9. Does the Compass map thing work with iphone 3gs in sweden?
10. Rented itune movies: how long do they last on ur computer or iphone?

...

Safari is a web browser developed by Apple. As the native browser for the iPhone OS, Safari lets you surf the web and view webpages on iPhone in the same way as if you were on your computer...

Figure 1: Categorizing a Yahoo! Answers dataset using the Prototype Hierarchy of *IPhone*

sive, and organized. Unlike in general clustering schemes where no label is provided for the resulting groups, or in conceptual clustering [12] where extra efforts are needed to extract labels, in PHC the labels or the descriptions of clusters are already provided by the prototype hierarchy used. Therefore the trouble of inventing cluster labels is eliminated. What's more, the labels provided by the supervision hierarchy are logically and systematically arranged, while the labels extracted from unsupervised clustering might be inconsistent and disorganized.

PHC allows flexible forms of supervision: the prototype hierarchy can come in different level of granularity, in different forms, and even tailored to any specific applications. Thanks to the diversity of web content, the hierarchy can even be automatically extracted. It is thus less rigid than the example-based learning and constraining.

In the rest of this paper, we first provide an overview of PHC in Section 3. Section 4 gives details of the PHC problem and algorithm. Section 5 presents experiments and results analysis. Related work is reviewed in Section 5, and with the conclusion in Section 6.

2. PROTOTYPE HIERARCHY BASED CLUSTERING

Generally, PHC takes in as input a *prototype hierarchy* and a target collection on the same topic, and produces as output a *data hierarchy* that contains all the data items from the collection. For ease of discussion, we first give some basic definitions and notations.

2.1 Preliminaries and Notations

Preliminary 1. A ***Hierarchy*** *(H) is defined as a tree that consists of a set of uniquely labeled nodes V and a set of parent-child relations R between these nodes. A* ***Concept Hierarchy*** *(CH) is a hierarchy whose V represents a set of concepts ℓ, with each ℓ being used as a label for each V.*

Definition 1. A **Prototype Hierarchy** (PH) is defined as a hierarchy whose nodes set V represents a set of $< \ell, \rho >$ tuples, with ρ a **Prototype** serving as a typical example, description, or standard for the concept ℓ.

Definition 2. **Data Hierarchy** (DH) is a hierarchy that organizes a collection of **objects** d. Each node of DH represents a category of objects CO. Non-leaf nodes subsumes their child nodes in a recursive manner. The root of DH consists of a single broadest category containing all objects, and the leaves correspond to the finest categories.

PH is the hierarchy that supervises the categorization process. It can be seen as a concept hierarchy CH with each ℓ labeled node embodied by a prototype ρ. With PH and DH, we define the problem of Prototype Hierarchy based Clustering (PHC) as follows. Given a collection D of objects on a topic τ, PHC partitions and maps D into the categories that are predefined by a PH on τ, such that the formed objects clusters CO_1, CO_2,..., CO_k are organized in a DH with similar structures. The output DH is readily labeled by the PH, and thus could be easily browsed by users to find information at different granularity.

2.2 An Example

Figure 1 illustrates how the prototype hierarchy based categorization works. Suppose the problem is to organize an archive of Yahoo! Answer questions on *iPhone*. Given the dataset of questions and the predefined prototype hierarchy as shown in Figure 1, PHC assigns each object (question) of the dataset into the leaf nodes of the hierarchy. For instance, question 1 is categorized into *MobileMe* and naturally becomes a member of *Online Services*. Note that the category *IPhone:Software:Interface* does not have a single object and question 7 has no appropriate category to assign to. These are two typical cases to be handled in the PHC algorithm. With all the objects being assigned, a data hierarchy that has the same structure with the prototype hierarchy is formed. A user may thus easily browse the organized dataset by navigating the prototype hierarchy and clicking on any node to view the questions from the corresponding node of the data hierarchy.

This example suggests that clustering of a dataset according to a supervision hierarchy is not trivial. We identify the following requirements in the study of this paper:

1. The data hierarchy, like a taxonomy or an ontology, is incrementally evolving into a compact structure *encoding* the underlying topics of the collection.
2. The data and prototype hierarchy are to be matched at both the node and relation level, while special techniques are needed to handle the mis-match between the data hierarchy and the prototype hierarchy.
3. The distance between objects are measured by appropriate metrics, so as to partition the objects into homogeneous clusters that are far apart from each other.

These requirements suggest the criteria in constructing a data hierarchy from a dataset. They form the basis of the PHC framework. In the following Section, we address the requirements in the first three subsections, and induce a multi-criterion optimization function in the last subsection.

3. PROBLEM FORMULATION

3.1 Data Hierarchy Structure Evolution

3.1.1 Hierarchy Metric and Information Function

To characterize the structure of a hierarchy, we introduce Hierarchy Metric and Information Function inspired by research in automatic taxonomy induction [19] and ontology

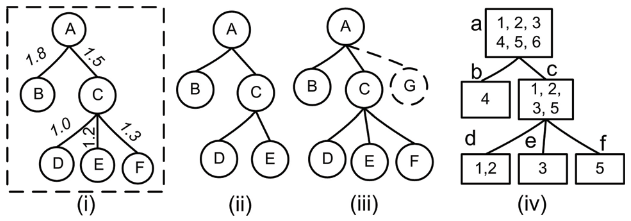


Figure 2: Illustration of Prototype Hierarchy (i-iii) and Data Hierarchy (iv). The prototype hierarchy (i) is a full match of (iv), (ii) an incomplete match of (iv), and (iii) an excess match of (iv).

learning, due to the similar nature of hierarchy and taxonomy.

We define a hierarchy metric as a function that operates on all the nodes in a hierarchy, similar to ontology metric [19] on an ontology. Formally, it is a function $h: V \times V \rightarrow R+$, where V is the set of nodes in H. $h(.,.)$ is recursively defined. For an adjacent pair of nodes v_p and v_q, the hierarchy metric is defined as the edge weight $w(e_{v_p v_q})$. For the other pairs, the hierarchy metric $h(.,.)$ on H with edge weights w for any node pair $v_i, v_j \in V$ is the sum of all edge weights along the shortest path between the pair:

$$h_{H,w}(v_i, v_j) = \sum_{e_{v_p v_q} \in P(v_i, v_j)} w(e_{v_p v_q}) \qquad (1)$$

where $P(v_i, v_j)$ is the set of edges defining the shortest path from nodes v_i to v_j. The quality of the structure of a hierarchy is measured by the amount of information carried in H, defined as the sum of all hierarchy metrics in H:

$$Info(H) = \sum_{i<j, v_i, v_j \in V} h(v_i, v_j) \qquad (2)$$

where $i < j$ reduces duplicated entries of $h(.,.)$ since the hierarchy metric is a symmetric measure.

Figure 2 (i) gives an example of a 6-node hierarchy. We can calculate the hierarchy metric between A and F as $h(A, F) = h(A, C) + h(C, F) = 2.8$, and the Information Function of the hierarchy as the sum of 15 pairs of nodes, resulting in $Info(H) = 38.5$.

3.1.2 Objective Functions

Minimum Evolution(obj_1) is designed to monitor the structural evolution of the data hierarchy. The data hierarchy is incrementally hosting more objects until the whole collection is categorized and allocated. We assume that $DH^{(n+1)}$ with $n+1$ nodes to be the one that introduces the least changes of information from its previous status $DH^{(n)}$:

$$DH^{(n+1)} = arg\,min_{DH'} ||Info(DH^{(n)}) - Info(DH')|| \quad (3)$$

Therefore the optimal DH organizes the whole collection so as to introduce the least information changes since the initial data hierarchy $DH^{(0)}$: $\hat{DH} = arg\,min_{DH'} ||Info(DH^{(0)}) - Info(DH')||$, where $Info(DH^{(0)}) = 0$ since the initial DH is empty. By plugging in Equation 1 and 2, the minimum evolution objective function becomes:

$$minimize \quad obj_1 = \sum_{i<j, v_i, v_j \in V} \sum_{e_{v_p v_q} \in P(i,j)} w(e_{v_p v_q}) \qquad (4)$$

obj_1 suggests that the optimal DH on a collection is the one that contains the least information. It makes intuitive sense that the DH that compactly "encodes" the collection into topic categories is the best.

3.1.3 Data Hierarchy Centroid

The hierarchy metric and information function discussed above are defined in "node space". For PH, the nodes are represented by prototypes. For DH, we use centroid to represent a node of objects CO. Note that in previous work [7] on classification, centroid and prototype are two equivalent and interchangeable concepts. In this paper, we distinguish the two concepts by emphasizing that prototype is knowledge oriented and centroid is data oriented.

The centroids for DH nodes are generated in an incremental manner. When the first object is categorized into a category, it acts as the initial centroid of the category (and its ancestor categories). With subsequent objects being inserted into the same category, the centroid is updated incrementally upon its previous status.

Suppose that the centroids and the objects are represented by vectors on the term space as $\overrightarrow{v}_{centroid}$ and $\overrightarrow{d}$. When a new object d is inserted into a node, its centroid is updated by taking the algorithmic average of all the existing objects and $\overrightarrow{d}$, as $(n\,\overrightarrow{v}^{(n)}_{centroid} + \overrightarrow{d})/(n+1)$.

The new object in a leaf node automatically becomes members of its ancestor nodes whose centroids are to be updated too. We consider that the magnitude of the change decreases with the levels from the leaf node. The updating formula for a data hierarchy centroid is defined as

$$\overrightarrow{v}^{(n+1)}_{centroid} = \frac{n\,\overrightarrow{v}^{(n)}_{centroid} + g(t)\,\overrightarrow{d}}{n+1} \qquad (5)$$

where t is the number of edges on the shortest path between the updated node and its descendent leaf nodes, and $g(t)$ is a monotonically decreasing updating coefficient. In the implementation, a heuristic function $g(t) = 1 - t/|H|$ is utilized, $|H|$ is the height of the data hierarchy H, and $g(t) \in (0.0, 1.0]$. For example,when v is a leaf node, $t = 0$, and $g(t) = 1.0$.

3.2 Matching of Prototype Data Hierarchy

*Preliminary 2. A **full match** between two hierarchies H_1 and H_2 is defined such that nodes $V_1 = V_2$ and relations $R_1 = R_2$. A **partial match** between H_1 and H_2 can be either an incomplete match or an excess match. When H_1 is an incomplete match of H_2, $V_1 + V_{in} = V_2$, $R_1 + R_{in} = R_2$, and the incomplete rate is defined as $|V_{in}|/|V_2|$; when H_1 is an excess match of H_2, $V_1 = V_2 + V_e$, $R_1 = R_2 + R_e$, and the excess rate is defined as $|V_e|/|V_2|$. In the case of a partial match, the matched nodes and relations constitute a sub-hierarchy called a **common hierarchy**.*

Figure 2(iv) shows a data hierarchy, and Figure 2(i-iii) show three prototype hierarchies that are respectively a full match, an incomplete match, and an excess match of the data hierarchy. The common hierarchies of Figure 2(iii) and (iv) are (A,B,C,D,E,F) and (a,b,c,d,e,f), and G is an excess node.

3.2.1 Objective Functions

Prototype Centrality (obj_2) We assume that a prototype is located at the center of an object cluster in the object space. Formally, the prototype centrality is expressed as

$$maximize \quad obj_2 = \frac{1}{|V|} \sum_{\forall v_{o_i} \neq \emptyset} c(v_{p_i}, v_{o_i})$$

where $|V|$ is the number of nodes in the hierarchy, v_{p_i} is represented by its prototype ρ_i, and v_{o_i} is represented by its centroid. v_{o_i} is updated with incrementally added new objects. $c(.,.)$ measures the similarity between a v_{p_i} and a v_{o_i} that is not empty. For this study, $c(.,.)$ employs the simple and effective cosine similarity function on the term vectors of v_{p_i} and v_{o_i}.

The maximization of the prototype centrality objective is actually equivalent to adding a data object into a node, so that the updated centroids (including the parental node centroids), are most similar to their corresponding prototypes.

Prototype-Data Hierarchy Resemblance (obj_3) considers the common part of the data hierarchy and the prototype hierarchy. Formally, the prototype-data hierarchy resemblance is defined as follows:

$$minimize \quad obj_3 = \frac{1}{|M|} \sum_{i<j} ||h(v_{p_i}, v_{p_j}) - h(v_{o_i}, v_{o_j})|| \quad (6)$$

where $|M|$ is the size of the common hierarchy, $v_{p_i}, v_{p_j} \in V_p$ and $v_{o_i}, v_{o_j} \in V_o$ are the corresponding nodes of PH and DH respectively.

Since PH is predefined and static, it is usually a partial match of DH. To enable DH to flexibly adjust to the collection distribution by having more or less nodes, obj_2 and obj_3 are measured on the common hierarchy.

3.2.2 Considerations for Partially Matched Prototype Hierarchy

In the ideal setting, the prototype hierarchy is the one that fully represents the underlying topic structure in the target collection, *i.e.*, the resulting data hierarchy is a full match. However, it is not possible to define such an ideal prototype hierarchy. Therefore the implementation should consider the cases when the predefined PH is a partial match.

If the predefined PH is an incomplete match of the underlying DH of a collection, we expand PH by adding more nodes to accommodate more categories. This is solved by adding dummy child nodes to the existing nodes in PH. For instance, in the Figure 1 example, equestion 7 has no appropriate category to assign to. This question will thus be assigned to an unnamed (dummy) node as the child of category *IPhone:Online Service: Itunes Store*[1].

The added nodes, however, do not have a specific concept label and prototype description. For such scenario, we adopt the existing label extraction algorithms [4] for these new categories. The disadvantage of post-categorizing labeled concepts is that they may be less consistent with the existing concept space.

If the predefined PH is an excess match of the underlying DH of a collection, some of the nodes in PH may result in empty category in DH, such as *IPhone:Software:Interface* in the Figure 1 example. For such cases, the empty nodes will be labeled as empty or removed from the browsing interface.

[1]This is a combined effect of all the objectives, if only obj_2 and obj_3 are considered, question 7 will be assigned to *IPhone:Online Service: Itunes Store:audio* category.

3.3 Object Metric

Under Data Hierarchy, Object Metric $M(d_i, d_j)$ is defined as the distance (similarity) between a pair of objects d_i and d_j within a node. Theoretically, any metric that satisfies the non-negativity, symmetricity, and triangular inequality criteria is a valid metric. For text clustering, a desirable metric is the one that captures the lexical, syntactic, and semantic features of the texts. We explore two state-of-the-art text retrieval models for estimation the distance metrics, the translation-based language model and the syntactic tree kernel matching model.

3.3.1 Translation-based Language Model

Translation-based language model (TBLM) is originally proposed to solve the lexical gap problem in document retrieval. The monolingual translation probabilities capture the lexical semantic relatedness between mismatched terms in the query and the documents. We employ TBLM to measure the semantic similarity between two texts d_1 and d_2. The similarity score function is similar to the retrieval function proposed by Xue *et al* [18]:

$$P_{TBLM}(d_1|d_2) = \Pi_{w \in d_1} P(w|d_2)$$
$$P(w|d_2) = (1-\lambda)P_{mx}(w|d_2) + \lambda P_{ml}(w|D)$$
$$P_{mx}(w|d_2) = (1-\beta)P_{ml}(w|d_2) + \beta \sum_{t \in d_2} P(w|t) P_{ml}(t|d_2)$$

where $P(w|d_2)$, the probability that w is generated from document d_2, is smoothed using $P_{ml}(w|D)$, the prior probability that w is generated from the document collection D. λ is the smoothing parameter. $P_{mx}(w|d_2)$ is the interpolated probability of $P_{ml}(w|d_2)$ and the sum of the probabilities that w is a translation of t, $P(w|t)$, weighted by $P_{ml}(t|d_2)$. P_{ml} is computed using the maximum likelihood estimator.

Due to its asymmetricity, $P_{TBLM}(d_1|d_2)$ cannot be directly applied as a metric between a pair of text d_1 and d_2. We thus define a symmetric distance metric by taking the average of the two scores that switch the role of d_1 and d_2 as the "query" and "document":

$$M_{TBLM}(d_1, d_2) = \frac{1}{2}(P_{TBLM}(d_1|d_2) + P_{TBLM}(d_2|d_1)) \quad (7)$$

3.3.2 Syntactic Tree Kernel Matching Model

The tree kernel function is one of the most effective ways to represent the syntactic structure of a sentence [15]. Syntactic Tree Kernel Matching Model(STKM) is designed based on the idea of counting the number of tree fragments (subtrees) that are common to both parsing trees:

$$Sim(T_1, T_2) = \sum_{w_1 \in W_1, w_2 \in W_2} C(w_1, w_2)$$

where W_1 and W_2 are sets of nodes(terms) in two syntactic trees T_1 and T_2, and $C(w_1, w_2)$ is the number of common tree fragments rooted in nodes w_1 and w_2. Wang *et al* [15] improved node matching function $C(w_1, w_2)$ by adapting the tree kernel function to take into account the syntactic as well as semantic variations.

STKM is originally designed to measure the similarity between two sentences. We generalize it into a similarity metric between multiple-sentence texts as follows:

$$M_{STKM}(d_1, d_2) = \frac{\sum_{s_i \in d_1} \sum_{s_j \in d_2} sim(T(s_i), T(s_j))}{|d_1||d_2|} \quad (8)$$

Table 1: Statistics of Dataset (♭ indicates webpages, † indicates question answer pairs).

statistics		ODP		Yahoo! Answers	
		Computer **S**cience	**R**eligion & **S**pirituality	Dental	IPod
Prototype Hierarchy	Max.depth	6	5	5	4
	♯concepts	145	177	104	87
	♯leaf concepts	83	106	62	51
Collection	♯objects	2085♭	3909♭	6735†	4381†

where s_i and s_j are sentences from d_1 and d_2 respectively.

M_{TBLM} and M_{STKM} emphasize on semantic and syntactic similarity of text objects respectively, we propose an integrated object metrics by taking their interpolation:

$$M_{TB-ST}(d_1, d_2) = \alpha M_{TBLM}(d_1, d_2) + (1-\alpha) M_{STKM}(d_1, d_2) \quad (9)$$

With the above defined $M(.,.)$, similar objects can be better clustered into the same category.

3.3.3 Objective Functions

Category Cohesiveness (obj_4) objective requires that the collection is categorized such that objects in the same category are similar to each other and those in different categories are dissimilar to each other. More specifically, when the intra-category similarity is the highest, and the inter-category similarity is the lowest, the categorization achieves the highest cohesiveness. Formally, the cohesiveness of the data hierarchy is defined as:

$$maximize\ obj_4 = \frac{\sum_{\forall v_k \neq root} \sum_{d_p, d_q \in v_k} M(d_p, d_q)}{\sum_{\forall v_p \neq leaf} \sum_{v_i, v_j \in child(v_p)} c(v_i, v_j)} \quad (10)$$

where $c(v_i, v_j)$ measures the cosine similarity between the centroids of v_i and v_j. Note that in the denominator, only sibling categories compared, rather than as a whole as in [4]. The assumption is that categories at different levels could be different in terms of abstractness and thus not comparable.

3.4 Multi-Criterion Optimization Function

In Section 3.1-3.3, we have discussed the criteria on constructing a data hierarchy, and induced four objective functions obj_1,obj_2, obj_3, and obj_4. In our proposed prototype hierarchy based clustering framework, all the criteria are to be satisfied, therefore the four objectives are to be optimized simultaneously:

$$minimize \quad O_m = \pi_1 obj_1 - \pi_2 obj_2 + \pi_3 obj_3 - \pi_4 obj_4 \quad (11)$$

where π_1, π_2, π_3, and π_4 are introduced to control the contribution of each objective within the range of 0 to 1.

The multi-criterion optimization function leads to a greedy optimization algorithm, which at each object insertion step, produces a new data hierarchy by adding the new object into an appropriate node, which minimizes O_m.

4. EXPERIMENTS

4.1 Datasets

To evaluate the proposed prototype hierarchy based clustering scheme, we apply the techniques developed to reconstruct the subdirectories of ODP, and to organize Yahoo! Answers questions according to prototype hierarchies from external knowledge source. Table 1 shows the statistics of the prototype hierarchies and the associated collections of the four datasets.

For the ODP datasets, the prototype hierarchies are constructed by extracting the subcategories of two topics, *Computer Science* (CS) and *Religion and Spirituality* (RS). The subcategory descriptions are extracted as prototypes. Some subcategories for portals (*e.g.*, classified, directory) or those labeled by alphabetic orders, which are uninformative for the purpose of categorization and navigation, are removed. The two collections contain websites belonging to the categories of the extracted prototype hierarchies; where homepages of these websites are the objects of the collections.

The datasets under the topics *Dental* and *IPod* are collected from YA. The prototype hierarchy for the *Dental* dataset is directly extracted from Wikipedia hierarchy under *Dentistry*, where the prototype for each subcategory is the first part (the definition) of the corresponding Wikipedia article. The prototype hierarchy for *IPod* is a manually constructed hybrid hierarchy by combining Wikipedia *IPod* article hierarchy, Wordnet *IPod* meronyms, and product specification from *IPod* website). The corresponding prototypes are also the combined descriptions from these three sources. The objects of the two collections are questions downloaded using YA API from *Dental* and *Music & Music players*. We asked two dentistry graduate students and two computer science graduate students to organize the two collections by reading through the downloaded archives. Inter-rater agreements in terms of Kappa statistics are 85% and 91% for *Dental* and *IPod* respectively. The differences between the two annotators are made consistent by discussion.

Each of the four collections are equally divided into two parts (C_1 and C_2) for training/developing and testing. The results are presented by taking the average of the two suits of experiments, using either part as testing sets.

The four datasets are carefully constructed to represent different scenarios. CS is a topic with a deep hierarchy, while RS has a broad hierarchy. IPod represents a concrete domain and RS an abstract domain. ODP hierarchies (CS and RS) are noisier than Wikipedia hierarchy (Dental); while the semi-manually constructed hierarchy (IPod) has better quality.

4.2 Overall Performance

4.2.1 Experimental Setting

To evaluate the performance of the proposed PHC model, we compare the following systems:

1) *proKmeans*: a prototype hierarchy enhanced K-means divisive hierarchical clustering. We choose a divisive algorithm as our baseline, as divisive algorithms has been found to be better solutions than agglomerative algorithms [20]. K-means works well when K and the initial partitioning are properly set. At each step of the division, we set K to be the number of leaf nodes in the prototype hierarchy; and the associate prototypes as the initial centroids. In this way, an intuitive unsupervised method is enhanced to be a relatively strong baseline.

2) *LiveClassifier* [9]: a state-of-the-art hierarchical classifier. We employ the approach 3 and KNN as the learning algorithm as in [9]. We adopt Yahoo BOSS API[2] as the

[2]http://developer.yahoo.com/search/boss/

Table 2: Comparison of the proposed PHC, the two baselines, and a supervised method CFC in terms of μF_1 and mF_1.

Methods	CS		RS		Dental		IPod	
	μF_1	mF_1	μF_1	mF_1	μF_1	mF_1	μF_1	mF_1
proKmeans-B1	0.623	0.547	0.63	0.644	0.601	0.592	0.612	0.608
LiveClassifier-B2	0.656	0.641	0.618	0.625	0.683	0.663	0.667	0.629
PHC-BOW	0.732	0.713	0.741	0.729	0.755	0.703	0.764	0.713
over B1	17.5%	30.3%	17.6%	13.2%	25.6%	18.8%	24.8%	17.3%
over B2	11.6%	11.2%	19.9%	16.6%	10.5%	6.0%	14.5%	13.4%
PHC-TBLM	0.757	0.732	0.749	0.756	0.803	0.783	0.822	0.767
over B1	21.5%	33.8%	18.9%	17.4%	33.6%	32.3%	34.3%	26.2%
over B2	15.4%	14.2%	21.2%	21.0%	17.6%	18.10%	23.2%	21.9%
PHC-STKM	0.791	0.78	0.775	0.788	0.764	0.721	0.778	0.719
over B1	26.9%	42.6%	23.0%	22.4%	27.1%	21.8%	27.1%	18.3%
over B2	20.5%	21.7%	25.4%	26.1%	11.9%	8.7%	16.6%	14.30%
PHC-TB-ST	0.869	0.853	0.842	0.851	**0.885**	0.879	**0.893**	**0.889**
over B1	39.5%	55.9%	33.7%	32.1%	47.2%	48.5%	45.9%	46.2%
over B2	32.5%	33.1%	36.2%	36.2%	29.6%	32.6%	33.9%	41.3%
supervised CFC	0.904	0.884	0.851	0.857	0.879	0.904	0.866	0.854

search engine to gather snippets as training examples. The number of pseudo nodes for leaf node is set to be 6. We set K=1 for result evaluation. This method is used as the second baseline.

3) *PHC-BOW, PHC-TBLM, PHC-STKM*, and *PHC-TB-ST*: 4 variations of PHC using Bag-of-Word, TBLM, STKM, and the combined TBLM and STKM respectively as the object metric. For π_1, π_2, π_3, and π_4 in Equation 10, we perform an exhaustive grid search of step size 0.1 on $[0, 1]$ to find parameters that produce the best μF_1 on the developing set. For TBLM, λ is set to 0.8 and β to 0.5; the translation probabilities are trained using the set C_1/C_2 and tested on C_2/C_1. For STKM, the four parameters (the node/size/depth weighting factors, and the weight of the matching tree fragment) are tuned by using the set C_1/C_2 as the development set. For TB-ST, α is set to 0.5.

4) *CFC* Classifier [7]: a state-of-the-art supervised text categorization technique. It is included to test the effectiveness of semi-supervised PHC against a supervised method. Experimental results are averaged using either set C_1/C_2 as the training set.

We use the average accuracy of categorizing the leaf categories as the performance measure for each dataset. In particular, we use the micro-averaging F_1 (μF_1) and macro-averaging F_1 (mF_1) as the performance metrics. F_1 is a combined form for precision (p) and recall (r), which is defined as $F_1 = 2rp/(r + p)$.

4.2.2 Results and Discussion

The results are evaluated on all the leaf nodes and displayed in Table 2. By comparing the vertical entries by different methods, we draw the following observations:

(1) Both (unsupervised) baselines achieve reasonably high μF_1 of about 0.6. For *LiveClassifier*, the results are consistent with that reported in the original experiment in [9] which shown it to be comparable to a supervised approach. For *proKmeans*, it achieves better μF_1 than that reported in a state-of-the-art K-means clustering algorithm [8] that employs the sophisticated semantic features. This suggests that by specifying a prototype hierarchy for a collection, even a simple method like divisive K-means can categorize the collection reasonably well.

(2) PHC with TB-ST surpasses all the other unsupervised systems. All the four PHC systems perform significantly better than the two baselines. Even the simplest PHC design with BOW-based metric achieves improvement of above 10% over both baselines. This indicates that PHC is superior in terms of utilizing the prototype hierarchy. *proKmeans* makes use of the prototypes and the number of children under each node; whereas *LiveClassifier* makes use of the relations of nodes and node labels (concepts). Either baseline benefits from the prototype hierarchy but not as comprehensively as the PHC's.

(3) PHC achieves the best performance when the object metrics (TBLM and STKM) are used in combination (TB-ST). It is however difficult to compare the PHC with TBLM and PHC with STKM. Generally, syntactic tree kernel based method works better on the two ODP collections; and translation based method works better on the two YA collections. We conjecture that the ODP collections are more standard than YA in term of English grammar and thus more suitable for syntactic tree parsing; while the YA collections use parallel question-answer corpus which is more suitable for generating translation probabilities. The combination of TBLM and STKM yields significant performance improvement when compared to the individual model. This shows that semantic and syntactic features complement each other and contribute to the overall result.

(4) PHC with TB-ST even achieves comparable result with CFC, a state-of-the-art supervised classification algorithm. CFC can be deemed as using the hand-labeled corpora, while PHC makes use of hand-built hierarchy. This implies that a prototype hierarchy created by experts or web community is enough to help create good categorization of a large web collection, instead of needing to manually organize and label the large corpus. Moreover, PHC provides the additional benefit of facilitating navigation.

(5) PHC introduces new nodes into predefined hierarchy. In PHC-TB-ST setting, the numbers of new nodes introduced are 7 for CS, 11 for RS, 5 for Dental, and 3 for IPod. Since we deem the prototype hierarchies and collections in Table 1 as fully matched, the objects in the newly added nodes are simply evaluated as incorrect for ease of experimentation. This indicates that the results in Table 2 may underestimate the actual performance of PHC.

4.3 Impact of Domain Abstractness and Prototype Quality

By comparing the horizontal entries of Table 2, we draw the following observations of PHC performance on different domains and prototype hierarchies:

(1) PHC works better on concrete domains than on abstract domains. This is evidenced by comparing PHC performances on the two ODP collections. *Computer Science* achieves higher μF_1 scores than *Religion and Spirituality* for the last 3 settings of PHC. It can be explained by two reasons: (i) the concepts in concrete domains like CS are more specific, therefore the prototypes associated with the concepts are more precise, informative, and have potentially more word overlapping with the target objects; (ii) the subtopics of abstract domains like RS are less systematically arranged in a hierarchical structure and the ancestor-descendent relations between the subtopics are less obvious;

Table 3: Objectives analysis. % of change in μF_1 when a single objective is removed.

objectives	CS	RS	Dental	IPod
All-obj_1	-1.4%	+3.2%	+2.7%	0.0
All-obj_2	-11.1%	-10.6%	-8.7%	-9.6%
All-obj_3	-5.8%	-4.0%	-6.6%	-7.4%
All-obj_4	-10.5%	-6.7%	-9.3%	-8.8%

therefore PHC is harder to benefit from the "loose" hierarchies of the abstract domains.

(2) A better prototype hierarchy can potentially enhance the categorization performance. Table 2 shows that in the PHC settings, IPod collections (with semi-manually compiled hierarchy) attains the best results, followed by Dental (with Wikipedia hierarchy), and the worst are the two ODP collections. This indicates that high quality hierarchy will lead to better results. Besides the quality of prototype hierarchy, the quality of the prototypes used may also influence the categorization performance. Prototypes from ODP category descriptions are of various qualities, depending on the devotion of the category editor. Prototypes from Wikipedia articles are mediated by a larger community and thus maintain a high level of quality. We conjecture that when more high quality hierarchies are available in digital form, PHC can achieve even better performance and wider adaptability.

4.4 Ablation Study on Optimization Objectives

We do a leave-one-out study on the optimization objectives to analyze the effects of each objective on the categorization. In the implementation, we set one of the parameters (π_1, π_2, π_3, and π_4) in Equation 11 to 0, and optimize the rest using grid search.

Table 3 shows that removing an objective from the multi-criterion optimization generally results in degraded performance. Prototype Centrality (obj_2) influences all the four collections greatly, followed by Category Cohesiveness (obj_4). Prototype-Data Hierarchy Resemblance (obj_3) influences Dental and IPod more than CS and RS.

obj_2 is a bit more complex. A prototype may not represent the perfect center for a cluster. Two cases exist: (i) the prototype provided is roughly located at the center of a category's object space; the objects that are closer to the true center but farther away from the prototype may be adversely influenced by the prototype centrality score. (ii) the target collection is not large enough to form typical categories; even though the prototype provided might be good, the object center itself may shift from general knowledge.

An interesting observation is that removing obj_1(minimum evolution) increases the μF_1 measure on RS and Dental. By examining the clustering output, we find that the data hierarchy varies less from the prototype hierarchy without the minimum evolution. For example, two websites about *Youth for Human Rights* are under ODP category *Religion and Spirituality: Scientology: Church of Scientology: Volunteer and Community Activities*; the two websites are correctly categorized when the minimum evolution objective is removed. Under the full-fledge multi-criterion setting, the output data hierarchy has a new node created to accommodate the two websites under *Religion and Spirituality: Scientology: Church of Scientology*, as a sibling of *Volunteer and Community Activities*. Since there is no standard on whether some objects should be assigned into a new node or an existing node, the phenomenon implies that minimum evolution objective leads to a self-contained data hierarchy.

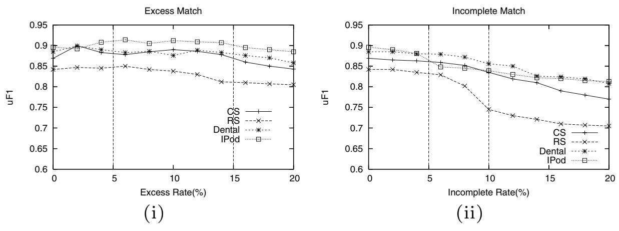


Figure 3: The influence of partially matched prototypes on PHC performance.

4.5 Robustness with Mismatched Prototype Hierarchy

To further study the tolerance of PHC with partially matched prototype hierarchies, we deliberately manipulate the prototype hierarchies and the collections to examine the two types of mismatching effects: incomplete and excess prototype hierarchy.

We mimic an incomplete (insufficient) prototype hierarchy by deleting nodes from the completely matched prototype hierarchy. Similarly, we mimic an excess(overfitted) prototype hierarchy by inserting dummy nodes into the current hierarchy. Alternatively, we remove objects belong to a certain node from the collection, such that the node in the prototype hierarchy becomes redundant. We adopt the second approach because lacking certain groups of objects is more common in practical applications.

In Figure 3, we plot μF_1 of PHC results on the four collections with excess rate and incomplete rate ranging from 0 to 20%. The two rates are defined in Section 3.2. From Figure 3(i), we can see that the μF_1 measures on all the four collections slight degrade over the $0 - 15\%$ excess rate. By examining the output object hierarchy, we find that for most excess nodes, PHC produces empty clusters. It suggests that PHC's is robust against overfitted prototype hierarchies.

From Figure 3(ii), we can see that incomplete match is well tackled at $0 - 5\%$ range and degrades drastically from 10% onwards. In the experimental setting, at the lower incomplete rate, only some leaf nodes are removed; whereas at higher incomplete rate, even those sub-hierarchies are removed. This suggests that PHC has only limited ability to "create" categories. We conjecture that it is because PHC is designed to add one level of child nodes to existing nodes in the prototype hierarchy without considering multiply levels. In other words, our system needs further improvement to tackle the high incomplete rate, or seek more complete match prototype hierarchies to avoid the problem.

5. RELATED WORK

Hierarchical clustering has long been recognized as a natural way to organize and navigate text collections [10, 5, 17]. Existing algorithms for hierarchical clustering are generally either agglomerative, divisive, or combined [20]. The automatically generated clusters are however less neatly organized as a manually constructed hierarchical tree like the ODP and Wikipedia hierarchies. Another limitation is that the clusters do not have labels to indicate the topics contained. Further more hierarchical clustering, labeling is more complicated since an internal node in the hierarchy has to be distinguished from its siblings, parent, and children.

World knowledge has been found to be useful in enhancing clustering and labeling. For clustering, metric based [16] and constraints based [14] approaches utilize knowledge in the form of a small amount of labeled samples. Bilenko *et al* [2] integrated the two approaches and obtained further improvement over either approach. Hu *et al* [8] proposed to enrich short texts representation for clustering with syntactic and semantic features from WordNet and Wikipedia. For cluster labeling, Carmel *et al* [4] successfully enhanced it by extracting candidate labels from Wikipedia, in addition to the important terms that are extracted directly from the text. In our work, world knowledge comes in the form of a topic hierarchy and prototype descriptions.

Various criterion functions for document clustering have been studied in [20]. These functions represent some of the most widely used criteria for document clustering, but not cover the structural aspects of the hierarchies.

Hierarchy and taxonomy induction has long been studied on concepts [13], noun-phrases [19], and word-based topics [3]. For the first time it has been applied as a criterion on hierarchical clustering in this work. Moreover, the concept hierarchies [13] and taxonomies [19] could be used to automatically extract prototype hierarchies for the PHC framework.

Our work is similar in spirit to conceptual clustering [12] which is distinguished from ordinary data clustering by generating a concept description for each generated class. Another similar work is LiveClassifier [9]. It tries to tackle the clustering and labeling problem in a reverse order. Assuming a predefined topic hierarchy, it augments the hierarchy and uses search engines to automatically gather training corpus for classifying websites into the hierarchy. Our framework is similar to theirs in term of specifying a topic hierarchy, but exploits more on the structure and evolution of the hierarchy rather than searching for training data.

6. CONCLUSION

This paper proposed a prototype hierarchy based clustering framework for web collection categorization and navigation. By minimizing the hierarchy evolution, maximizing category cohesiveness and inter-hierarchy structural and semantic resemblance, the hierarchical clustering task is modeled as a multi-criterion optimization problem. Empirical results on categorizing 4 web collections of various domains have shown that PHC is superior to the two strong unsupervised baseline methods and comparable to a state-of-the-art supervised method.

In future work, we plan to optimize the efficiency of the proposed PHC algorithm and explore its applicability to multimedia collections with domain specific hierarchy and object metrics.

7. REFERENCES

[1] R. Baeza-Yates. User generated content: how good is it? In *Proc. of the 3rd workshop on Information credibility on the web*, pages 1–2, Spain, 2009. ACM.

[2] M. Bilenko, S. Basu, and R. J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proc. Machine learning*, page 11, Banff, Alberta, Canada, 2004. ACM.

[3] D. M. Blei, T. L. Griffiths, M. I. Jordan, and J. B. Tenenbaum. Hierarchical topic models and the nested chinese restaurant process. In *Advances in Neural Information Processing Systems*, page 2003. MIT Press, 2003.

[4] D. Carmel, H. Roitman, and N. Zwerdling. Enhancing cluster labeling using wikipedia. In *Proc. SIGIR*, pages 139–146, Boston, MA, USA, 2009. ACM.

[5] S. Dumais and H. Chen. Hierarchical classification of web content. In *Proc. SIGIR*, pages 256–263, Athens, Greece, 2000. ACM.

[6] P. Ferragina and A. Gulli. A personalized search engine based on web-snippet hierarchical clustering. In *Proc. WWW*, pages 801–810, Japan, 2005. ACM.

[7] H. Guan, J. Zhou, and M. Guo. A class-feature-centroid classifier for text categorization. In *Proc. WWW*, pages 201–210, Spain, 2009. ACM.

[8] X. Hu, N. Sun, C. Zhang, and T.-S. Chua. Exploiting internal and external semantics for the clustering of short texts using world knowledge. In *Proc. CIKM*, pages 919–928, Hong Kong, China, 2009. ACM.

[9] C.-C. Huang, S.-L. Chuang, and L.-F. Chien. Liveclassifier: creating hierarchical text classifiers through web corpora. In *Proc. WWW*, pages 184–192, New York, NY, USA, 2004. ACM.

[10] S. Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.

[11] D. J. Lawrie and W. B. Croft. Generating hierarchical summaries for web searches. In *Proc. SIGIR*, pages 457–458, Toronto, Canada, 2003. ACM.

[12] R. Michalski and R. Stepp. Learning from observation: Conceptual clustering. *Machine Learning*, 1:331–363, 1983.

[13] M. Sanderson and B. Croft. Deriving concept hierarchies from text. In *Proc. SIGIR*, pages 206–213, New York, NY, USA, 1999. ACM.

[14] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl. Constrained k-means clustering with background knowledge. In *Proc. Machine Learning*.

[15] K. Wang, Z. Ming, and T.-S. Chua. A syntactic tree matching approach to finding similar questions in community-based qa services. In *Proc. SIGIR*, pages 187–194, Boston, MA, USA, 2009. ACM.

[16] E. Xing, A. Ng, M. Jordan, and S. Russell. Distance metric learning with application to clustering with side-information. *Advances in neural information processing systems*, pages 521–528, 2003.

[17] G.-R. Xue, D. Xing, Q. Yang, and Y. Yu. Deep classification in large-scale text hierarchies. In *Proc. SIGIR*, pages 619–626, Singapore, 2008. ACM.

[18] X. Xue, J. Jeon, and W. B. Croft. Retrieval models for question and answer archives. In *Proc. SIGIR*, pages 475–482, Singapore, 2008. ACM.

[19] H. Yang and J. Callan. A metric-based framework for automatic taxonomy induction. In *Proc. ACL*, pages 271–279, Suntec, Singapore, 2009. ACL.

[20] Y. Zhao, G. Karypis, and U. Fayyad. Hierarchical clustering algorithms for document datasets. *Data Min. Knowl. Discov.*, 10(2):141–168, 2005.