Automatic Maintenance of Category Hierarchy

Hai Zhuge and Lei He

Institute of Cyber-Physical-Social Intelligence, Nanjing University of Posts and Telecommunications, Nanjing, China
Key Laboratory of Intelligent Information Processing, University of Chinese Academy of Sciences, Chinese Academy of Sciences, Beijing, China
Systems Analytics Research Institute, School of Engineering and Applied Science, Aston University, Birmingham, UK

Abstract — Category hierarchy is an abstraction mechanism for efficiently managing large-scale resources. In an open environment, a category hierarchy will inevitably become inappropriate for managing resources that constantly change with unpredictable pattern. An inappropriate category hierarchy will mislead the management of resources. The increasing dynamicity and scale of online resources increase the requirement of automatically maintaining category hierarchy. Previous studies about category hierarchy mainly focus on either the generation of category hierarchy or the classification of resources under a pre-defined category hierarchy. The automatic maintenance of category hierarchy has been neglected. Making abstraction among categories and measuring the similarity between categories are two basic behaviors to generate a category hierarchy. Humans are good at making abstraction but limited in ability to calculate the similarities between large-scale resources. Computing models are good at calculating the similarities between large-scale resources but limited in ability to make abstraction. To take both advantages of human view and computing ability, this paper proposes a two-phase approach to automatically maintaining category hierarchy within two scales by detecting the internal pattern change of categories. The global phase clusters resources to generate a reference category hierarchy and gets similarity between categories to detect inappropriate categories in the initial category hierarchy. The accuracy of the clustering approaches in generating category hierarchy determines the rationality of the global maintenance. The local phase detects topical changes and then adjusts inappropriate categories with three local operations. The global phase can quickly target inappropriate categories top-down and carry out cross-branch adjustment, which can also accelerate the local-phase adjustments. The local phase detects and adjusts the local-range inappropriate categories that are not adjusted in the global phase. By incorporating the two complementary phase adjustments, the approach can significantly improve the topical cohesion and accuracy of category hierarchy. A new measure is proposed for evaluating category hierarchy considering not only the balance of the hierarchical structure but also the accuracy of classification. Experiments show that the proposed approach is feasible and effective to adjust inappropriate category hierarchy. The proposed approach can be used to maintain the category hierarchy for managing various resources in dynamic application environment. It also provides an approach to specialize the current online category hierarchy to organize resources with more specific categories.

Keywords — Category, Category Hierarchy, Classification, Clustering, Maintenance, Resource Space Model

1 Introduction

ORGANIZING resources in category hierarchy is one of the most basic and natural ways to efficiently manage resources in the physical space and cyberspace. Goods in supermarket are classified in clear sections, which correspond to a category hierarchy of goods. File systems use category hierarchy for efficiently managing files in computer. Online hierarchical categories such as DMOZ (Open Directory Project) and Yahoo! Directory have been widely used to organize online resources. XML (eXtensible Mark-up Language) is a Web standard for specifying Web resources in a hierarchical structure.

Organizing resources in a category tree can well represent common human abstraction. The categories closer to the root are more general, and categories gradually become more special following the paths from the root to the leaves.

The unpredictable influence of constantly coming resources in an open environment makes it necessary to adapt the category hierarchy to the change of the patterns in resources. Current studies mainly focus on hierarchy generation and hierarchical classification under a pre-defined hierarchy. In March 2014, DMOZ created a new category relating to Malaysia Airlines flight 370 under the category "*Accidents*" and earlier in May 2013 it added category "*Wearable Electronics*" under category "*Hardware*". The ACM Classification System has also modified its category hierarchy three times (in 1991, 1998 and 2012 respectively) in recent twenty years. Some commercial websites like Amazon and eBay adjusted their category hierarchies more frequently since some new types of items often emerge.

The initial hierarchical category represents its creator's view on the pattern of resources, but the pattern may change after continually adding various resources. The initial hierarchy needs to be updated otherwise it may mislead resource management, e.g., saving, updating or retrieving resources according to inappropriate categories.

The following two cases lead to the change of the pattern of the resources in a category when new resources are added to the hierarchy.

1. On one hand, new resources may be added to less relevant categories when it is difficult to determine their categories. On the other hand, the pattern within one category may change with constantly adding new resources to a category. This will lead to less topical cohesive categories. When less relevant resources within a category accumulate to a certain degree, it

will seriously influence the semantics of the category hierarchy and consequently hurt the effectiveness of applications and user experience when operating resources.

2. The semantics of resources rendering the categories significantly changes with the change of the real world, which are hard to be predicted at the stage of creating the category hierarchy. For example, the documents on topic "*Crimea*" belongs to the category "*Geography*" of the initial hierarchy. When the Crimea referendum happened, Crimea has become a sensitive term of the relations among Russia, Ukraine and European Union. New documents about "*Crimea*" will have a stronger link to the category "*Politics*" than to the category "*Geography*". It is reasonable to put the new resources into the category "*Politics*" for the applications that concern politics.

Manual maintenance is difficult because it is hard to discover the changes of the patterns of large-scale resources in categories and the emerging topics. This motivates our research on automatic maintenance of the category hierarchy by detecting the changes of the internal patterns of categories. An automatic maintenance mechanism is useful in many applications, e.g., for specializing the current general online category hierarchies (such as Wikipedia, DMOZ directory and Yahoo! Directory, which mainly contain general categories) to organize resources with more specific categories.

Little work has studied the category hierarchy maintenance problem. In 2006, an approach to modifying a hierarchy using three operations (*Promote*, *Merge* and *Demote*) was proposed [22]. It tests each operation on each category. Its major problem is that it cannot change incohesive leaf categories because all of the three operations aim to change relations between categories instead of directly changing categories. Thus this approach cannot solve the aforementioned cases where leaf categories may need to be split into finer subcategories. Another data-driven approach defines three operations (*Sprout*, *Merge* and *Assign*) [23], relying on an auxiliary hierarchy to discover emerging topics. The auxiliary hierarchy limits its ability in discovering new topics.

Two kinds of inappropriate cases will happen in the initial category hierarchy with the change of resources: the inappropriate location of category within global scope, and the change of patterns of resources within category. This requests a global adjustment and a local adjustment.

This paper proposes an approach Automatic Maintenance of Hierarchical Category (*AMHC*) to modify category hierarchy through two-phase adjustment to meet the needs of managing resources in dynamic environments.

A way to conduct the global adjustment is to generate a cluster tree and adjust the category hierarchy considering both the category hierarchy and the binary category tree generated by the clustering approach. By using a reliable clustering approach, the global phase can detect inappropriately located categories and adjust them to the appropriate locations within the global scope.

A way to the local adjustment is to detect topical changes within categories by using the Latent Dirichlet Allocation (LDA) topic model [3, 15]. The statistical topic model can potentially discover a broad range of hidden themes but lacks interpretability. The initial category hierarchy (usually designed by human) reflects human views but it is hard to cover all future topics. Combining the initial category hierarchy and the topic model, the local adjustment can take both advantages.

The global adjustment carries out cross-branch adjustments while the local adjustment modifies the categories that are only related to their parent categories or sibling categories by three elementary operations (*Merge*, *Pull-Up* and *Split*).

To evaluate the new category hierarchy, we propose a new evaluation measure *UC_Score* (Uncertain Score) that considers not only the balance of the hierarchical structure but also the ability of expressing the semantics of the category hierarchy. *UC_Score* uses the entropy to measure the uncertainty of classification, the balance of structure and resource distribution.

We conduct experiments on the datasets of various scales, comparing *UC_Score* with traditional F1-Measure and classification accuracy on three types of hierarchies (original hierarchy, automatically generated hierarchy and AMHC-modified hierarchy). The experimental results show that classifiers trained on the modified hierarchy can get better classification performance than that on the original category hierarchy and automatically generated category hierarchy, which verifies that the modified hierarchy has more topically cohesive categories than the other two hierarchies. Besides, the comparison of the evaluation measures also shows that *UC_Score* is more suitable for evaluating the quality of a hierarchy than the traditional measures.

2 Representation, Typical Structures Analysis and Modification Strategies

2.1 Representation

A category C_k represents a set of resources R sharing a pattern P, represented as $C_k = (R, P)$. The pattern may change with the change of resources. This paper uses topic model to represent the pattern. So the representation of a category consists of an identity, entities of resources, the number of resources, and the topic distribution.

A category hierarchy CH is a partial order on a category set such that if two categories satisfy the partial order then the resources belong to them also satisfy a partial order, represented as $CH = (\{C_1, \ldots, C_n\}, \leq, \subseteq)$, where $\{C_1, \ldots, C_n\}$ denotes a set of categories, $\leq$ denotes the subcategory relation between categories, and $\subseteq$ denotes the inclusion relation between resource sets, and satisfies:

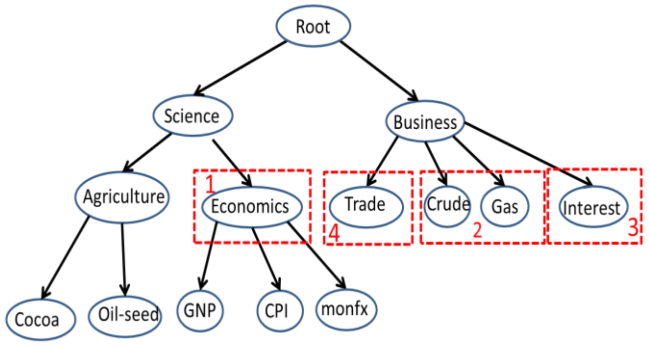

Fig. 1 The category hierarchy generated from ODP.

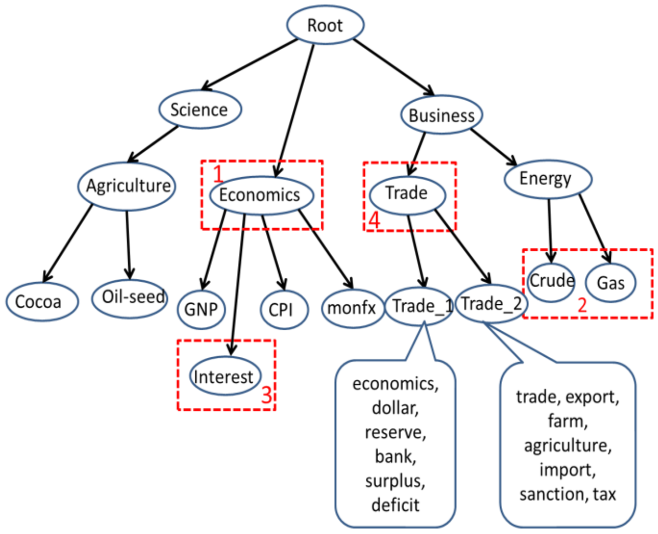

Fig. 2 The category hierarchy evolved from Fig.1.

(1) If $C_i \leq C_j$, then $R_i \subseteq R_j$.
(2) For all children of a category $C=(R, P)$, $C_1=(R_1, P_1)$, …, $C_k=(R_k, P_k)$, $R=R_1\cup \ldots \cup R_k$.
(3) For any two children $C_l=(R_l, P_l)$ and $C_m=(R_m, P_m)$ of the same category, $R_l \cap R_m=\varnothing$.

2.2 Typical Structure Analysis

When using existing hierarchical categories like Wikipedia to organize resources, inconsistence often exists between hierarchical categories and resources, which will lead to inefficient management of the resources. There are four typical cases of a category hierarchy that need adjustments.

Case 1: *Parent category can no longer represent its child category*.

Case 2: *Two categories under the same parent share too many common features to distinguish them clearly*.

Case 3: *A category belongs to more than one parent category*.

Case 4: *Leaf categories become less cohesive with new coming resources*.

The above cases are illustrated in Fig.1, a category hierarchy that is generated for Reuters-21578 dataset according to DMOZ directory (Open Directory Project). The categories marked by dashed rectangles in red colour correspond to the above cases.

2.3 Modification Strategies

To adjust these typical inappropriate structures in a category hierarchy, four modification strategies are proposed as follows. Fig.2 shows a better category hierarchy evolved from Fig.1 with using the modification strategies.

Modification Strategy for Case 1. *Pull the child category up to its parent level to avoid the inappropriate influence from the parent. And exclude the representation of the child category from the current parent, and add the representation of the child category to the target parent*.

In Fig.1, both the category *"Agriculture"* and the category *"Economics"* are under the category *"Science"*, but the resources of Retuers-21578 in *"Economics"* are more related to the category *"Business"* than to the category *"Agriculture"*. So a better solution is to pull the category *"Economics"* up to the upper level as shown in Fig.2. This operation leads to a better classification performance according to the F1-Measure (raised from 0.84 to 0.93).

Modification Strategy for Case 2. *Merge similar categories under the same parent to form a new node. And, create a new representation of the new node and update the representation of the parent node to represent the new node*.

By selecting the common features, we can firstly distinguish the similar categories from others and then focus on more specific features to separate the similar categories at the lower level.

As shown in Fig.1, the category *''Business''* contains two similar subcategories *''Crude''* and *''Gas''*. They can be merged into a super category *''Energy''* shown in Fig.2. As the result of the operation, we get a better classification performance as indicated by F1-Measure, increasing from 0.65 to 0.79.

Modification Strategy for Case 3. *If a category belongs to more than one parent, put the category under the parent to achieve better classification accuracy. And, Remove the representation from the parent that loses the category and update the representation of the target parent.*

As shown in the example, the category *''Interest''* is moved to a new parent category *''Economics''* in Fig.2 from the category *''Business''* in Fig.1. After this operation, the F1-Measure increases from 0.83 to 0.90.

When new resources are continually added to the category hierarchy, leaf categories are more likely to emerge new topics, and thus become less cohesive. The following strategy is necessary.

Modification Strategy for Case 4. *Split the leaf category into finer subcategories. And, create the representations for the newly generated subcategories and update the representation of the current category to reflect subcategories.*

Algorithm 1: AMHC Global Modification

```
Input:
        Cla_HT: the predefined hierarchical classification tree;
        Clu_HT: the resources hierarchical clustering tree.
Output:
        HT: the modified hierarchy tree.
Algorithm:
  1.  Eva_Score = evaluateHT(Cla_HT);
  2.  HT ← Cla_HT;
  3.  AdjustNodesList = null;
  4.  AdjustNodesList ← Mapping(Cla_HT, Clu_HT);
  5.  Repeat
  6.     Node ← getNode(AdjustNodesList);
  7.     H_List ← generateCandidates (Node, HT, Clu_HT);
  8.     [H_temp,score] ← getBest(H_List);
  9.     if score < Eva_Score
  10.        Eva_Score = score; HT = H_temp;
  11. Until AdjustNodesList == null;
  12. HT ← PostProcess(HT);
  13. return HT;
```

Algorithm 1. The Global Modification Algorithm.

Applying this strategy to split the less cohesive category *''Trade''* in Fig.1 into *"Trade-1"* (a category related to the relationship between trade and economics) and *"Trade-2"* (a category related to the import and export trade policy among countries) as shown in Fig.2. After this operation, the F1-Measure increases from 0.68 to 0.73.

Based on the above four cases and modification strategies, we develop a two-phase category hierarchy maintenance approach. The global phase solves the issue of case 3 by directly moving inappropriate child categories to their better parents within a global scope. The local phase addresses the other three cases through detecting topical changes in some categories and using three elementary operations (*Merge*, *Pull-Up* and *Split*) to modify a category that is only related to its parent or sibling category within a local range.

The two-phase approach can make a hierarchical category more suitable to organize the resources that cannot be represented by existing categories.

3 Automatic Maintenance Mechanism

Making abstraction among categories and measuring the similarity between categories are two basic behaviors to generate a category hierarchy. Humans are good at making abstraction but limited in ability to calculate the similarities between large-scale resources. Computing models are good at calculating the similarities between large-scale resources but limited in ability to make abstraction. To make both advantages of humans and computers, our Automatic Maintenance of Hierarchical Category (*AMHC*) approach use a global phase and a local phase to maintain the category hierarchy within two different scales.

The global phase gets initial human-defined hierarchy and then makes use of hierarchical clustering to get similarity between categories to detect inappropriately located categories. The local phase detects topical changes by LDA topic model [3] and then adjusts with three local operations: *Merge*, *Pull-Up* and *Split*.

3.1 Phase 1: Global Modification

A hierarchy evolves when the number of new resources reaches a certain degree. To adjust the category hierarchy, we need to detect the pattern change of similarity between categories to guide the category hierarchy evolvement. Hierarchical clustering can generate a cluster tree that reflects the similarity of categories, but it can only generate a binary tree with specific clusters.

How to adjust a category hierarchy according to the hierarchical cluster tree of resources and keep the levels of abstraction similar to the original one is the main problem of global modification. To address the problem, we firstly build one-to-one mappings between categories in category hierarchy and cluster tree, and then adjust the category hierarchy. Algorithm 1 illustrates the global phase process.

The general process of global modification consists of two major procedures mapping procedure (line 4) and candidates generating procedure (line 7). The algorithm takes a classification tree (*Cla_HT*) and a cluster tree (*Clu_HT*) as the input and then outputs the final modified category hierarchy *HT*. It firstly evaluates *Cla_HT* (line 1) by the evaluation measure *UC_Score*. The smaller the value, the better quality a hierarchy has. Then, it proceeds with the mapping procedure (line 4) between *Cla_HT* and *Clu_HT*. After that, we will get a list of categories to be adjusted (*AdjustNodesList*). For each node in the list (line 5-11), it generates the candidates (line 7) and gets the best one (*H_temp*) in terms of the *UC_Score* (line 8). It tests the evaluation score of *H_temp* and decides whether to accept it or not (line 9-10). At last, it carries out a post-process (line 12) on the final hierarchy to avoid single-child situations that commonly occur in candidates.

Fig.3 shows an example of the global modification on *Cla_Tree*. There is only one category that needs to be adjusted (the square leaf node). This example includes the mapping and candidates generating procedures and shows two types of mapping *Complete-Image* and *Incomplete-Image* on each pair of green and red nodes. If a node in *Cla_Tree* can be mapped by *Complete-Image* to a node in *Clu_Tree*, all subnodes under it are located appropriately. Otherwise, it contains some improperly located categories (like the square node) that need modifications.

3.1.1 Mapping Procedure

To build link between the category hierarchy and the cluster tree, we define two types of mapping *Complete-Image* and *Incomplete-Image* and give a new concept of *Pattern Consistence* based on *Complete-Image* to reflect whether the category hierarchy has the consistent topical clusters within that cluster tree.

DEFINITION 1. (Node with Labels) *Given a Category Hierarchy Tree H_c or a Cluster Tree C_T, the Label Set of a Node n in H_c or C_T is defined as follows:*

①$\forall$Node n $\in$ Leaf node of H_c or C_T, Labels(n) = Cat_ID;

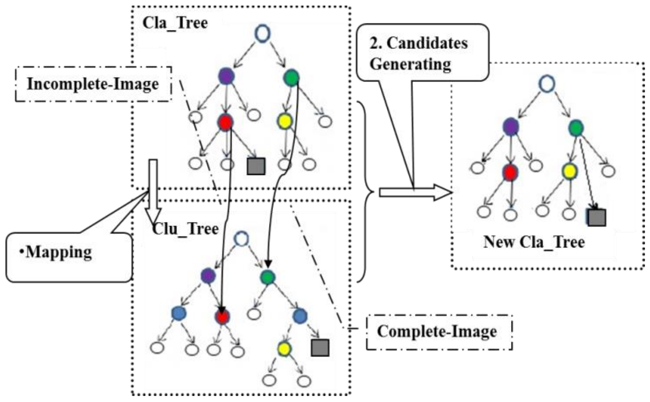

Fig. 3 Global Modification Example.

②$\forall Node\ n \in Internal\ node\ of\ H_c$ or C_T, *Labels(n)* = $\bigcup_{n^* \in Child(n)} Labels(n^*)$;

where Cat_ID is the category identification of the node n.

Using the label set, we define the following concepts.

DEFINITION 2. (Complete-Image) *Given an internode n in H_c, if there exists a node n^* in C_T, such that:*

①*Labels(n^*) ⊇ Labels(n);*

②*There doesn't exist node $n^{'} \in Sub\_Tree(n^*)$, satisfying Labels($n^{'}$)⊇ Labels(n);*

Then there is a Complete-Image mapping between n and n^*.

DEFINITION 3. (Incomplete-Image) *Given an internode n of H_c, if there exists a node n^* in C_T, such that:*

①$\forall Node\ n^{'} \in C_T$, *L($n^*$) = Labels($n^*$) ∩ Labels(n), L($n^{'}$) = Labels($n^{'}$) ∩ Labels(n), |L($n^*$)| ≧ |L($n^{'}$)|;*

②*There doesn't exist node m, such that* $m \in Sub\_Tree(n^*)$, *Labels(m)⊇ L(n^*);*

Then there is an Incomplete-Image mapping between n and n^*.

DEFINITION 4. (Pattern Consistence) *Given a Hierarchy* $Tree\ H_c$ *and a Cluster Tree C_T, there is a one-to-one mapping between leaf categories of H_c and C_T. The classification pattern of H_c and the clustering pattern of C_T are satisfied with Pattern Consistence under the following two conditions:*

①$\forall Node\ n \in Internal\ node\ of\ H_c$, $\exists\ n^* \in C_T$, *such that f(n)= n^*, f:=Complete Impl;*

②$\forall n_1,\ n_2 \in Internal\ node\ of\ H_c$, *if f($n_1$)=$n_1^*$, f($n_2$)=$n_2^*$, f:=Complete Impl, then* $n_1^* \neq n_2^*$.

Condition ① and ② ensure that if H_c and C_T are satisfied with pattern consistence, then for each node in H_c there is a different mapping node in C_T and the mapping type is the *Complete-Image*.

Mapping is to find a *Complete-Image* for each node in category hierarchy and if all nodes can be mapped by *Complete-Image* into the cluster tree, we don't need to modify the hierarchy. If some nodes are mapped by *Incomplete-Image*, then there are some categories to be adjusted to achieve *Pattern Consistence*.

In Fig.3, the mapping procedure is a top-down manner to project nodes in category hierarchical tree (*Cla_Tree*) into nodes in cluster tree (*Clu_Tree*). Nodes of the same colour between *Cla_tree* and *Clu_tree* are pairs of one-to-one mapping. In this example, the mapping between the green pair is *Complete-Image*, while the mapping of the red pair is *Incomplete-Image*. Because of the square node, the red node in *Cla_Tree* can't find a *Complete-Image* mapping node in *Clu_Tree*, so that *Cla_Tree* and *Clu_Tree* can't achieve *Pattern Consistence*. We will adjust the square node through candidates' generation procedure to change the category hierarchy into a new one (*New Cla_Tree*) that satisfies *Pattern Consistence* with the original *Clu_Tree*.

3.1.2 Candidates Generating Procedure

Categories leading to a failure of pattern consistence should be relocated at the appropriate position in the hierarchy. We generate the candidates by moving the node to a child of the nearest ancestor that has been mapped.

In Fig.3, the grey square node needs to be relocated. In *Clu_Tree*, we find the nearest mapped ancestor of the grey square node is the green node. So we move the white square node to the child of the green one and get the new *Cla_Tree* shown in the right part of Fig.3.

3.2 Phase 2: Local Adjustment

Global modification is to break up some obviously inappropriate parent-child relations to make the original hierarchy satisfy with the *Pattern Consistence* of the clustering results. However, the satisfaction of *Pattern Consistence* cannot guarantee the best expression of classification. For example, a global modification can solve the problems of case 1 and case 3 which are described in Section 2, but it cannot handle the problem of case 2 and case 4. Since different category hierarchy can all satisfy *Pattern Consistence* with the same cluster tree, it is necessary to do some localized adjustments on the category hierarchy.

Three elementary operations conduct local adjustments. Relevant representations will be updated as described in the modification strategies.

1. *Pull-Up*: pull up one node to its parent's level to be a sibling of its parent.
2. *Merge*: merge two nodes under the same parent into one.
3. *Split*: split a leaf node into finer nodes and add these new nodes as the children of the leaf node.

Local adjustment is achieved by testing the three elementary operations on some specific nodes. With the feedback of the classification results, we can pick up nodes satisfying the following premises: (1) $P \ll \bar{P}$ and (2) $P \gg R$, where P and R represent the classification precision and recall of each category and $\bar{P}$ is the average precision of categories at the same level.

```
Procedure: Merge-Operation()
Input:   A, the node to check;
         Cla_HT, the current hierarchy;
Output: HT, the modified hierarchy.
Algorithm:
    1.  HT ⟵ Cla_HT;   Oflag = True;
    2.  Pa ⟵ getParent(A);
    3.  Eva_Score = evaluateHT(Pa, HT);
    4.  While (Oflag)
    5.      Clist ⟵ getChildren(Pa);
    6.      For each node n in Clist:
    7.          B ⟵ arg min Sim(A, n);   s = Sim(A, B);
    8.      For each node n in Clist:
    9.          Threshold = min Sim(n, B);
    10.     if s < Threshold
    11.         [A, H_temp] ⟵ Merge(A, B, HT);
    12.         score = evaluateHT(Pa, H_temp);
    13.         if score < Eva_Score
    14.             HT ⟵ H_temp;   Eva_Score = score;
    15.     else
    16.         Oflag = False;
    17. return HT;
```

Algorithm 2. *Merge* Operation Procedure.

We set trigger conditions for each operation. If a category satisfies the trigger conditions, we test the corresponding operation and compare the new evaluation score with the original one to make a decision whether to accept the operation or not. Here we use the proposed evaluation measure *UC_Score* to judge whether the quality of a hierarchy is improved. Compared to using traditional measures like F-Measure and classification accuracy, the advantage of applying *UC_Score* is that it takes not only classification performance but also navigation balance and resource distribution into consideration.

We conduct LDA topic model to make the category associated with a topic distribution that gives a coarse description of the category. LDA is a probabilistic generative model [3], where documents are represented as random mixtures over latent topics and a topic is a distribution over words. For each category, we compute the average topic mixtures over documents to get the category-topic distribution (the mean document-topic distribution over documents in the category). We can use the category-topic distribution to describe the inner pattern of categories.

The generative process of LDA topic model for each document w is described as follows [3] with notations descriptions in Table 1:

1. Choose $\theta \sim Dir(\alpha)$.
2. For each of the N words w_n:
   a. Choose a topic $z_n \sim Multinomial(\theta)$.
   b. Choose a word w_n from $p(w_n|z_n, \beta)$, a multinomial probability conditioned on the topic z_n.

Gibbs sampler [18] is applied to infer the topic distribution and the word distribution. In our experiment, we empirically set the number of topics $K=100$ and hyper-parameters $\alpha=50/K$ and $\beta=0.1$. After obtaining the topic distribution, we can use it to define trigger conditions for *Merge, Pull-Up* and *Split* operations.

TABLE 1
DESCRIPTIONS OF NOTATIONS USED IN LDA

	Descriptions
N	The number of words in a document
α	Hyper parameter
Z	The topic set, z is a topic in Z
Θ	The topic distribution set, θ is a topic distribution
β	The word distribution set

3.2.1 Merge Operation

When performing the *Merge* operation, we need to detect whether there is another category that is similar to a certain degree with the current one under the same parent. *Merge* operation is triggered if the category similarity exceeds a threshold value. The key challenges to set the trigger condition for Merge operation include the following two questions:

1. How to measure the similarity between categories?
2. How to set the threshold value?

We define the similarity of two categories using category-topic distribution.

DEFINITION 5. (Category Similarity) *Given two categories A and B with their topic distributions* $\vec{\theta}_A$ and $\vec{\theta}_B$, *the similarity is defined as:*

$$\text{Sim}(A, B) = \sum_{k\in K} I_{R\neq 0}(\theta_{Ak})\, I_{R\neq 0}(\theta_{Bk})\,\theta_{Ak}\, log\frac{\theta_{Ak}}{\theta_{Bk}}.$$

$I_A(x)$ is the indicator function, if $x \in A$, $I_A(x)$ is equal to 1 else 0. θ_{Ak} and θ_{Bk} represent the k$^{\text{th}}$ topic proportion of category A and B respectively. The smaller the value, the more similar the category is. This metric says that two categories similar to each other share a similar combination of topics.

We show the general *Merge* procedure in Algorithm 2. Suppose that we pick up category A to check. Then we compute the most similar category to A under the same parent, denoted as category B (line 5-7). The threshold value can be set to the minimum category similarity between B and any other categories under the same parent except A (line 8-9). If the *Merge* operation can improve the hierarchy, then we accept it (line 10-14).

3.2.2 Pull-Up Operation

If a parent node cannot cover the topics of its child category, it should be pulled-up to the upper level in order to avoid the influence from the inappropriate parent node.

For each category, we define the *Cover-Ratio* for a given parent category A and its child B as the trigger condition.

DEFINITION 6. (Cover Ratio) *Given a parent category A and its child B with their topic distributions* $\vec{\theta}_A$ and $\vec{\theta}_B$, *the Cover Ratio is defined as:*

$$\textit{Cover-Ratio}\ (A, B) = \sum_{\text{Topic}_k \in Cat(B)} (\log \theta_{Bk} + \log \theta_{Ak}).$$

Cat (B) is the significant topic set consisting of the top-k

major topics in category *B*. θ_{Ak} and θ_{Bk} represent the kth topic proportion of category *A* and *B* respectively. If *Cover-Ratio* (*A*, *B*) exceeds a threshold value, then we say that category *A* can cover its child category *B*. Otherwise *A* can't cover *B*. So *Pull-Up* operation is triggered if category *A* can't cover its child category *B*.

Suppose that we pick up category *B* to check. Category *A* is *B*'s parent. For *Pull-Up* operation, the threshold value can be set to the average *Cover-Ratio* of all the children under category *A* with a multiplier $\delta \in [0, 1]$ to control the degree of coverage. Too small δ will overload CPU to test improper *Pull_Up* operations, while too big δ may lead to missing some necessary *Pull_Up* modifications on inappropriately located categories. Thus δ is empirically set to 0.7 in our study. There is another way to set δ according to the resource distribution on child category *B*. In this way, δ is set to the percentage of the number of resources in category *B* to the number of resources in the parent category *A*.

As the general procedure of *Pull_Up* operation is just similar to the *Merge* operation, we don't give the full algorithm for it.

3.2.3 Split Operation

As new resources are increasingly added into the category hierarchy, some of them can't find proper categories and we may put them under less relevant categories. This behavior will lead to less cohesive categories, especially for leaf categories. When less relevant resources in a leaf category accumulate to a certain degree, we need to split the category into finer sub-categories.

Split is operated when the category cohesion is smaller than a threshold value and the percentage of the number of resources in the category to the number of resources in its parent category is larger than a threshold value (empirically set to 50% in our experiment). For each category, we define the concept of *category cohesion.*

DEFINITION 7. (Category Cohesion) *Given a category A with its topic distribution $\vec{\theta}_A$, the Cohesion is defined as:*

$$Coh\ (A) = \sum_{\mathrm{Topic_i,Topic_j} \in Cat(A)} (\log \theta_{\mathrm{Ai}} + \log \theta_{\mathrm{Aj}}) * Dist(i,j).$$

Cat (*A*) is the significant topic set of category *A*. θ_{Ai} and θ_{Aj} represent the ith and jth topic proportion of category *A* respectively. Since topic is represented by a distribution over words, Dist(*i*, *j*) computes the cosine similarity of word distribution between $\mathrm{Topic_i}$ and $\mathrm{Topic_j}$. The smaller the value of *Coh(A)*, the less cohesive the category *A* is.

Suppose that we pick up category *A* to check for *Split* operation. The threshold value can be set to the average category cohesion of all the categories under the same parent with category *A* also with a multiplier $\xi \in [0, 1]$. ξ is set to the ratio of the number of resources in category *A* to the number of resources in its parent category in our experiment.

Unlike the other two operations, how to perform the *Split* operation is a major problem. Clustering algorithms can help partition topics in the significant topic set, but it is still difficult to anticipate a proper number of clusters. A split with neither too few nor too many subcategories is preferable to humans. To solve this problem, we firstly use hierarchical clustering algorithm to generate a binary tree of topics. The average-linkage function defined as the average of all similarities among the topics in both clusters is used to measure the similarity between any pair of clusters. Then we apply *Min-Max Partitioning* proposed in [5] to select the best cutting level that minimizes the criteria function combining the *cluster set quality* and the *cluster number preference*.

Let C be a set of clusters. The *cluster set quality Q(C)* is defined as:

$$Q(C) = \frac{1}{|C|} \sum_{C_i \in C} \frac{Sim(C_i, \bar{C_i})}{Sim(C_i, C_i)}$$

where $Sim(C_i, \bar{C_i})$ is the inter-similarity between cluster C_i and C_j (j≠ i). Let Sim(C_i, C_j) be the average of all pairwise similarities among the topics in C_i and C_j. Sim(C_i, C_i) is the intra-similarity within cluster C_i. Let Sim(C_i, C_i) be the average of all pairwise similarities among topics within C_i. The smaller the value *Q*(C), the better the quality of the cluster set C is.

The *cluster number preference* uses a gamma distribution function to measure the degree of preference on the number of clusters at each layer. We change $\alpha!$ into $(\alpha - 1)!$ to make this formula reflect the preference cluster number. Let C be a set of clusters. The *cluster number preference N(C)* is defined as:

$$f(x) = \frac{1}{(\alpha-1)!\beta^{\alpha}} x^{\alpha-1} e^{-x/\beta};$$

$$N(C) = f(|C|)$$

where α and β are two parameters to tune the smoothness of the preference function and they are empirically set as α = 3 and $\beta = N_{clus}/2$. N_{clus} is the expected number of clusters and in our experiment it is empirically set to the square root of the number of topics in the significant topic set.

The best cutting level *l* should minimize the criteria function of $Q(C(l))/N(C(l))$. $C(l)$ is the set of clusters produced by cutting level *l* on the hierarchical clustering binary tree.

The *Split* operation uses generated clusters on the best cutting level as new finer subcategories and uses the top-*k* ranked keywords of the topic nearest to the centroid of the cluster to re-label the new category.

3.3 Evaluation Measure

To evaluate the quality of a hierarchy, we propose *UC_Score* that combines structural aspect and classification aspect to judge whether a hierarchy is comprehensive to use. Previous studies on hierarchy generation and hierarchy maintenance mainly use F-Measure [23], macro-averaged recall [22] or classification accuracy [24] to guide the hierarchy evolvement. However, all these traditional measures only aim to judge the performance of classification algorithms instead of the hierarchy itself.

An evaluation approach to judging the quality of a hierarchy proposed in [5] lists several qualitative measures including:

1. *Cohesiveness, which* is for judging whether the instances in each category are semantically similar.
2. *Isolation, which* is for judging whether categories under the same parent are discriminative from each other.
3. *Hierarchy, which* is for judging whether hierarchical categories go more and more specific from top to bottom with different comprehensive abstraction levels.
4. *Navigation Balance*, which is for judging whether the number of child categories for each internal category is appropriate.
5. *Readability,* which is for judging whether the concepts represented by each category are easy to understand.

Each measure can be assigned numerical scores by humans to reflect the satisfactory degree. However, there is no united calculation form of these measures, thus they can only be judged in an isolated way. For the hierarchy maintenance task, we need an evaluation measure for hierarchies that can be automatically computed in a clear united form. That is why it is necessary to propose *UC_Score* in this paper for automatic hierarchy maintenance.

A good hierarchy is expected to classify resources into each category not only with high classification accuracy but also with a relatively high certainty at each level. The larger the certainty is, the less ambiguity of classification semantics the hierarchy has. Besides the classification aspect, an appropriate hierarchy should try to keep navigation balance among all branches and to avoid heavily leaning on one side. Furthermore, we should also consider whether resources are evenly distributed to the categories of the same level, which is beneficial to user retrieval.

UC_Score uses the Entropy to measure the classification uncertainty, the balance of the hierarchical structure and the uniformity of resources distribution. Entropy is an effective and widely-adopted measure of the uncertainty for a random variable in the field of information theory [14, 20]. The three aspects of a hierarchy in fact measure the uncertainty for classification, structure and distribution and that is why we name the evaluation measure *UC_Score* (*UC* is short for uncertainty).

Therefore, we define *UC_Score* to evaluate the quality of a hierarchy by considering three aspects of a hierarchy: the classification uncertainty represented by *Hc*, the structural balance represented by *Hs,* and the resource distribution represented by *Hr*.

The *UC_Score* of a tree-structured hierarchy rooted by node *n* can be recursively calculated level by level in a top-down manner. As shown in Fig.4, each node in the hierarchy is associated with three values represented by *UC_Score*, *CH_UC* and *Eva*. The *UC_Score* value is a final evaluation value of the hierarchy rooted by the node. The *CH_UC* value is an average of *UC_Score* over all the children nodes. The *Eva* value is an evaluation value only

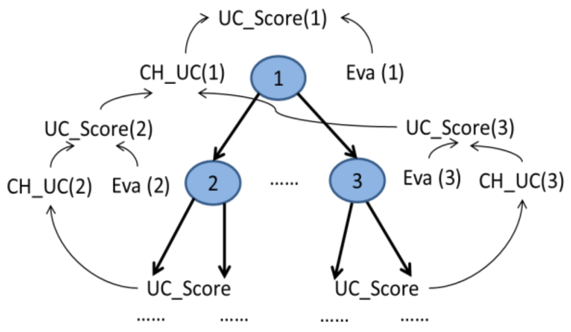

Fig. 4 *UC_Score* Calculation.

related to the current node instead of the hierarchy. In Fig.4, the *UC_Score* value of a node includes two parts. One is its own *Eva* value and the other is the *CH_UC* value.

The calculation form of *UC_Score* of node *n* is defined as:

$$UC\_Score(n) = \begin{cases} \frac{1}{L} * \{Eva(n) + \gamma * CH\_UC(n), & \text{non-leaf node;} \\ 0, & \text{leaf node.} \end{cases}$$

The *UC_Score* of non-leaf node *n* includes its own *Eva* value and the *CH_UC* value (average *UC_Score* over its children nodes) with a discount factor γ. *L* is the number of levels. The discount factor γ is to control the degree of effect on the final *UC_Score* from categories on different levels. It is empirically set to 0.8. The discount factor for each level will have an accumulated effect when going down the hierarchy. The lower the level, the less effect it will have on the final *UC_Score* of a hierarchy.

The calculation form of *CH_UC* of node *n* is defined as:

$$CH\_UC(n) = \frac{1}{m}\sum_{n^* \in Child(n)} UC\_Score(n^*).$$

The *CH_UC* value of node *n* is an average *UC_Score* over all children nodes. In the formula, n^* is the child node of n. *Child(n)* is a set of children nodes of *n*. *m* is the size of *Child(n)* .

The calculation form of *Eva* value of node *n* is defined as:

$$Eva(n) = \frac{H_c}{\alpha H_s + (1-\alpha)H_r}.$$

The *Eva* value of node *n* is computed by combining three variables of the current node *n*: the classification uncertainty *Hc,* the structural balance *Hs* and the resource distribution *Hr*. The three aspects will be detailed respectively. α is a balance factor between *Hs* and *Hr,* which is set to 0.5 empirically in our study.

3.3.1 Classification Uncertainty

The classification uncertainty of a hierarchy reflects the ability to express classification semantics. A preferable category hierarchy is expected to contain categories with maximum intra-category similarity and inter-category discrimination. In other words, resources within a category should be semantically similar and resources from different categories should be discriminative from each other. Category hierarchies satisfying these two characteristics can express clear classification semantics.

When performing classification, a preferable hierarchy is expected to classify resources into each category not only with high classification accuracy but also with a relatively high certainty at each level. The larger the certainty is, the less ambiguity of classification semantics the category hierarchy has.

The resources classification uncertainty is represented by *Hc*. For each resource *r*, we get a probability distribution $p_{r_1}, p_{r_2}, \dots\dots, p_{r_m}$ with which it is classified into m child categories of node n. We compute the entropy of this probability distribution divided by the max entropy to make the value fall into the interval [0, 1]. The max entropy is calculated by classifying the resources into m categories with the same probability of $1/m$.

The calculation form is defined as:

$$H_c = \frac{1}{R} * \sum_{r=1}^{R} \frac{H\left(p_{r_1}, p_{r_2}, \dots\dots, p_{r_m}\right)}{H\left(\frac{1}{m}, \frac{1}{m}, \dots\dots, \frac{1}{m}\right)}.$$

R is the number of resources in the category of node n. m is the number of child nodes of n. *H(-)* represents the entropy of the parameters and the parameters must satisfy the constraints of being a probability distribution. p_{r_i} is the probability of the r$^{\text{th}}$ resource classified to the i$^{\text{th}}$ child category of node n.

3.3.2 Structural Balance

Structural balance is important for user navigation to category hierarchy. A well-structured hierarchy should keep appropriate number of child categories for each internal category.

The balance of the hierarchical structure is represented by *Hs*. We compute the entropy of a probability distribution with which the number of leaf categories assigned to each child node. To make the value fall into the interval [0, 1], it should be divided by the max entropy that is calculated by offering each child node with equal number of leaf categories.

The calculation form is defined as:

$$H_s = \frac{H(\frac{C_1}{C}, \frac{C_2}{C}, \dots\dots, \frac{C_m}{C})}{H(\frac{1}{m}, \frac{1}{m}, \dots\dots, \frac{1}{m})}.$$

C is the number of leaf categories assigned to node n. m is the number of child nodes of n. C_i is the number of leaf categories assigned to the i$^{\text{th}}$ child node of n. *H(-)* represents the entropy of the parameters and the parameters must satisfy the constraints of being a probability distribution.

3.3.3 Resource Distribution

It is beneficial to user retrieval if resources are evenly distributed to categories in a hierarchy. So we consider it as an aspect of the evaluation measure of a hierarchy.

Whether resources are evenly distributed or not is represented by *Hr*. We calculate the entropy of a probability distribution with which the number of resources assigned to each child node. It should also be divided by the max entropy to make the value fall into the interval [0, 1].

TABLE 2
SUMMARY OF DATA SETS

Data Set	Number of Categories			Number of Documents	
	Lev 1	Lev 2	Lev 3	Train	Test
Reuters-25	7	25	-	2,760	994
20Newgroups	7	20	-	11,293	7,061
DMOZ	8	59	121	32,654	43,982

The calculation form is defined as:

$$H_r = \frac{H(\frac{R_1}{R}, \frac{R_2}{R}, \dots\dots, \frac{R_m}{R})}{H(\frac{1}{m}, \frac{1}{m}, \dots\dots, \frac{1}{m})}.$$

R is the number of resources assigned to node n. m is the number of child nodes of n. R_i is the number of resources assigned to the i$^{\text{th}}$ child node of n. *H(-)* represents the entropy of the parameters and the parameters must satisfy the constraints of being a probability distribution.

4 EXPERIMENT RESULTS

4.1 DATA SETS

We use Reuters-21578, 20Newsgroups and DMOZ (Open Directory Project) datasets in our experiments, which are standard datasets for data classification.

Reuters-21578 (https://archive.ics.uci.edu/ml/datasets/Reuters-21578+Text+Categorization+Collection) data set contains documents collected from 135 categories mainly related to economy. We construct a subset from the original dataset. Reuters-25 includes 25 categories among the 135 topics after removing categories that has less than 10 documents in the training set and test set. For each category, we just retain documents with a single label.

20Newsgroup (http://qwone.com/~jason/20Newsgroups) has about 20,000 articles evenly divided into among 20 categories. We use the "Bydate" version for a standard train/test split.

DMOZ (http://www.dmoz.org) dataset is the largest human-edited directory on the Web with over 5,169,995 sites listed in over 1,017,500 categories. However, we just extract a meaningful 3-level hierarchy from the original one, including 8 top categories from the total 16 ones in DMOZ taxonomy, including *"Arts"*, *"Business"*, *"Computers"*, *"Health"*, *"Games"*, *"Recreation"*, *"Science"* and *"Sports"*. Under these categories, we choose 188 categories within the three levels as our hierarchy. After data collecting and cleaning, we remain 46,636 documents.

The general characteristics of our experiment datasets are summarized in Table 2, from which we can find that the smallest data set Reuters-25 just contains 3,754 documents and the largest data set DMOZ contains 46,636 documents. The total number of leaf categories varieties from 25 to 121.

All the data sets are attached with an original coarse hierarchy dividing the topics into several groups of similar classification semantics. They are used as the

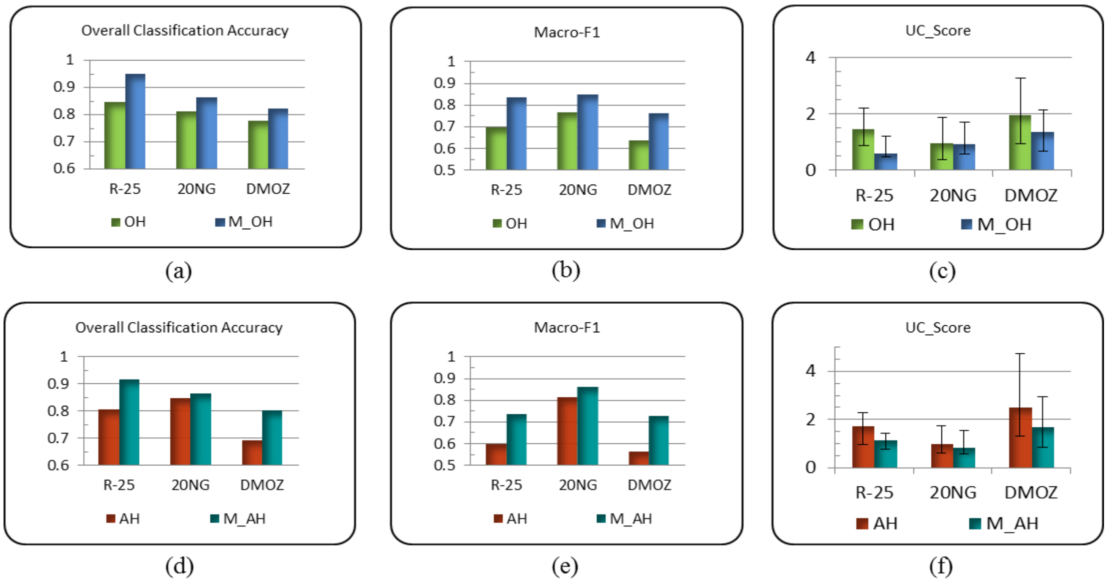

Fig. 5 Classification Performance Comparisons on Classification Accuracy/Macro-F1/*UC_Score*. (a) is the comparison of overall classification accuracy between *OH and M_OH*; (b) is the comparison of Macro-F1 measure between *OH and M_OH*; (c) is the comparison of *UC_Score* between *OH and M_OH*; (d) is the comparison of overall classification accuracy between *AH and M_AH*; (e) is the comparison of Macro-F1 measure between *AH and M_AH*; (f) is the comparison of *UC_Score* between *AH and M_AH*.

initial hierarchy by our *AMHC* approach.

To pre-process the datasets, we remove the stop words with stop word list and prune words occurring less than 5 times and less than 3 documents across the corpus and perform the stemming operations with a Porter Stemmer.

4.2 Baseline Hierarchies

Baseline Hierarchy 1: *Original Hierarchy (OH)*. This topic hierarchy is attached to each dataset dividing the topics into several groups of similar classification semantics. However, it has many inconsistencies with resources.

Baseline Hierarchy 2: *Automatically Generated Hierarchy (AH)*. This hierarchy is generated by the approach HAC+P proposed by [5], which is reviewed in Section 5.

Modified Hierarchy 1: *Modified Original Hierarchy (M_OH)*. This hierarchy is modified from the original hierarchy (*OH*) by our *AMHC* approach.

Modified Hierarchy 2: *Modified Automatically Generated Hierarchy (M_AH)*. This hierarchy is modified from the automatically generated hierarchy (*AH*) by our *AMHC* approach.

4.3 Evaluation Results of Classification

To investigate the effectiveness of our approach, we conduct two groups of comparison experiments. One group is on the original hierarchy (*OH*) and its modified hierarchy (*M_OH*). The other group is between the automatic generated hierarchy (*AH*) and the *AMHC* modified hierarchy (*M_AH*). In our experiments, LibSVM [4] is used as the base classifier to implement the standard hierarchical SVM [6, 12]. We used all the default settings, including the radial basis function kernel. We get a validation set by splitting the training set into two small subsets (70% for training and 30% for validation). JGibbLDA (http://jgibblda.source forge.net) is applied for LDA topic modelling.

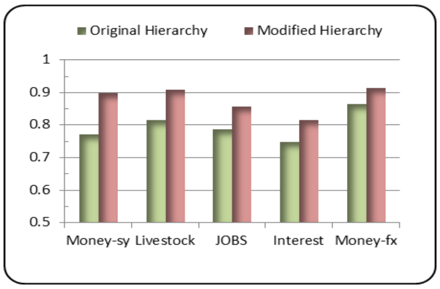

Fig. 6 *F1-Measures* of the top 5 categories in Reuters-25 on *OH* and *M_OH* hierarchies.

We use classification accuracy as the overall evaluation measure, which equals the proportion of correctly classified instances. It is more suitable to multi-class classification problems than F1-Measure, Precision and Recall [10, 21], which are only defined for a specific category. However, it can't reflect the classification performance on each category, so we also list Macro-F1 and show some categories' F1-Measure to explain the overall improvements brought by hierarchy evolvements. We also calculate *UC_Score* with $\alpha = 0.5$ and $\gamma = 0.8$.

Fig.5 consists of 6 figures giving the classification performance and the hierarchy quality comparisons of two groups' hierarchies in terms of the three measures classification accuracy, Macro-F1 and *UC_Score*.

Fig.6 shows the F1-Measures of top 5 improved categories in Reuters-25. Category ''*Money-sy*'' increases mostly by 12.7%. In *OH*, almost all documents in category ''*Money-sy*'' are misclassified into category ''*Money-fx*''. Category ''*Money-fx*'' and category ''*Interest*'' are less distinguishable, however, in *M_OH* we group category ''*Money-fx*'' and category ''*Interest*'' to enhance their common features and it can also enable easier discrimination of category *"Money-sy"*. At a lower level we use more specific features to separate category *"Money-fx"* and category *"Interest"*, increasing 4.8% and 6.8% respectively. For category *"Livestock"* (9.4%) and category *"Jobs"* (7.2%) we adjust them in the first phase by the cross-branches movements to place them under more suitable parents that can better reflect their classification features. That is why we can get the overall improvement of classification accuracy (12.1%) and Macro-F1 (19.8%) on Reuters-25 with *M_OH*.

20Newsgroup achieves almost the same results on *AH* and *M_AH* around 85% in accuracy, which outperform their counterparts (*OH* and *M_OH*) slightly, since auto-generated hierarchy clusters category *"alt.atheism"* and category *"talk.region.misc"* whose resources are more similar. In *M_OH*, we can still observe improvements of 6.5% and 10.5% both on classification accuracy and Macro-F1. Because category *"sci.crypt"* and category *"soc.religion.christian"* are rearranged into their more related parent and cluster, this change directly contributes to the improvement.

DMOZ topic hierarchy is human-edited and its original hierarchy (*OH*) is already a good one to express clear classification, reaching 77.5% of classification accuracy compared with 69.4% on AH. This also shows the inadequate power of HAC+P in generating large taxonomies with wider range of topics. However, a significant improvement of accuracy (15.7%) is achieved on *M_AH*, reaching 80.3%, which is almost the same as that on *M_OH* (82.1%). This indicates that *AMHC* approach can reach a satisfactory hierarchy no matter how terrible conditions the initial hierarchy has.

Compared with classification accuracy and Macro-F1, *UC_Score* has opposite tendency that the smaller the value, the better the hierarchy is, but it reflects consistent results with the other two evaluation measures. In Fig.5, we show the min/max *UC_Scores* (error bars) of different levels for each hierarchy and the final *UC_Score* of the whole hierarchy. The shorter the bar is, the more consistent quality evaluations of different levels of a hierarchy has. In terms of *UC_Score*, the hierarchy *M_OH* on Reuters-25 and the hierarchy *M_AH* on DMOZ have the most significant improvement with 60.3% and 40.0%. In addition, *UC_Score* is more sensitive when detecting bad evolvement of a hierarchy. For example, it can kill the *Merge* operation of category *"Business"* & category *"Economics"* in Reuters-25 and category *"Recreation"* & category *"Sports"* in DMOZ on the first level of the hierarchy, which will result in a heavily skewed tree structure in spite of an increasing in F1-Measure. *UC_Score* enjoys a larger value range $[0, +\infty]$ and considers wider aspects of a hierarchy. That is why it can show more reliable and effective results.

4 RELATED WORK AND COMPARISON

The generation of category hierarchy is to construct a tree-structured category hierarchy from a set of resources (e.g, documents) reflecting abstraction of different levels.

One line of research explored traditional hierarchical clustering, either agglomerative or divisive, generating a tree-structured hierarchy by grouping documents according to a similarity measure [1, 2, 5, 13, 17].

Agglomerative algorithms [9, 11] build a hierarchy from bottom up by initially assigning each document to a cluster and merging the most similar pair of clusters at each step until only one left. The partitional algorithms [7] carries out top down with the most inclusive cluster, and then splits a least cohesive cluster at each step until it reaches the expected number of clusters. In most cases, partitional approaches are inferior to the agglomerative approaches in terms of clustering quality such as F-measure and Entropy measure [16].

There have been a large number of studies following this route to solve the problem of hierarchy generation. One representative study is to employ hierarchical agglomerative clustering (HAC) to build a hierarchy [1], where centroids of each category were used as initial seeds. However, this approach can only produce a binary tree and discard leaf categories with too few documents. The approach HAC+P was then proposed to overcome the problem by adding a post-processing min-max partition to change the binary tree into a multi-branches tree [5]. In min-max partition process, the hierarchy is recursively decomposed into sub-hierarchies by selecting the best level to minimize a criteria function that considers the cluster set quality and the cluster number preference. However, in most cases the criteria function is prone to the upper cut levels. The difficulties in setting too many parameters make this approach perplexed. A linear discriminant projection was used to transform all data into a lower dimensional space and HAC was used to generate a binary tree [13]. The binary tree is changed into a two-level hierarchy according to the cluster coherence. However, the rationality of the two-level hierarchy is unclear.

A parallel line of study explored the hierarchical probabilistic topic models, such as hLDA and nHDP, with the goal of learning a latent topic hierarchy from a corpus of documents. In such hierarchies, each internal node or topic reflects the shared terminology or vocabulary of the documents.

The hierarchical latent Dirichlet allocation (hLDA) model [8] is to learn a tree-structured topic hierarchy from a corpus of documents by placing a structure prior on possible hierarchies. In hLDA, the nested Chinese restaurant process (nCRP) is used as the nonparametric Bayesian prior. It is limited in that each document is generated from the topics on a single path of the tree. According to nCRP, hLDA first chooses a path for each document and then samples L-dimensional topic mixture proportions along the path from a Dirichlet distribution. Finally, it draws each word in the document from the L topics on the path from the root to a leaf. This single-path limitation has practical drawbacks in modelling cross-field documents with parallel topics, because hLDA restricts any two topics of a document must have a relationship that one topic is a subtopic of the other.

To overcome the limitations in hLDA, the most recent model nested hierarchical Dirichlet processes (nHDP) is

TABLE 3
COMPARISONS OF DIFFERENT METHODS WITH *AMHC*.

Aspects	Aggarwal 1999	Li 2007	Chuang 2004	Tang 2006	Yuan 2012	Our AMHC
Use initial hierarchy	NO	NO	NO	YES	YES	YES
Use new hierarchy measure (exclude traditional classification or cluster measures)	NO	YES	YES	NO	NO	YES
Use auxiliary hierarchy	NO	NO	NO	NO	YES	NO
Multi-way or Binary Tree	Binary Tree	Multi-Way	Multi-Way	Multi-Way	Multi-Way	Multi-Way
Multi-level or Two-level	Multi-level	Two-level	Multi-level	Multi-level	Multi-level	Multi-level
Change leaf categories	YES	NO	NO	NO	YES	YES

proposed in [19], which develops a new Bayesian nonparametric prior nHDP to replace nCRP providing uncertainty on possible tree structures. This new prior enables each word in a document to access to the entire tree rather than a single path, through associating each document a document-specific distribution on the paths within the tree.

However, the limitation of this kind of hierarchical topic models for hierarchy generation is that each internal node is a distribution of words across vocabulary and lacks interpretability. The word-distribution-represented topics are far from what we expect as a category in common sense. In hierarchical topic models, the internal nodes just reflect the co-occurrence patterns of words rather than the summarization of their children.

In short, hierarchical topic models are not suitable for directly constructing a reliable and satisfactory category hierarchy to organize and classify resources. It needs much more post-processing operations on the tree to transform it into a subject-based category hierarchy to become semantically meaningful.

Different from the hierarchy generation, hierarchy maintenance focuses on the modification of an existing hierarchy to make it better reflect the topics of its resources and achieve higher classification accuracy.

An approach to modifying a hierarchy using three operations (*Promote*, *Merge* and *Demote*) was proposed [22]. For each category, promote operation is tested, followed by merge and demote operations, in a top-down manner. The operation comes into effect if it can improve the classification accuracy. The approach iterates the process until no improvement can be observed. In experiments, this approach outperforms clustering-based hierarchy generation approach in terms of classification accuracy. However, there are two major problems. One is that this approach has a high time-complexity since it tests three operations on all nodes in the hierarchy. The other is that it retains less cohesive leaf categories, which will occur in most cases of real life applications.

A data-driven approach for hierarchy maintenance defines three operations (*Sprout*, *Merge* and *Assign*) with reference to an auxiliary hierarchy that covers a similar set of topics [23]. This approach can discover finer categories by projecting the documents in the given hierarchy to an auxiliary hierarchy. However, the discovery of some new topics depends on the auxiliary hierarchy which is not always easy to get, so in some cases this will become a limitation of this approach.

As for hierarchy evaluation, it is non-trivial for computers to simulate human evaluation method, judging whether the hierarchy taxonomy can reflect accurate classification semantics and keep balance among all branches and whether the resources are evenly distributed. Most of the current studies rely on F-measure, Precision and Recall to evaluate the hierarchical classification performance [21, 24]. However, these measures are not adequate to evaluate the quality of a hierarchy since they have totally ignored the impact of the structural balance and the resource distribution. For the hierarchy maintenance task, we need an evaluation measure that can be automatically computed with combining different aspects of a hierarchy in the united form. This is why we propose *UC_Score* in this paper.

In short, previous works paid little attention to the hierarchy maintenance and hierarchy evaluation.

In Table 3, different approaches are compared with our *AMHC* approach. Although our approach relys on HAC to generate a binary cluster tree to judge the similarity patterns of categories, our modified hierarchy is a multi-way tree that keeps similar levels of abstraction to the original hierarchy satisfying human understanding of taxonomy and bypass the problem in reference [13]. We can also solve the two problems proposed in reference [22] by adding a global modification phase that significantly speeds up the cross-branch movements of inappropriately located categories thus reducing the time-complexity. We split less cohesive leaf categories to overcome unchanged leaf categories. Compared to reference [23], we do not need an auxiliary hierarchy to discover new topics, since we use LDA topic model in the local phase to detect the topics and guide the *Merge*, *Pull-Up* and *Split* operations.

A systematic theory for defining and maintaining category hierarchy is needed for studying, system development and applications. An early hierarchical category formalism is the Resource Space Model (RSM), which coordinates multiple category hierarchies (or called classification trees) to form a hierarchical category space as a semantic model for specifying, storing, managing and retrieving various resources [26, 28, 31, 32]. Each dimension (axis) of the space is a hierarchy of categories (coordinates). Each point in the space is a fine category defined by corresponding coordinates at every axis. A point is a category that contains a set of resources. RSM is a way to manage big volume of resources through partitioning resources from multiple dimensions. The initial design of a space needs to be adapted to manage the constantly coming new resources, which may significantly influence existing categories and may form new categories. So the generation and maintenance of the category hierarchy is a part of the model [25, 27, 31]. This paper focuses on the maintenance of single category hierarchy (i.e., one dimensional category space), which has wide applications in efficiently managing online resources. More work on Resource Space Model is available at http://www.knowledgegrid.net/~h.zhuge/RSM.html.

5 Rationality

The evaluation of computing model is a basic problem of computing. There are three types of creators/users of a category hierarchy: system, human, and both system and human.

The proposed approach is suitable for maintaining the category hierarchy created and used by system, or created and used by non-expert people. For the category hierarchies defined and used by experts or by both machines and experts, the automatic adjustments can be regarded as recommendations, which enable experts to make adjustment decision quickly and easily.

This approach emphasizes objective patterns in resources while considering the initial views, which may be subjective (given by human).

The global phase relies on the clustering of resources. On one hand, the clustering process is to form a binary tree of resources, so it satisfies the representation of category hierarchy. On the other hand, hierarchical clustering approaches can provide reliable results with the average F-Score reaching 92.7% and 98.9% with agglomerative and partitional clustering methods respectively [34]. So, clustering approaches can satisfy human view of partitioning resources.

The local phase emphases on the detection of topic change within a category, which assumes that significant change of topics may happen with the coming of new resources, and topics can be used to stand for the pattern of resources belonging to a category. It is clear that if the topic of a category cannot cover the topic of its sub-categories, the subcategory needs adjusting.

Three measures category similarity, category cohesion and cover ratio based on the topic distributions from LDA topic model are used to trigger local adjustments. The topic distribution is more suitable to represent the pattern of resources within a category than keyword-based representations.

For completeness, the following assumptions need to be incorporated into our approach.

Assumption 1. The topics of the resources belonging to a category can well represent the category.

Assumption 2. The topics in the significant topic set of a subcategory also occur in the significant topic set of its parent category with higher probability than other topics.

6 Conclusion

This paper proposes an approach to maintaining category hierarchy through the analysis of the typical structures of inappropriate category hierarchies and making corresponding modification strategies. The approach incorporates the global-phase adjustments and the local-phase adjustments to significantly improve the topical cohesion and classification accuracy of categories. The classification semantics, structure balance and resources distribution are judged by the entropy-based measure UC_Score. A series of classification experiments demonstrates that the proposed approach is effective to generate topically cohesive hierarchies with better classification semantics.

The proposed approach can be used to maintain the category hierarchy for managing large, dynamic and diverse resources in cyberspace. It also provides a way to specialize the current online category hierarchy to organize resources with more specific categories.

Ongoing research is to realize automatic maintenance of the multiple category hierarchy, a multi-dimensional category space for manage small, medium and big data [29-31]. The challenge is the automatic discovery and coordination of dimensions according to the theory of the Resource Space Model and the Semantic Link Network model [33].

Acknowledgment

This work started in 2012 when the authors worked in China and finished in 2016 when the authors worked in the UK. The authors thank Aston University, Chinese Academy of Sciences and Nanjing University of Posts and Telecommunications for providing a cooperative research environment for this research. Research was supported by National Science Foundation of China and the research funding from Nanjing University of Posts and Telecommunications (NY212003).

References

[1] C. C. Aggarwal, S. C. Gates and P. S. Yu, On the merits of building categorization systems by supervised clustering. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (1999) 352-356.

[2] K.A. Alam, V. Chang, R. Ahmad, M. Tahir, A. Akhunzada and A.V. Vasilakos, Clustering and classification techniques for Web service discovery: a systematic review. *International Journal of Information Management*, (2016) 1-29.

[3] D. M. Blei, Probabilistic topic models. *Communications of the ACM*, 55(4) (2012) 77-8.

[4] C. C. Chang and C. J. Lin, LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *2*(3) (2011) 27.

[5] S. L. Chuang and L. F. Chien, A practical web-based approach to generating topic hierarchy for text segments. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*, (2004) 127-136.

[6] S. Dumais and H. Chen, Hierarchical classification of Web content. In *Proceedings of the 23rd annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, (2000) 256-263.

[7] I. S. Dhillon and D. S. Modha, Concept decompositions for large sparse text data using clustering. *Machine learning*, *42*(1-2) (2001)143-175.

[8] D. M. B. T. L. Griffiths and M. I. J. J. B. Tenenbaum, Hierarchical topic models and the nested Chinese restaurant process. *Advances in Neural Information Processing Systems*, 16 (2004) 17.

[9] S. Guha, R. Rastogi and K.Shim, CURE: an efficient clustering algorithm for large databases. In *ACM SIGMOD Record* , 27(2) (1998) 73-84.

[10] M. A. Kłopotek, Very large Bayesian multinets for text classification. *Future Generation Computer Systems*, 21(7) (2005) 1068-1082.

[11] G. Karypis, E. H. Han and V. Kumar, Chameleon: Hierarchical clustering using dynamic modeling. *Computer*, *32*(8) (1999) 68-75.

[12] T. Y. Liu, Y. Yang, H. Wan, H. J. Zeng, Z. Chen and W. Y. Ma, Support vector machines classification with a very large-scale taxonomy. *ACM SIGKDD Explorations Newsletter*, 7(1) (2005) 36-43.

[13] T. Li, S. Zhu and M. Ogihara, Hierarchical document classification using automatically generated hierarchy. *Journal of Intelligent Information Systems*, 29(2) (2007) 211-230.

[14] I. D. Melamed, Measuring semantic entropy. In *ACL-SIGLEX Workshop Tagging Text with Lexical Semantics: Why, What, and How* (1997) 4-5.

[15] T. Minka, Estimating a Dirichlet distribution. 2000.

[16] J. Puzicha, T. Hofmann and J. M. Buhmann, A theory of proximity based clustering: Structure detection by optimization. *Pattern Recognition*, *33*(4) (2000) 617-634.

[17] K. Punera, S. Rajan and J. Ghosh, Automatically learning document taxonomies for hierarchical classification. In *Special Interest Tracks and Posters of the 14th International Conference on World Wide Web*, (2005) 1010-1011.

[18] I. Porteous, D. Newman, A. Ihler, A. Asuncion, P. Smyth and M. Welling, Fast collapsed gibbs sampling for latent dirichlet allocation. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (2008) 569-577.

[19] J. Paisley, C.Wang, D. M. Blei and M. I. Jordan, Nested hierarchical Dirichlet processes. *Pattern Analysis and Machine Intelligence*, IEEE Transactions, 37(2) (2015) 256-270.

[20] C. E. Shannon, A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1) (2001) 3-55.

[21] A. Sun and E. P. Lim, Hierarchical text classification and evaluation. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference*, (2001) 521-528.

[22] L. Tang, J. Zhang and H. Liu, Acclimatizing taxonomic semantics for hierarchical content classification. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (2006) 384-393.

[23] Q. Yuan, G. Cong, A. Sun, C. Y. Lin and N. M. Thalmann, Category hierarchy maintenance: a data-driven approach. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, (2012) 791-800.

[24] Y. Yang and X. Liu, A re-examination of text categorization methods. In *Proceedings of the 22nd annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, (1999) 42-49.

[25] H. Zhuge, Resource space model, its design method and applications, *Journal of Systems and Software*, 72(1) (2004) 71-81.

[26] H. Zhuge, Resource space grid: model, method and platform. *Concurrency and Computation: Practice and Experience*, 16(14) (2004)1385-1413.

[27] H. Zhuge, *The Web Resource Space Model*. Springer. 2008.

[28] H. Zhuge, Y. Xing and P. Shi, Resource Space Model, OWL and Database: Mapping and Integration, *ACM Transactions on Internet Technology*, 8/4. 2008.

[29] H. Zhuge, The Complex Semantic Space Model, Keynote at 20th *IEEE International Conference on Collaboration Technologies and Infrastructures*, June 27th-29th, Paris, France (WETICE2011). 2011.

[30] H. Zhuge and Y. Xing, Probabilistic resource space model for managing resources in cyber-physical society. *IEEE Transactions on Services Computing*, 5(3) (2012)404-421.

[31] H. Zhuge, The Knowledge Grid, World Scientific. 2012 (2nd edition).

[32] H. Zhuge, Multi-Dimensional Summarization in Cyber-Physical Society, Elsevier, 2016.

[33] H. Zhuge and Y. Sun, The schema theory for semantic link network. *Future Generation Computer Systems*, 26(3) (2010) 408-420.

[34] Y. Zhao, G. Karypis, and U. Fayyad, Hierarchical clustering algorithms for document datasets. *Data Mining and Knowledge Discovery*, 10(2) (2005) 141-168.

Biography

Hai Zhuge is a Chair in Computer Science at Aston University in the UK, a joint professor of the Key Laboratory of Intelligent Information Processing in Chinese Academy of Sciences, and a guest professor of Nanjing University of Posts and Telecommunications in China. He has made systematic contribution to semantics modelling and knowledge management environment through lasting fundamental research on the semantic link network, the multi-dimensional category space and knowledge modelling. He is leading research towards Cyber-Physical Society, which concerns multi-disciplinary methodological, theoretical and technical innovation. Professor Zhuge is a Distinguished Scientist of the ACM (Association of Computer Machinery) for "Significant contribution and impact in computing field" and a Fellow of British Computer Society. He gave keynotes at many international conferences and invited lectures in many universities as a Distinguished Speaker of the ACM. He was a distinguished visiting fellow of Royal Academy of Engineering. He is serving as an associate editor of IEEE Intelligent Systems. Homepage: http://www.knowledgegrid.net/~h.zhuge. Email: haizhuge@gmail.com.

Lei He is a PhD student of the Key Lab of Intelligent Information Processing at the Institute of Computing Technology in Chinese Academy of Sciences. She is a research assistant of the University of Chinese Academy of Sciences and Nanjing University of Posts and Telecommunications. Her research interest is the automatic generation and maintenance of the Resource Space Model.