Integrating Web Directories by Learning their Structures

Christopher C. Yang
The Chinese University of Hong Kong
yang@se.cuhk.edu.hk

Jianfeng Lin
The Chinese University of Hong Kong
jflin@se.cuhk.edu.hk

ABSTRACT

Documents in the Web are often organized using category trees by information providers (e.g. CNN, BBC) or search engines (e.g. Google, Yahoo!). Such category trees are commonly known as Web directories. The category tree structures from different internet content providers may be similar to some extent but are usually not exactly the same. As a result, it is desirable to integrate these category trees together so that web users only need to browse through a unified category tree to extract information from multiple providers. In this paper, we address this problem by capturing structural information of multiple category trees, which are embedded with the knowledge of professional in organizing the documents. Our experiments with real Web data show that the proposed technique is promising.

Categories and Subject Descriptors
H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – *Retrieval models, Search process*

General Terms: Algorithms, Experimentation, Theory

Keywords: Category tree, Integration, Hierarchical structure

1. INTRODUCTION

Category trees and web directory are often used to organize information in the web. However, different participants on the internet often construct and maintain different category trees with different structures to facilitate the organization and retrieval of information. For example, Yahoo! and Google have edited their directories in their own way. Category tree integration arises in a variety of situations, ranging from B2B and B2C e-business, personal information management, supply chain management etc.

The number of category trees on the internet is so large that manual integration is tedious, error-prone or even impossible. Previous work about data sharing and integration mainly focus on ontology integration [2][3] and schema matching [4], but ontology and schema capture the structure of specific semantic. On the other hand, category trees capture the parent and child relationship between topics of documents. Agrawal and Srikant [1] first explore how to integrate web categories trees. Unfortunately, such approach only considers flat hierarchical structures where documents are assigned to the leave nodes. It does not take the hierarchical structure of a category tree into consideration during category tree integration. Our approach captures the knowledge of category tree structures that are generated by information professionals in the process of integrating category trees. The contributions can be summarized as:

1) Extend the Bäyes rule to determine the category relationship between categories from different category trees.
2) Develop four decision rules to map a category from the source category tree to a category in the master category tree.
3) Develop an integration technique that satisfies the constraints imposed by the structures of the source category trees.
4) The integration technique is able to expand or modify the master category tree by learning the organization of documents from the source category trees.

2. PROBLEM DEFINITION

For category tree integration problem, There exists a source category tree $T^s = \{C_1^s, C_2^s, ..., C_{|T^s|}^s\}$ and a master category tree $T^m = \{C_1^m, C_2^m, ..., C_{|T^m|}^m\}$. Both trees have a set of categories with certain hierarchical structure, and a set of documents are assigned to each category. The only relationship between these categories within a tree is the subsumption relationship between parents and the children. When the source category tree is integrated with the master category tree, two integration operators can be applied to the category of the source category tree, C_i^s:

- *Map*: C_i^s may be mapped to a existing category in the master category tree, C_j^m, noted as $Map(C_i^s; C_j^m)$; or
- *Add*: C_i^s may be mapped to an expanded category in the master category tree, $C_{|T^m|+1}^m$, noted as $Add(C_i^s; C_{new}^m, C_{parent}^m, C_{child}^m)$, where C_{parent}^m is the parent of C_{new}^m and C_{child}^m is the child of C_{new}^m; if C_{child}^m is omitted, C_{new}^m is add as a leaf category

At the current stage of research, we do not consider *splitting* or *merging* nodes. However, we shall investigate splitting when the source node can be split to map with two master nodes and merging when two source nodes can be merged to map with one master nodes in our future work.

3. INTEGRATION TECHNIQUES

3.1 Category Relationships

The mapping algorithm is based on the relationships between categories in the master and source category tree. We adopt the Bäyes rule *P(A|B)*, to determine the category relationships [5].

$$P(A \mid B) = \frac{number\ of\ documents\ in\ B\ predicted\ to\ be\ in\ A}{number\ of\ documents\ in\ B}$$

We identify 5 types of relationships and the relationships between categories, and they should be determined as follows:

- $Match(A,B)$: $P(A\mid B) \ge th_H \wedge P(B\mid A) \ge th_H$
- $Disjoint(A,B)$: $P(A\mid B) \le th_L \wedge P(B\mid A) \le th_L$
- $Subconcept(A,B)$: $P(A\mid B) < th_H \wedge P(B\mid A) \ge th_H$
- $Superconcept(A,B)$: $P(A\mid B) \ge th_H \wedge P(B\mid A) < th_H$
- $Overlap(A,B)$: $th_L < P(A\mid B) < th_H \wedge 0 < P(B\mid A) \le th_H$ or $0 < P(A\mid B) < th_H \wedge th_L < P(B\mid A) < th_H$

th_H and th_L are the parameters. In the ideal case, they should be 1 and 0 respectively, but in real system, they are usually a little smaller or larger than 1 or 0.

3.2 Decision Rules and algorithm

Rule learning methods usually attempt to select the best from all possible covering rules according to some minimality criterions. In this work, we develop four rules to integrate categories. The objectives of the rules are to maintain the structure of the source category trees while integrating with the master category tree.

Rule 1 (Figure 1, Left): Maintaining Parent-Child Relationship Given $Map(S_j, M_i)$, $S_j^{'} \in Child(S_j)$ and $M_i^{'} \in Descedant(M_i)$, if $Match(S_j^{'}, M_i^{'})$, then $Map(S_j^{'}, M_i^{'})$.

Rule 2 (Figure 1, Right): Expanding With A New Branch Given $Map(S_j, M_i)$, $S_j^{'} \in Child(S_j)$, if $S_j^{'}$ disjoint with $\forall M_j \in Decencent(M_i)$, $Disjoint(S_j^{'}, M_j)$, then $Add(S_j^{'}; M_{new}, M_i)$, and all the descendants of $S_j^{'}$ is also added as the descendants of $S_j^{'}$

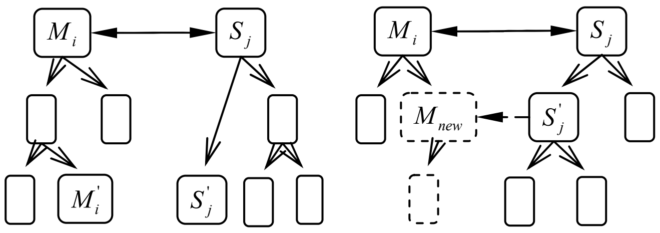

Figure 1. Rule 1, Rule 2

Rule 3 (Figure 2, Left): Expanding As A Subconcept Given $Map(S_j, M_i)$, $S_j^{'} \in Child(S_j)$ and $M_i^{'} \in Descedant(M_i)$, if $SubConcept(S_j^{'}, M_i^{'})$, then $Add(S_j^{'}; M_{new}, M_i^{'})$.

Rule 4 (Figure 2, Right): Expanding As A Superconcept Given $Map(S_j, M_i)$, $S_j^{'} \in Child(S_j)$ and $M_i^{'} \in Descedant(M_i)$, if $SuperConcept(S_j^{'}, M_i^{'})$, then $Add(S_j^{'}; M_{new}, M_i, M_i^{'})$.

Rule 1-4 are the basic rules of category integration, but there is a shortcoming of rule 2. When a category, say $S_j^{'}$, is incorrectly mapped to an expanded category M_{new}, all its descendants will also have incorrect mappings. We develop some adjustment rules in these cases. The correct position of $S_j^{'}$ will be further affected by its descendants. Our experiments show that this rule is useful.

We adopt a top down, level based method to run the algorithm. The nodes are processed in breadth first order. In this process, when given a node S_j, we will find one rule out of four to fire based on the condition of the rules.

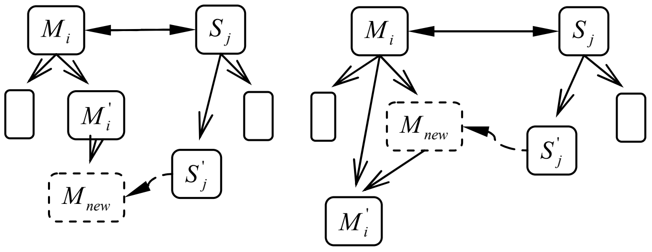

Figure 2. Rule 3, Rule 4

4. EXPERIMENTS

We collect 10 data sets as experiment data from Yahoo! and Open Directory Project. Each data set consists of two category trees, one from Yahoo! and one from Open Directory Project, serving as source and master category tree respectively. The root nodes of the two category trees match with each other. They rooted at “Science”, “Shopping” and “Society”. The average number of categories in source and master category trees are 7.1 and 9.7 respectively. The “number of documents in B predicted to be A” in Section 3.1 is decided by an automatic text classifier. We use SVM^{light} developed by Joachims [6], which is a fast and effective implementation of SVM.

We measure the accuracy of the integration result by measuring how many categories are correctly processed. Our experiment based on the ten datasets show that it obtains 85% accuracy on average, and two of ten data sets even reach 100% accuracy.

5. CONCLUSION

In this work, we explore how to make use of implicit information embedded in the hierarch category tree structure to integrate different category trees in this paper. Nodes in one category tree are mapped or inserted to proper position of the other. We use real world data to conduct our experiment and get good result. For simplicity, we omit nodes splitting or merging problem in this poster, which is also very important and hard. We will extend our techniques to handle this work in our future research.

6. REFERENCES

[1] R. Agrawal and R. Srikant. On Integrating Catalogs. In proceedings of WWW10 Conference, May 1-5, 2001, Hong Kong, pp 603-612

[2] H. Li and K. Yamanishi, Text Classification using ESC-Based Stochastic Decision Lists, In CIKM , Kansac City, Mo, UAS, 1999.

[3] N.F. Noy and M.A. Musen. Anchor-PROMPT: Using Non-Local Context for Semantic Matching. In Proceedings of workshop on OIS at IJCAI, 2001

[4] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. The VLDB Journal 10(4):334-350, 2001

[5] S. Zhu, C.C. Yang and W. Lam. CatRelate: A New Hierarchical Document Category Integration Algorithm by Learning Category Relationship. ICADL 2004, LNCS 3334, pp. 280-289, 2004

[6] T. Joachims, Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning, B. Scholkopf and C. Burges and A. Smola (ed.), MIT-Press, 1999.