Web Taxonomy Integration through Co-Bootstrapping

Dell Zhang[1,2]
[1]Department of Computer Science
School of Computing
S15-05-24, 3 Science Drive 2
National University of Singapore
Singapore 117543
[2]Singapore-MIT Alliance
E4-04-10, 4 Engineering Drive 3
Singapore 117576
+65-68744251
dell.z@ieee.org

Wee Sun Lee[1,2]
[1]Department of Computer Science
School of Computing
SOC1-05-26, 3 Science Drive 2
National University of Singapore
Singapore 117543
[2]Singapore-MIT Alliance
E4-04-10, 4 Engineering Drive 3
Singapore 117576
+65-68744526
leews@comp.nus.edu.sg

ABSTRACT

We address the problem of integrating objects from a source taxonomy into a master taxonomy. This problem is not only currently pervasive on the web, but also important to the emerging semantic web. A straightforward approach to automating this process would be to learn a classifier that can classify objects from the source taxonomy into categories of the master taxonomy. The key insight is that the availability of the source taxonomy data could be helpful to build better classifiers for the master taxonomy if their categorizations have some semantic overlap. In this paper, we propose a new approach, co-bootstrapping, to enhance the classification by exploiting such implicit knowledge. Our experiments with real-world web data show substantial improvements in the performance of taxonomy integration.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications – *data mining*; H.2.5 [**Database Management**]: Heterogeneous Databases; I.2.6 [**Artificial Intelligence**]: Learning; I.5.2 [**Pattern Recognition**]: Design Methodology – *classifier design and evaluation*.

General Terms

Algorithms, Experimentation, Theory.

Keywords

Semantic Web, Ontology Mapping, Taxonomy Integration, Machine Learning, Classification, Bootstrapping, Boosting.

1. INTRODUCTION

A taxonomy, or directory or catalog, is a division of a set of objects (documents, images, products, goods, services, etc.) into a set of categories. There are a tremendous number of taxonomies on the web, and we often need to integrate objects from a source taxonomy into a master taxonomy.

This problem is currently pervasive on the web, given that many websites are aggregators of information from various other websites [2]. A few examples will illustrate the scenario. A web marketplace like Amazon[1] may want to combine goods from multiple vendors' catalogs into its own. A web portal like NCSTRL[2] may want to combine documents from multiple libraries' directories into its own. A company may want to merge its service taxonomy with its partners'. A researcher may want to merge his/her bookmark taxonomy with his/her peers'. Singapore-MIT Alliance[3], an innovative engineering education and research collaboration among MIT, NUS and NTU, has a need to integrate the academic resource (courses, seminars, reports, softwares, etc.) taxonomies of these three universities.

This problem is also important to the emerging semantic web [4], where data has structures and ontologies describe the semantics of the data, thus better enabling computers and people to work in cooperation. On the semantic web, data often come from many different ontologies, and information processing across ontologies is not possible without knowing the semantic mappings between them. Since taxonomies are central components of ontologies, ontology mapping necessarily involves finding the correspondences between two taxonomies, which is often based on integrating objects from one taxonomy into the other and vice versa [10, 14].

If all taxonomy creators and users agreed on a universal standard, taxonomy integration would not be so difficult. But the web has evolved without central editorship. Hence the correspondences between two taxonomies are inevitably noisy and fuzzy. For illustration, consider the taxonomies of two web portals Google[4] and Yahoo[5]: what is "Arts/Music/Styles/" in one may be "Entertainment/Music/Genres/" in the other, category "Computers_and_Internet/Software/Freeware" and category

[1] http://www.amazon.com/

[2] http://www.ncstrl.org/

[3] http://web.mit.edu/sma/

[4] http://www.google.com/

[5] http://www.yahoo.com/

"Computers/Open_Source/Software" have similar contents but show non-trivial differences, and so on. It is unclear if a universal standard will appear outside specific domains, and even for those domains, there is a need to integrate objects from legacy taxonomy into the standard taxonomy.

Manual taxonomy integration is tedious, error-prone, and clearly not possible at the web scale. A straightforward approach to automating this process would be to formulate it as a classification problem which has being well-studied in machine learning area [18]. Normally the classifier would be constructed using objects in the master taxonomy as training examples, and the source taxonomy would be completely ignored during learning. However, the availability of the source taxonomy data could be helpful to build better classifiers for the master taxonomy if their categorizations have some semantic overlap, particularly when the number of training examples is not very large.

Possible useful semantic relationships between a master category C and a source category S include:

- $C = S$ (identical): an object belongs to C if and only if it belongs to S;
- $C \bigcap S = \varnothing$ (mutual exclusion): if an object belongs to S it cannot belong to C;
- $C \supseteq S$ (superset): any object that belonging to S must also belong to C;
- $C \subseteq S$ (subset): any object not belonging to S also cannot belong to C;
- C and S overlap but neither is a superset of the other.

In addition, semantic relationships may involve multiple master and source categories. For example, a master category C may be a subset of the union of two source categories S_a and S_b, so if an object does not belong to either S_a or S_b, it cannot belong to C. The real-world semantic relationships are noisy and fuzzy, but they can still provide valuable information for classification. For example, knowing that most (80%) objects in a source category S belong to one master category C_a and the rest (20%) examples belong to another master category C_b is obviously helpful. The difficulty is that knowledge about those semantic relationships is not explicit but hidden in the data.

In this paper, we propose a new approach, co-bootstrapping, to enhance the classification by exploiting such implicit knowledge. Our experiments with real-world web data show substantial improvements in the performance of taxonomy integration.

The rest of this paper is organized as follows. In §2, we give the formal problem statement. In §3, we describe a state-of-the-art solution. In §4, we present our approach in detail. In §5, we conduct experimental evaluations. In §6, we review the related work. In §7, we make concluding remarks.

2. PROBLEM STATEMENT

Taxonomies are often organized as hierarchies. In this work, we assume for simplicity, that any objects assigned to an interior node really belong to a leaf node which is an offspring of that interior node. Since we now have all objects only at leaf nodes, we can flatten the hierarchical taxonomy to a single level and treat it as a set of categories [2].

Now we formally define the taxonomy integration problem that we are solving. Given two taxonomies:

- a master taxonomy $\mathcal{M}$ with a set of categories $C_1, C_2, ..., C_M$ each containing a set of objects, and
- a source taxonomy $\mathcal{N}$ with a set of categories $S_1, S_2, ..., S_N$ each containing a set of objects,

we need to find the categories in $\mathcal{M}$ for each object in $\mathcal{N}$.

To formulate taxonomy integration as a classification problem, we take $C_1, C_2, ..., C_M$ as classes, the objects in $\mathcal{M}$ as training examples, the objects in $\mathcal{N}$ as test examples, so that taxonomy integration can be automatically accomplished by predicting the classes of each test example. Such a classification problem is multi-class and multi-label, in the sense that there are usually more than two possible classes and one object may be relevant to more than one class.

3. A STATE-OF-THE-ART SOLUTION

Agrawal and Srikant recently proposed an elegant approach to taxonomy integration by enhancing the Naïve Bayes algorithm [2].

The Naïve Bayes (NB) algorithm is a well-known text classification technique [18]. NB tries to fit a generative model for documents using training examples and apply this model to classify test examples. The generative model of NB assumes that a document is generated by first choosing its class according to a prior distribution of classes, and then producing its words independently according to a (typically multinomial) distribution of terms conditioned on the chosen class [15]. Given a test document d, NB predicts its class to be $\arg\max_C \Pr[C \mid d]$. The posterior probability $\Pr[C \mid d]$ can be computed via Bayes's rule:

$$\Pr[C \mid d] = \frac{\Pr[C, d]}{\Pr[d]} = \frac{\Pr[C]\Pr[d \mid C]}{\Pr[d]} \propto \Pr[C]\Pr[d \mid C]$$
$$= \Pr[C]\prod_{w \in d} \left(\Pr[w \mid C]\right)^{n(d,w)},$$

where $n(d, w)$ is the number of occurrences of w in d. The probability $\Pr[C]$ can be estimated by the proportion of training documents in C. The probability $\Pr[w \mid C]$ can be estimated by $\frac{n(C, w) + \eta}{\sum_{w_i \in V}\left(n(C, w_i) + \eta\right)}$, where $n(C, w)$ is the number of occurrences of w in training documents in C, V is the vocabulary of terms, and $0 < \eta \le 1$ is the Lidstone's smoothing parameter [1]. Taking logs, we see that NB is actually a linear classifier:

$$\log \Pr[C \mid d] \propto \log\left(\Pr[C]\prod_{w \in d}\left(\Pr[w \mid C]\right)^{n(d,w)}\right)$$
$$= \sum_{w \in d}\left(n(d, w) \times \log \Pr[w \mid C]\right) + \log \Pr[C].$$

The enhanced Naïve Bayes (ENB) algorithm [2] uses the categorization of the source taxonomy to get better probability estimations. Given a test document d that is know to be in category S in $\mathcal{N}$, ENB predicts its category in $\mathcal{M}$ to be $\arg\max_C \Pr[C \mid d, S]$. The posterior probability $\Pr[C \mid d, S]$ can be computed as $\Pr[C \mid d,S] = \frac{\Pr[C,d,S]}{\Pr[d,S]} = \frac{\Pr[S]\Pr[C,d \mid S]}{\Pr[d,S]}$ $\propto \Pr[C,d \mid S]$. ENB invokes a simplification that assumes d and S are independent given C, therefore

$$\Pr[C,d \mid S] = \Pr[C \mid S]\Pr[d \mid S,C] = \Pr[C \mid S]\Pr[d \mid C] = \Pr[C \mid S]\prod_{w \in d}\left(\Pr[w \mid C]\right)^{n(d,w)}.$$

The probability $\Pr[w \mid C]$ can be estimated in the same way of NB. For the probability $\Pr[C \mid S]$, ENB estimates it by $\frac{|C| \times |C \leftarrow S|^{\omega}}{\sum_{C_i}\left(|C_i| \times |C_i \leftarrow S|^{\omega}\right)}$, where $|C|$ is the number of documents in C, $|C \leftarrow S|$ is the number of documents in S classified into C by the NB classifier, and $\omega \geq 0$ is a parameter reflecting the degree of semantic overlap between the categorizations of $\mathcal{M}$ and $\mathcal{N}$. The optimal value of ω can be found using a tune set (a set of objects whose categories in both taxonomies are known). The tune set can be made available via random sampling or active learning [2]. Taking logs, we see that ENB is still a linear classifier:

$$\log \Pr[C \mid d,S] \propto \log\left(\Pr[C \mid S]\prod_{w \in d}\left(\Pr[w \mid C]\right)^{n(d,w)}\right)$$
$$= \sum_{w \in d}\left(n(d,w) \times \log \Pr[w \mid C]\right) + \log \Pr[C \mid S].$$

Comparing the classification functions of NB and ENB, it is obvious that all ENB does is to shift the classification threshold of its base NB classifier, no more and no less.

To achieve multi-class multi-label classification that is required by taxonomy integration, we use the "one-vs-rest" method to create an ensemble of binary (yes/no) NB or ENB classifiers, one for each category C in $\mathcal{M}$.

4. OUR APPROACH

Here we present our approach in detail. In §4.1, we introduce the boosting technique. In §4.2, we propose the co-bootstrapping method. In §4.3, we discuss the advantages of our approach.

4.1 Boosting

In our approach to taxonomy integration, we utilize a powerful machine learning method, boosting [17, 23], to build classifiers. The main idea of boosting is to combine many weak hypotheses (simple and moderately accurate classification rules), into a highly accurate classifier. In this paper, we focus on boosting for text classification. Generalization to other kinds of data and learning algorithms would be straightforward.

4.1.1 Term-Features

Text objects (documents) can be represented using a set of term-features $F_T = \{f_{T1}, f_{T2}, \ldots f_{Tn}\}$. The term-feature f_{Th} $(1 \leq h \leq n)$ of a given object x is a binary feature indicating the presence or absence of w_h (the *h*-th distinct word in the document collection) in x, i.e., $f_{Th} = \begin{cases} 1 & \text{if } w_h \in x \\ 0 & \text{if } w_h \notin x \end{cases}$.

4.1.2 Weak Hypotheses

Let $\mathcal{X}$ denote the domain of possible objects, and let $\mathcal{Y}$ be a set of k possible classes. A labeled example is a pair (x,Y) where $x \in \mathcal{X}$ is an object and $Y \subseteq \mathcal{Y}$ is the set of classes which x belongs to. We define $Y[l]$ for $l \in \mathcal{Y}$ to be $Y[l] = \begin{cases} +1 & \text{if } l \in Y \\ -1 & \text{if } l \notin Y \end{cases}$.

A hypothesis is a real-valued function $h: \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$. The sign of $h(x,l)$ is a prediction of $Y[l]$ for x, i.e., whether object x is contained in class l. The magnitude of $h(x,l)$ is interpreted as a measure of confidence in the prediction.

Based on a binary feature f, we are interested in weak hypotheses h which are simple decision stumps of the form $h(x,l) = \begin{cases} c_{1l} & \text{if } f = 1 \\ c_{0l} & \text{if } f = 0 \end{cases}$, where $c_{1l}, c_{0l} \in \mathbb{R}$.

4.1.3 AdaBoost Algorithm

The most popular boosting algorithm is AdaBoost introduced in 1995 by Freund and Schapire [12]. Our work is based on a multi-class multi-label version of AdaBoost, AdaBoost.MH [24, 25], which is described in Figure 1.

```
Given: $(x_1,Y_1),\ldots,(x_m,Y_m)$ where each $x_i \in \mathcal{X}$, $Y_i \in \mathcal{Y}$.
Initialize $D_1(i,l) = 1/(mk)$.
for $t = 1,\ldots,T$ do
    Pass distribution $D_t$ to weak learner.
    Get weak hypothesis $ht: \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$.
    Choose $\alpha_t \in \mathbb{R}$.
    Update:
        $D_{t+1}(i,l) = \frac{D_t(i,l)\exp(-\alpha_t Y_i[l] h_t(x_i,l))}{Z_t}$
        where $Z_t$ is the normalization factor
end for
Output the final hypothesis:
    $H(x,l) = \sum_{t=1}^{T} \alpha_t h_t(x,l)$.
```

Figure 1: The boosting algorithm AdaBoost.MH.

Given m training examples $(x_1,Y_1),\ldots,(x_m,Y_m)$ where each $x_i \in \mathcal{X}$, $Y_i \in \mathcal{Y}$, AdaBoost.MH dynamically maintains a distribution D_t over all objects and classes. Initially this distribution D_1 is uniform. In the *t*-th round, the optimal weak hypothesis h_t is selected based on the set of training examples and the current distribution D_t. Then a parameter α_t is chosen, and the distribution D_t is updated in a manner that puts more weights on "difficult" examples (object-class pairs) that are misclassified by h_t. Please be referred to [24, 25] for the details on computing optimal h_t and α_t. This procedure repeats for T rounds. The final hypothesis $H(x,l)$ is actually a weighted vote of weak hypotheses $\sum_{t=1}^{T}\alpha_t h_t(x,l)$, and the final prediction can be computed according to the sign of $H(x,l)$.

4.2 Co-Bootstrapping

Thus far we have completely ignored the categorization of $\mathcal{N}$. Although $\mathcal{M}$ and $\mathcal{N}$ are usually not identical, their categorizations often have some semantic overlap. Therefore the categorization of $\mathcal{N}$ contains valuable implicit knowledge about the categorization of $\mathcal{M}$. Hereby we propose a new approach, co-bootstrapping, to enhance the classification by exploiting such implicit knowledge.

4.2.1 Category-Features

If we have indicator functions for each category in $\mathcal{N}$, we can imagine taking those indicator functions as features when we learn the classifier for $\mathcal{M}$. This allows us to exploit the semantic relationship among the categories of $\mathcal{M}$ and $\mathcal{N}$ without explicitly figuring out what the semantic relationships are. More specifically, for each object in $\mathcal{M}$, we augment the ordinary term-features with a set of category-features $F_{\mathcal{N}} = \{f_{\mathcal{N}1}, f_{\mathcal{N}2}..., f_{\mathcal{N}N}\}$ derived from $\mathcal{N}$. The category-feature $f_{\mathcal{N}j}$ $(1 \le j \le N)$ of a given object x is a binary feature indicating whether x belongs to category S_j (the *j*-th category of $\mathcal{N}$), i.e., $f_{\mathcal{N}j} = \begin{cases} 1 & \text{if } x \in S_j \\ 0 & \text{if } x \notin S_j \end{cases}$.

In the same way, we can get a set of category-features $F_{\mathcal{M}} = \{f_{\mathcal{M}1}, f_{\mathcal{M}2}..., f_{\mathcal{M}M}\}$ derived from $\mathcal{M}$ to be used for supplementing the features of objects in $\mathcal{N}$. The remaining problem is to obtain these indicator functions, which are initially not available.

4.2.2 Co-Bootstrapping Algorithm

When building the classifier for $\mathcal{M}$, the training examples are the objects in $\mathcal{M}$ and the test examples are the objects in $\mathcal{N}$. To leverage the categorization of $\mathcal{N}$ to reinforce classification, our classifier uses term-features F_T as well as category-features $F_{\mathcal{N}}$. However, we do not know the exact values of $F_{\mathcal{N}}$ of the training examples.

Our proposed algorithm overcomes the above obstacle by utilizing the bootstrapping idea. Let $\boldsymbol{B}_r^{\mathcal{T}}(F)$ denote a boosting-classifier for taxonomy $\mathcal{T}$'s categorization based on feature set F at step r. Initially we build a classifiers $\boldsymbol{B}_0^{\mathcal{N}}(F_T)$ based on only term-features, then use it to classify the objects in $\mathcal{M}$ (the training examples) into the categories of $\mathcal{N}$, thus we can predict the value of each category-feature $f_{\mathcal{N}j} \in F_{\mathcal{N}}$ for each object $x \in \mathcal{M}$. At next step we will be able to build $\boldsymbol{B}_1^{\mathcal{M}}(F_T \cup F_{\mathcal{N}})$ using the predicted values of $F_{\mathcal{N}}$ of the training examples. Similarly we can build $\boldsymbol{B}_0^{\mathcal{M}}(F_T)$ and $\boldsymbol{B}_1^{\mathcal{N}}(F_T \cup F_{\mathcal{M}})$. The new classifier $\boldsymbol{B}_1^{\mathcal{N}}(F_T \cup F_{\mathcal{M}})$ ought to be better than $\boldsymbol{B}_0^{\mathcal{N}}(F_T)$ because $\boldsymbol{B}_1^{\mathcal{N}}(F_T \cup F_{\mathcal{M}})$ leverages more knowledge. Hence we can predict the value of each category-feature $f_{\mathcal{N}j} \in F_{\mathcal{N}}$ for each object $x \in \mathcal{M}$ more accurately using $\boldsymbol{B}_1^{\mathcal{N}}(F_T \cup F_{\mathcal{M}})$ instead of $\boldsymbol{B}_0^{\mathcal{N}}(F_T)$, and afterwards we can build $\boldsymbol{B}_2^{\mathcal{M}}(F_T \cup F_{\mathcal{N}})$. Also $\boldsymbol{B}_2^{\mathcal{M}}(F_T \cup F_{\mathcal{N}})$ is very likely to be better than $\boldsymbol{B}_1^{\mathcal{M}}(F_T \cup F_{\mathcal{N}})$ because $\boldsymbol{B}_2^{\mathcal{M}}(F_T \cup F_{\mathcal{N}})$ is based on a more accurate prediction of $F_{\mathcal{N}}$. This process can be repeated iteratively in a "ping-pong" manner. We name this approach co-bootstrapping since the two classifiers $\boldsymbol{B}_r^{\mathcal{M}}(F_T \cup F_{\mathcal{N}})$ and $\boldsymbol{B}_r^{\mathcal{N}}(F_T \cup F_{\mathcal{M}})$ collaborate to bootstrap themselves together. Figure 2 presents the co-bootstrapping algorithm, and Figure 3 depicts its process.

4.3 Discussion

4.3.1 Why Choose Boosting

We have selected to employ the boosting technique to build classifiers in our co-bootstrapping approach to taxonomy integration because of its following virtues.

- Boosting has shown outstanding classification performance on many kinds of data such as text documents [17, 23, 24].
- Boosting finds the optimal combination of heterogeneous weak hypotheses automatically, therefore alleviates the problem of how to weight ordinary features (e.g. term-features) and category-features appropriately. In contrast, approaches based on other machine learning algorithms like Support Vector Machines (SVMs) [9] would require to adjust relative combination weights, which is a non-trivial problem.
- Boosting generates descriptive and human-readable hypotheses as the final classifier, and the learned classifier is usually sparse despite the large feature set.

Although boosting looks an ideal choice, other machine learning algorithms can also be utilized in the co-bootstrapping approach. We have not investigated this issue yet.

4.3.2 Comparison with ENB

Although ENB [2] has been shown to work well for taxonomy integration, we think that a more general approach is still attractive. It has been experimentally shown that AdaBoost is more promising than NB for text classification [24]. The co-bootstrapping approach allows more powerful machine learning algorithms like AdaBoost to be utilized.

Both ENB and our co-bootstrapping approach exploit the categorization of $\mathcal{N}$ to enhance classification. While all ENB does is to shift the classification threshold of its base NB classifier (see §3), co-bootstrapping has the ability to achieve more complex adjustments on the classification function of its base classifier.

Furthermore, ENB needs a stand-alone tune set to find the optimal value of parameter ω which controls the influence of source categorization information on classification, whereas co-bootstrapping based on boosting does not have such burdens.

Although co-bootstrapping looks more effective, ENB still holds an advantage in efficiency.

```
Given: two taxonomies $\mathcal{M}$ and $\mathcal{N}$.
• Build classifier $B_0^{\mathcal{M}}(F_T)$, then use it to predict the
  value of each category-feature $f_{\mathcal{M}i} \in F_{\mathcal{M}}$ for each
  object $x \in \mathcal{N}$.
• Build classifier $B_0^{\mathcal{N}}(F_T)$, then use it to predict the
  value of each category-feature $f_{\mathcal{N}j} \in F_{\mathcal{N}}$ for each
  object $x \in \mathcal{M}$.
for $r = 1,\ldots,R$ do
    • Build classifier $B_r^{\mathcal{M}}(F_T \cup F_{\mathcal{N}})$, then use it to
      predict the value of each category-feature
      $f_{\mathcal{M}i} \in F_{\mathcal{M}}$ for each object $x \in \mathcal{N}$.
    • Build classifier $B_r^{\mathcal{N}}(F_T \cup F_{\mathcal{M}})$, then use it to
      predict the value of each category-feature
      $f_{\mathcal{N}j} \in F_{\mathcal{N}}$ for each object $x \in \mathcal{M}$.
end for
• For each object $x \in \mathcal{N}$, if the value of its category-
  feature $f_{\mathcal{M}i} \in F_{\mathcal{M}}$ is positive, then we classify it
  into $C_i \in \mathcal{M}$.
• For each object $x \in \mathcal{M}$, if the value of its category-
  feature $f_{\mathcal{N}j} \in F_{\mathcal{N}}$ is positive, then we classify it
  into $S_j \in \mathcal{N}$.
```

Figure 2: The co-bootstrapping algorithm.

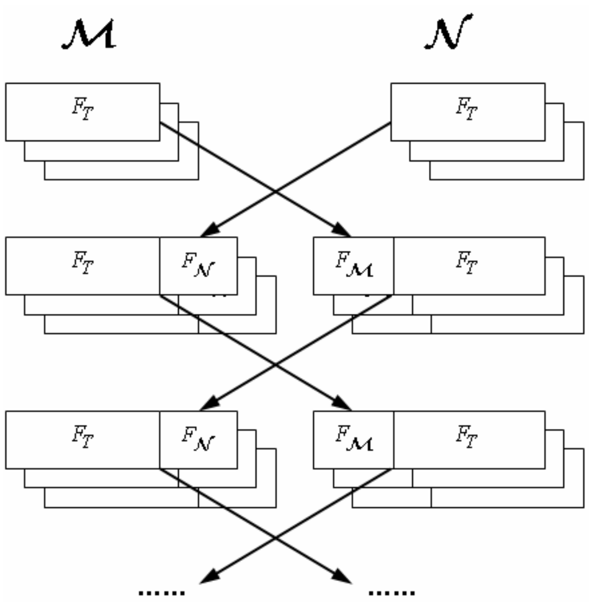

Figure 3: The co-bootstrapping process.

5. EXPERIMENTS

5.1 Datasets

We have collected 5 datasets from Google and Yahoo. One dataset includes the slice of Google's taxonomy and the slice of Yahoo's taxonomy about websites on one specific topic, as shown in Table 1.

Table 1: The datasets.

	Google	Yahoo
Book	/ Top/ Shopping/ Publications/ Books/	/ Business_and_Economy/ Shopping_and_Services/ Books/ Bookstores/
Disease	/ Top/ Health/ Conditions_and_Diseases/	/ Health/ Diseases_and_Conditions/
Movie	/ Top/ Arts/ Movies/ Genres/	/ Entertainment/ Movies_and_Film/ Genres/
Music	/ Top/ Arts/ Music/ Styles/	/ Entertainment/ Music/ Genres/
News	/ Top/ News/ By_Subject/	/ News_and_Media/

In each slice of taxonomy, we take only the top level directories as categories, e.g., the "Movie" slice of Google's taxonomy has categories like "Action", "Comedy", "Horror", etc.

Table 2: The number of categories.

	Google	Yahoo
Book	49	41
Disease	30	51
Movie	34	25
Music	47	24
News	27	34

For each dataset, we show in Table 2 the number of categories occurred in Google and Yahoo respectively.

In each category, we take all items listed on the corresponding directory page and its sub-directory pages as its objects. An object (list item) corresponds to a website on the world wide web, which is usually described by its URL, its title, and optionally a short annotation about its content. Here each object is considered as a text document composed of its title and annotation. All documents are pre-processed by removal of stop-words and stemming.

Table 3: The number of objects.

	Google	Yahoo	$G \cup Y$	$G \cap Y$
Book	10,842	11,268	21,111	999
Disease	34,047	9,785	41,439	2,393
Movie	36,787	14,366	49,744	1,409
Music	76,420	24,518	95,971	4,967
News	31,504	19,419	49,303	1,620

For each dataset, we show in Table 3 the number of objects occurred in Google (G), Yahoo (Y), either of them ($G \cup Y$), and both of them ($G \cap Y$) respectively. The set of objects in $G \cap Y$ covers only a small portion (usually less than 10%) of the set of objects in Google or Yahoo alone, which suggests the great benefit of automatically integrating them. This observation is consistent with [2].

The number of categories per object in these datasets is 1.54 on average. This observation justifies the necessity of building multi-class multi-label classifiers.

5.2 Tasks

For each dataset, we pose 2 symmetric taxonomy integration tasks: G←Y (integrating objects from Yahoo into Google) and Y←G (integrating objects from Google into Yahoo).

As described in §2, we formulate each task as a classification problem. The objects in $G \cap Y$ can be used as test examples, because their categories in both taxonomies are known to us [2]. We hide the test examples' master categories but expose their source categories to the learning algorithm in training phase, and then compare their hidden master categories with the predictions of the learning algorithm in test phase. Suppose the number of the test examples is n. For G←Y tasks, we randomly sample n objects from the set G-Y as training examples. For Y←G tasks, we randomly sample n objects from the set Y-G as training examples. This is to simulate the common situation that the sizes of $\mathcal{M}$ and $\mathcal{N}$ are roughly in same magnitude. For each task, we do such random sampling 5 times, and report the classification performance averaged over these 5 random samplings.

5.3 Measures

As stated in §2, it is natural to accomplish taxonomy integration tasks via building multi-class multi-label classifiers. To measure classification performance for each class (category in $\mathcal{M}$), we use the standard *F*-score (F_1 measure) [3]. The *F*-score is defined as the harmonic average of precision (*p*) and recall (*r*), $F = 2pr/(p+r)$, where precision is the proportion of correctly predicted positive examples among all predicted positive examples, and recall is the proportion of correctly predicted positive examples among all true positive examples. The *F*-scores can be computed for the binary decisions on each individual category first and then be averaged over categories. Or they can be computed globally over all the $M \times n$ binary decisions where M is the number of categories in consideration (the number of categories in $\mathcal{M}$) and n is the number of total test examples (the number of objects in $\mathcal{N}$). The former way is called *macro-averaging* and the latter way is called *micro-averaging* [27]. It is understood that the micro-averaged *F*-score (*miF*) tends to be dominated the classification performance on common categories, and that the macro-averaged *F*-score (*maF*) is more influenced by the classification performance on rare categories [27]. Providing both kinds of scores is more informative than providing either alone.

5.4 Settings

We use our own implementation of NB and ENB. The Lidstone's smoothing parameter η is set to an appropriate value 0.1 [1]. The performance of ENB would be greatly affected by its parameter ω. We run ENB with a series of exponentially increasing values of ω: (0, 1, 3, 10, 30, 100, 300, 1000) [2] for each taxonomy integration task, and report the best experimental results. We use BoosTexter [24] for the implementation of AdaBoost, taking single words as terms. We set the boosting rounds $T = 1000$ and the co-bootstrapping iteration number $R = 8$ (see Figure 1 & 2). In the following sections, we denote the normal AdaBoost approach by AB, and denote the co-bootstrapping approach based on AdaBoost algorithm by CB-AB.

5.5 Results

Table 4: Experimental Results of NB and ENB.

		NB		ENB	
		maF	*miF*	*maF*	*miF*
G←Y	Book	0.1286	0.2384	0.1896	0.5856
	Disease	0.4386	0.5602	0.5230	0.6895
	Movie	0.1709	0.3003	0.2094	0.5331
	Music	0.2386	0.3881	0.2766	0.5408
	News	0.2233	0.4450	0.2578	0.5987
Y←G	Book	0.1508	0.2107	0.2227	0.5471
	Disease	0.2746	0.4812	0.3415	0.6370
	Movie	0.2319	0.4046	0.2884	0.5534
	Music	0.3124	0.5359	0.3572	0.6824
	News	0.2966	0.4219	0.3639	0.6007

Table 5: Experimental Results of AB and CB-AB.

		AB		CB-AB	
		maF	*miF*	*maF*	*miF*
G←Y	Book	0.1740	0.4499	0.2540	0.6030
	Disease	0.5375	0.6674	0.6533	0.7703
	Movie	0.1930	0.4892	0.3172	0.6716
	Music	0.3316	0.5025	0.4851	0.6826
	News	0.2150	0.4625	0.3083	0.6218
Y←G	Book	0.2436	0.3853	0.3516	0.6341
	Disease	0.3719	0.6350	0.4371	0.7287
	Movie	0.2559	0.5214	0.3922	0.7154
	Music	0.4369	0.6397	0.5799	0.7994
	News	0.3774	0.4942	0.4340	0.6421

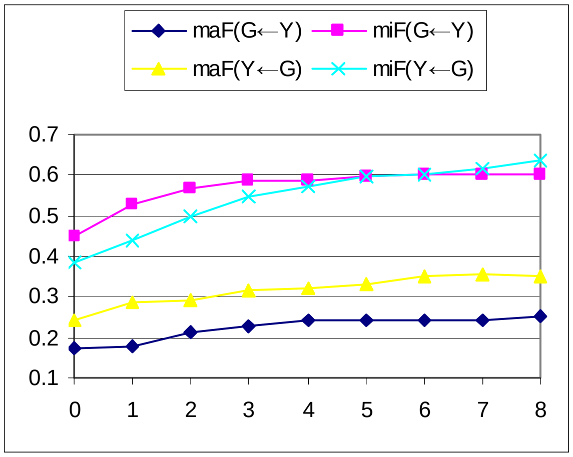

Figure 4: The taxonomy integration performance increases along with the number of co-bootstrapping iterations, on the Book dataset.

The experimental results of NB and ENB are shown in Table 4. We see that ENB really can achieve much better performance than NB for taxonomy integration.

The experimental results of AB and CB-AB are shown in Table 5. Obviously AB beats NB, which is consistent with the conclusion of [24]. Also we find that CB-AB works better than AB for taxonomy integration, which suggests that co-bootstrapping makes effective use of the categorization of $\mathcal{N}$ to enhance classification for $\mathcal{M}$.

Figure 4 shows that the taxonomy integration performance increases along with the number of co-bootstrapping iterations, on the Book dataset. This implies that the two boosting-classifiers learned from two taxonomies do mutually boost each other until they become stable.

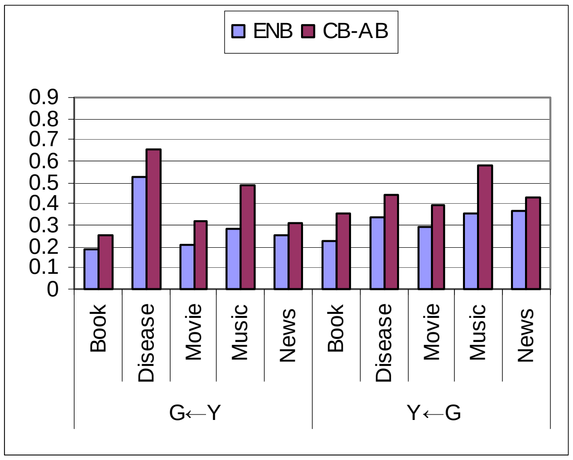

Figure 5: Comparing the macro-averaged F-scores of ENB and CB-AB.

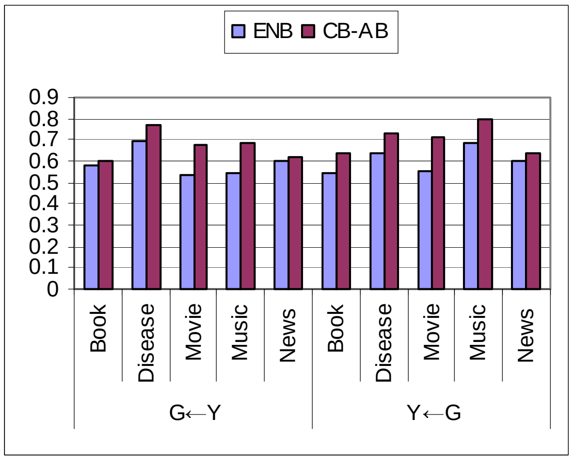

Figure 6: Comparing the micro-averaged F-scores of ENB and CB-AB.

The experimental results of ENB and CB-AB are compared in Figure 5 and 6. It is clear that CB-AB outperforms ENB consistently and significantly.

6. RELATED WORK

Most of the recent research efforts related to taxonomy integration are in the context of ontology mapping on semantic web. An ontology specifies a conceptualization of a domain in terms of concepts, attributes, and relations [11]. The concepts in an ontology are usually organized into a taxonomy: each concept is represented by a category and associated with a set of objects (called the extension of that concept). The basic goal of ontology mapping is to identify (typically one-to-one) semantic correspondences between the taxonomies of two given ontologies: for each concept (category) in one taxonomy, find the most similar concept (category) in the other taxonomy. Many works in this field use a variety of heuristics to find mappings [7, 16, 19, 21]. Recently machine learning techniques have been introduced to further automate the ontology mapping process [10, 13, 14, 20, 26]. Some of them derive similarities between concepts (categories) based on their extensions (objects) [10, 13, 14], therefore they need to first integrate objects from one taxonomy into the other and vice versa (i.e., taxonomy integration). So our work can be utilized as a basic component of an ontology mapping system.

As stated in §2, taxonomy integration can be formulated as a classification problem. The Rocchio algorithm [3, 22] has been applied to this problem in [14]; and the Naïve Bayes (NB) algorithm [18] has been applied to this problem in [10], without exploiting information in the source taxonomy. To our knowledge, the most advanced approach to taxonomy integration is the enhanced Naïve Bayes (ENB) algorithm proposed by Agrawal and Srikant [2], which we have reviewed and compared with our approach.

In [6], AdaBoost is selected as the framework to combine term-features and automatically extracted semantic-features in the context of text categorization. We also choose AdaBoost to combine heterogeneous features (term-features and category-features), but it is for a different problem (taxonomy integration) and it works in a more complex way (through co-bootstrapping).

In [8], an approach called co-boosting is proposed for named entity classification. Essentially co-boosting is a co-training [5] method that attempts to utilize unlabeled data to help classification through exploiting a particular form of redundancy in data: each instance is described by multiple views (disjoint feature sets) which are both compatible and uncorrelated (conditionally independent). However, the multi-view assumption does not hold in the context of taxonomy integration: the set of category features should not be considered as a view because category features alone are not sufficient for classification and they are strongly correlated with term features. In contrast to co-boosting (co-training), co-bootstrapping works with two taxonomies but not two views.

7. CONCLUSION

Our main contribution is to propose a new approach, co-bootstrapping, that can effectively exploit the implicit knowledge in the source taxonomy to improve taxonomy integration.

The future work may include: theoretical analysis of the co-bootstrapping approach, incorporating commonsense knowledge and domain constraints into the taxonomy integration process, and so forth.

8. ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their helpful comments and suggestions.

9. REFERENCES

[1] Agrawal, R., Bayardo, R. and Srikant, R. Athena: Mining-based Interactive Management of Text Databases. in *Proceedings of the 7th International Conference on Extending Database Technology (EDBT)*, Konstanz, Germany, 2000, 365-379.

[2] Agrawal, R. and Srikant, R. On Integrating Catalogs. in *Proceedings of the 10th International World Wide Web Conference (WWW),* Hong Kong, 2001, 603-612.

[3] Baeza-Yates, R. and Ribeiro-Neto, B. *Modern Information Retrieval*. Addison-Wesley, New York, NY, 1999.

[4] Berners-Lee, T., Hendler, J. and Lassila, O. The Semantic Web *Scientific American*, 2001.

[5] Blum, A. and Mitchell, T. Combining Labeled and Unlabeled Data with Co-Training. in *Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT),* Madison, WI, 1998, 92-100.

[6] Cai, L. and Hofmann, T. Text Categorization by Boosting Automatically Extracted Concepts. in *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR),* Toronto, Canada, 2003, 182-189.

[7] Chalupsky, H. OntoMorph: A Translation System for Symbolic Knowledge. in *Proceedings of the 7th International Conference on Principles of Knowledge Representation and Reasoning (KR),* Breckenridge, CO, 2000, 471-482.

[8] Collins, M. and Singer, Y. Unsupervised Models for Named Entity Classification. in *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP)*, College Park, MD, 1999, 189-196.

[9] Cristianini, N. and Shawe-Taylor, J. *An Introduction to Support Vector Machines*. Cambridge University Press, Cambridge, UK, 2000.

[10] Doan, A., Madhavan, J., Domingos, P. and Halevy, A. Learning to Map between Ontologies on the Semantic Web. in *Proceedings of the 11th International World Wide Web Conference (WWW),* Hawaii, USA, 2002.

[11] Fensel, D. *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*. Springer-Verlag, 2001.

[12] Freund, Y. and Schapire, R.E. A Decision-theoretic Generalization of On-line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, *55* (1). 119-139.

[13] Ichise, R., Takeda, H. and Honiden, S. Rule Induction for Concept Hierarchy Alignment. in *Proceedings of the Workshop on Ontologies and Information Sharing at the 17th International Joint Conference on Artificial Intelligence (IJCAI),* Seattle, WA, 2001, 26-29.

[14] Lacher, M.S. and Groh, G. Facilitating the Exchange of Explicit Knowledge through Ontology Mappings. in *Proceedings of the Fourteenth International Florida Artificial Intelligence Research Society Conference (FLAIRS),* Key West, FL, 2001, 305-309.

[15] McCallum, A. and Nigam, K. A Comparison of Event Models for Naive Bayes Text Classification. in *AAAI-98 Workshop on Learning for Text Categorization*, Madison, WI, 1998, 41-48.

[16] McGuinness, D.L., Fikes, R., Rice, J. and Wilder, S. The Chimaera Ontology Environment. in *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI),* Austin, TX, 2000, 1123--1124.

[17] Meir, R. and Ratsch, G. An Introduction to Boosting and Leveraging. in Mendelson, S. and Smola, A.J. eds. *Advanced Lectures on Machine Learning, LNCS,* Springer-Verlag, 2003, 119-184.

[18] Mitchell, T. *Machine Learning*. McGraw Hill, 1997.

[19] Mitra, P., Wiederhold, G. and Jannink, J. Semi-automatic Integration of Knowledge Sources. in *Proceedings of The 2nd International Conference on Information Fusion*, Sunnyvale, CA, 1999.

[20] Noy, N.F. and Musen, M.A. Anchor-PROMPT: Using Non-Local Context for Semantic Matching. in *Proceedings of the Workshop on Ontologies and Information Sharing at the 17th International Joint Conference on Artificial Intelligence (IJCAI),* Seattle, WA, 2001, 63-70.

[21] Noy, N.F. and Musen, M.A. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. in *Proceedings of the National Conference on Artificial Intelligence (AAAI),* Austin, TX, 2000, 450-455.

[22] Rocchio, J.J. Relevance Feedback in Information Retrieval. in Salton, G. ed. *The SMART Retrieval System: Experiments in Automatic Document Processing*, Prentice-Hall, 1971, 313-323.

[23] Schapire, R.E. The Boosting Approach to Machine Learning: An Overview. in *MSRI Workshop on Nonlinear Estimation and Classification*, Berkeley, CA, 2002.

[24] Schapire, R.E. and Singer, Y. BoosTexter: A Boosting-based System for Text Categorization. *Machine Learning*, *39* (2/3). 135-168.

[25] Schapire, R.E. and Singer, Y. Improved Boosting Algorithms Using Confidence-rated Predictions. *Machine Learning*, *37* (3). 297-336.

[26] Stumme, G. and Maedche, A. FCA-MERGE: Bottom-Up Merging of Ontologies. in *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI),* Seattle, WA, 2001, 225-230.

[27] Yang, Y. and Liu, X. A Re-examination of Text Categorization Methods. in *Proceedings of the 22nd ACM International Conference on Research and Development in Information Retrieval (SIGIR)*, Berkeley, CA, 1999, 42-49.