Web Taxonomy Integration using Support Vector Machines

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 train a classifier for each category in the master taxonomy, and then classify objects from the source taxonomy into these categories. In this paper we attempt to use a powerful classification method, Support Vector Machine (SVM), to attack this problem. Our key insight is that the availability of the source taxonomy data could be helpful to build better classifiers in this scenario, therefore it would be beneficial to do transductive learning rather than inductive learning, i.e., learning to optimize classification performance on a particular set of test examples. Noticing that the categorizations of the master and source taxonomies often have some semantic overlap, we propose a method, Cluster Shrinkage (CS), to further 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.

Keywords
Semantic Web, Ontology Mapping, Taxonomy Integration, Classification, Support Vector Machines, Transductive Learning.

WWW 2004, May 17–22, 2004, New York, New York, USA.
ACM 1-58113-844-X/04/0005.

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 [8, 15].

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

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

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

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

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 "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]. In this paper, we attempt to use a powerful classification method, Support Vector Machine (SVM) [7], to attack this problem.

Our key insight is that the availability of the source taxonomy data could be helpful to build better classifiers in this scenario, therefore it would be beneficial to do transductive learning rather than inductive learning, i.e., learning to optimize classification performance on a particular set of test examples. Noticing that the categorizations of the master and source taxonomies often have some semantic overlap, we propose a method, Cluster Shrinkage (CS), to further 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

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 category 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 class of each test example.

It is possible that an object in $\mathcal{N}$ belongs to multiple categories in $\mathcal{M}$. Besides, some objects in $\mathcal{N}$ may not fit well in any existing category in $\mathcal{M}$, so users may want to have the option to form a new category for them. It is therefore instructive to create an ensemble of binary (yes/no) classifiers, one for each category C in $\mathcal{M}$. When training the classifier for C, an object in $\mathcal{M}$ is labeled as a positive example if it is contained by C or as a negative example otherwise. All objects in $\mathcal{N}$ are unlabeled and wait to be classified. This is called the "one-vs-rest" ensemble method.

Taxonomies are often organized as hierarchies. In this paper, we focus on flat taxonomies. Generalizing our approach to hierarchical taxonomies is straightforward and will be discussed later.

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 [16]. 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

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

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

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

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 $\mathcal{N}$. 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.

4. OUR APPROACH

Here we present our approach in detail. In §4.1, we review Support Vector Machine (SVM). In §4.2, we review transductive learning and explain why it is more suitable to our task. In §4.3, we propose the Cluster Shrinkage (CS) method and analyze its effect. In §4.4, we compare our approach with ENB.

4.1 Support Vector Machines

Support Vector Machine (SVM) [7, 13] is a powerful classification method which has shown outstanding classification performance in practice. It is based on a solid theoretical foundation — *structural risk minimization* [24].

In its simplest linear form, an SVM is a hyperplane that separates the positive and negative training examples with maximum margin, as shown in Figure 1. Large margin between positive and negative examples has been proven to lead to good generalization [24].

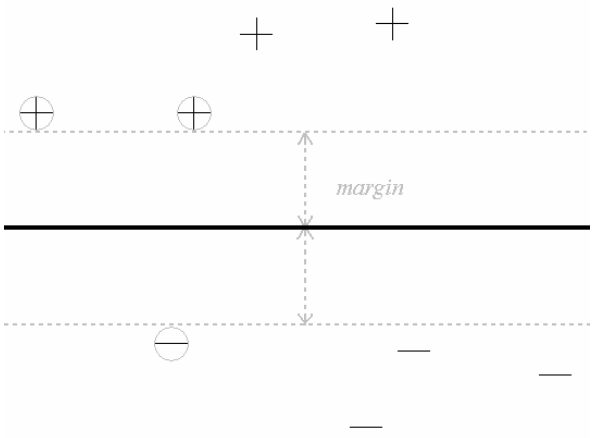

Figure 1: An SVM is a hyperplane that separates the positive and negative training examples with maximum margin. The examples closest to the hyperplane are called support vectors (marked with circles).

The decision function of an SVM is $f(\mathbf{x}) = \langle \mathbf{w} \bullet \mathbf{x} \rangle + b$, where $\langle \mathbf{w} \bullet \mathbf{x} \rangle$ is the dot product between $\mathbf{w}$ (the normal vector to the hyperplane) and $\mathbf{x}$ (the feature vector representing an example). The margin for an input vector $\mathbf{x}_i$ is $y_i f(\mathbf{x}_i)$ where $y_i \in \{-1, 1\}$ is the correct class label for $\mathbf{x}_i$. In the linear case, the margin is geometrically the distance from the hyperplane to the nearest positive and negative examples. Seeking the maximum margin can be expressed as an quadratic optimization problem: minimizing $\langle \mathbf{w} \bullet \mathbf{w} \rangle$ subject to $y_i (\langle \mathbf{w} \bullet \mathbf{x}_i \rangle + b) \geq 1$, $\forall i$. When positive and negative examples are linearly inseparable, soft-margin SVM tries to solve a modified optimization problem that allows but penalizes the examples falling on the wrong side of the hyperplane.

4.2 Transductive Learning

A regular SVM tries to induce a general classifying function which has high accuracy on the whole distribution of examples. However, this so-called inductive learning setting is often unnecessarily complex. For the classification problem in taxonomy integration situations, the set of test examples to be classified are already known to the learning algorithm. In fact, we do not care about the general classifying function, but rather attempt to achieve good classification performance on that particular set of test examples. This is exactly the goal of transductive learning [25].

Transductive SVM (TSVM) introduced by Joachims [14] extends SVM to transductive learning setting. A TSVM is essentially a hyperplane that separates the positive and negative training examples with maximum margin on both training and test examples, as shown in Figure 2.

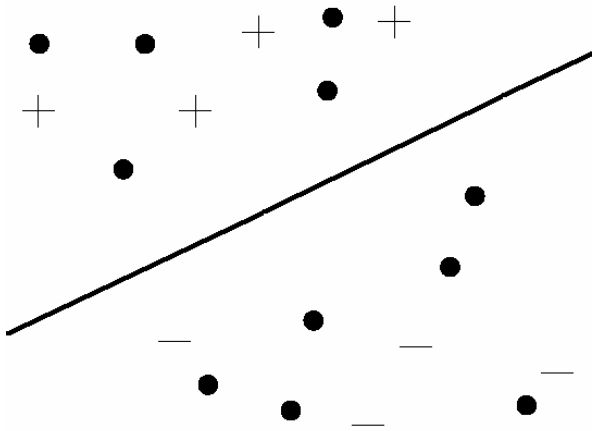

Figure 2: A TSVM is essentially a hyperplane that separates the positive and negative training examples with maximum margin on both training and test examples (cf. Figure 1).

Why can TSVM be better than SVM? There usually exists a clustering structure of training and test examples: the examples in same class tend to be close to each other in feature space. As explained in [14], it is this clustering structure of examples that TSVM exploits as prior knowledge to boost classification performance. This is especially beneficial when the number of training examples is small.

Most machine learning algorithms (including NB, SVM and TSVM) assume that both the training and test examples come from the identical data distribution. This assumption does not necessarily hold in the case of taxonomy integration. Intuitively, TSVM seems to be more robust than SVM to the violation of this assumption, since TSVM takes the test examples into account for learning. This interesting issue needs to be stressed in the future.

4.3 Cluster Shrinkage

Applying TSVM, we can effectively use the objects in $\mathcal{N}$ (test examples) to boost classification performance. However, 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 $\mathcal{M}$. For example, if two objects belong to the same category S in $\mathcal{N}$, they are more likely to belong to the same category C in $\mathcal{M}$ rather than to be assigned into different categories. We hereby propose a method, Cluster Shrinkage (CS), to further enhance the classification by exploiting such implicit knowledge.

4.3.1 Algorithm

for each category S **{**

compute its center: $\mathbf{c} = \frac{1}{|S|}\sum_{\mathbf{x}\in S}\mathbf{x}$;

for each example $\mathbf{x}\in S$ **{**

replace it with $\mathbf{x}' = \lambda\mathbf{c} + (1-\lambda)\mathbf{x}$,

where $0 \le \lambda \le 1$;

}

}

Figure 3: The Cluster Shrinkage algorithm.

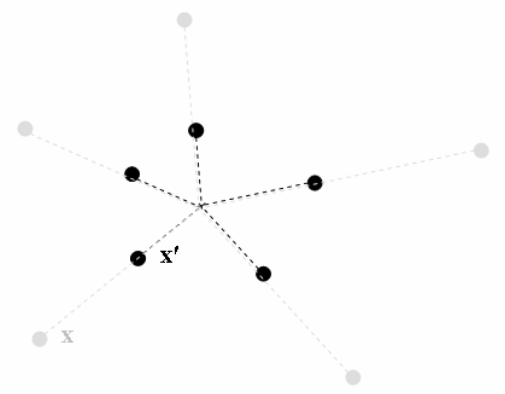

Figure 4: The Cluster Shrinkage process.

Since the success of TSVM relies on the clustering structure of examples, we intend to use the categorization information in the taxonomies to strengthen this clustering structure and thus help TSVM to find better classification. This can be achieved by treating each category S (or C) as a cluster and shrinking it. Figure 3. presents our proposed Cluster Shrinkage (CS) algorithm, and Figure 4. depicts its process.

The formula $\mathbf{x}' = \lambda\mathbf{c} + (1-\lambda)\mathbf{x}$ is actually a linear interpolation of the example $\mathbf{x}$ and its category's center $\mathbf{c}$. When an example $\mathbf{x}$ belongs to multiple categories $S^{(1)}, S^{(2)}, ..., S^{(g)}$ whose centers are $\mathbf{c}^{(1)}, \mathbf{c}^{(2)}, ..., \mathbf{c}^{(g)}$ respectively, the above formula should be amended to $\mathbf{x}' = \lambda\left(\frac{1}{g}\sum_{h=1}^{g}\mathbf{c}^{(h)}\right) + (1-\lambda)\mathbf{x}$.

Our approach to taxonomy integration is in three steps: first apply CS on all objects in $\mathcal{M}$ and $\mathcal{N}$, then train TSVMs on these objects, finally use the learned TSVMs to classify the objects in $\mathcal{N}$ into the categories in $\mathcal{M}$. We name this approach CS-TSVM.

4.3.2 Analysis

We first study the effect of CS in inductive learning setting.

Denoting the Euclidean distance between two examples (vectors) with function $d(\cdot,\cdot)$, we can get the following theorem.

THEOREM 1. For any example $\mathbf{x}\in S$, suppose the center of S is $\mathbf{c}$, CS makes $\mathbf{x}$ become $\mathbf{x}'$, then
$d(\mathbf{x}', \mathbf{c}) = (1-\lambda)d(\mathbf{x}, \mathbf{c}) \le d(\mathbf{x}, \mathbf{c})$.
Proof:
Since $\mathbf{x}' = \lambda\mathbf{c} + (1-\lambda)\mathbf{x}$, we get
$d(\mathbf{x}', \mathbf{c}) = \|\mathbf{x}' - \mathbf{c}\| = \|(\lambda\mathbf{c} + (1-\lambda)\mathbf{x}) - \mathbf{c}\|$
$= \|(1-\lambda)(\mathbf{x} - \mathbf{c})\| = (1-\lambda)\|\mathbf{x} - \mathbf{c}\| = (1-\lambda)d(\mathbf{x}, \mathbf{c})$.
Since $0 \le \lambda \le 1$, we get
$0 \le 1-\lambda \le 1$, $(1-\lambda)d(\mathbf{x}, \mathbf{c}) \le d(\mathbf{x}, \mathbf{c})$.

From the above theorem, we see that CS is actually moving all examples in a category towards their center. Hence, applying CS on the objects in $\mathcal{M}$ (training examples) would make SVM behave alike the Rocchio algorithm [3, 22], which is not going to provide much help because Rocchio is not as powerful as SVM..

Given a linear classifier $f(\mathbf{x}) = \langle \mathbf{w} \bullet \mathbf{x} \rangle + b$, we can get the following theorem.

THEOREM 2. For any example $\mathbf{x}\in S$, suppose the center of S is $\mathbf{c}$, CS makes $\mathbf{x}$ become $\mathbf{x}'$, then
$f(\mathbf{x}') = \lambda f(\mathbf{c}) + (1-\lambda) f(\mathbf{x})$.
Proof:
Since $\mathbf{x}' = \lambda\mathbf{c} + (1-\lambda)\mathbf{x}$, we get
$f(\mathbf{x}') = \langle \mathbf{w} \bullet \mathbf{x}' \rangle + b = \langle \mathbf{w} \bullet (\lambda\mathbf{c} + (1-\lambda)\mathbf{x}) \rangle + b$
$= \lambda\langle \mathbf{w} \bullet \mathbf{c} \rangle + (1-\lambda)\langle \mathbf{w} \bullet \mathbf{x} \rangle + (\lambda + 1 - \lambda)b$
$= \lambda\left(\langle \mathbf{w} \bullet \mathbf{c} \rangle + b\right) + (1-\lambda)\left(\langle \mathbf{w} \bullet \mathbf{x} \rangle + b\right)$
$= \lambda f(\mathbf{c}) + (1-\lambda) f(\mathbf{x})$.

From the above theorem, we see that applying CS on the objects in $\mathcal{N}$ (test examples) can push these objects to get classifying function outputs more similar to those of their category centers. However, in inductive learning setting, the objects in $\mathcal{N}$ (test examples) are not involved in construction of the classifiers, i.e.,

applying CS on the objects in $\mathcal{N}$ would have no opportunity to change the classifiers. Therefore the benefits of CS to inductive learning algorithms for taxonomy integration would be limited. This thought has been confirmed by our experiments of CS-SVM (the combination of CS and SVM).

We then study the effect of CS in transductive learning setting.

THEOREM 3. For any pair of examples $\mathbf{x}_1 \in S$ and $\mathbf{x}_2 \in S$, suppose the center of S is $\mathbf{c}$, CS makes $\mathbf{x}_1$ and $\mathbf{x}_2$ become $\mathbf{x}_1'$ and $\mathbf{x}_2'$ respectively, then

$d(\mathbf{x}_1', \mathbf{x}_2') = (1-\lambda) d(\mathbf{x}_1, \mathbf{x}_2) \le d(\mathbf{x}_1, \mathbf{x}_2)$.

Proof:

Since $\mathbf{x}_1' = \lambda \mathbf{c} + (1-\lambda)\mathbf{x}_1$ and $\mathbf{x}_2' = \lambda \mathbf{c} + (1-\lambda)\mathbf{x}_2$, we get

$$d(\mathbf{x}_1', \mathbf{x}_2') = \left\| \mathbf{x}_1' - \mathbf{x}_2' \right\| = \left\| \left(\lambda \mathbf{c} + (1-\lambda)\mathbf{x}_1 \right) - \left(\lambda \mathbf{c} + (1-\lambda)\mathbf{x}_2 \right) \right\|$$

$$= \left\| (1-\lambda)(\mathbf{x}_1 - \mathbf{x}_2) \right\| = (1-\lambda) \left\| \mathbf{x}_1 - \mathbf{x}_2 \right\| = (1-\lambda) d(\mathbf{x}_1, \mathbf{x}_2)$$

Since $0 \le \lambda \le 1$, we get

$0 \le 1-\lambda \le 1$, $(1-\lambda) d(\mathbf{x}_1, \mathbf{x}_2) \le d(\mathbf{x}_1, \mathbf{x}_2)$.

From the above theorem, we see that CS lets all examples in a category become closer to each other. Because TSVM seeks the maximum margin hyperplane (the thickest slab) in both training and test examples, making the examples in category S closer to each other directs TSVM to avoid splitting S. Consequently applying CS on the objects in $\mathcal{N}$ (test examples) guides TSVM to reserve the original categorization of $\mathcal{N}$ to some degree while doing classification, as shown in Figure 5. On the other hand, applying CS on the objects in $\mathcal{M}$ (training examples) meanwhile can reduce TSVM's dependence on training examples and put more emphasis on taking advantage of the information in $\mathcal{N}$.

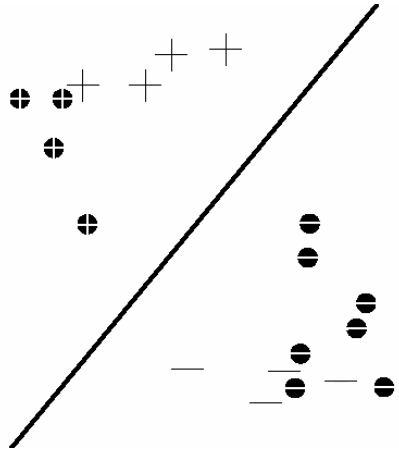

Figure 5: A CS-TSVM attempts to reserve the original categorization of the source taxonomy to some degree while doing classification (cf. Figure 2).

To sum up, the CS-TSVM approach can not only make effective use of the objects in $\mathcal{N}$ like TSVM, but also make effective use of the categorization of $\mathcal{N}$.

The CS parameter $0 \le \lambda \le 1$ controls the strength of the clustering structure of examples. Increasing λ results in more influence of the categorization information on classification. When $\lambda = 1$, CS-TSVM classifies all objects belonging to one category in $\mathcal{N}$ as a whole into a specific category in $\mathcal{M}$. When $\lambda = 0$, CS-TSVM is just the same as TSVM. As long as the value of λ is set appropriately, CS-TSVM should never be worse than TSVM because it includes TSVM as a special case. The optimal value of λ 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, as described in [2].

Another way to incorporate the categorization of $\mathcal{N}$ into TSVM is to treat the source category labels $S_1, S_2, ..., S_N$ as binary features, and expand each feature vector $\mathbf{x}$ to $\mathbf{x}''$ by appending extra columns for these label features. Similarly a parameter $0 \le \lambda \le 1$ can be used to decide the relative importance of category and ordinary features: category features are scaled by factor λ and ordinary features are scaled by $1-\lambda$. This method looks simpler, but it does not leverage as much categorization information as CS. For illustration, consider two different categories S_1 and S_2 whose centers are $\mathbf{c}_1$ and $\mathbf{c}_2$ respectively, given two examples $\mathbf{x}_1 \in S_1$ and $\mathbf{x}_2 \in S_2$, let parameter $\lambda = 1$, the above simpler method would get $\langle \mathbf{x}_1'' \bullet \mathbf{x}_2'' \rangle = 0$, while CS would provide a more reasonable dot product function $\langle \mathbf{x}_1' \bullet \mathbf{x}_2' \rangle = \langle \mathbf{c}_1' \bullet \mathbf{c}_2' \rangle$.

4.3.3 Extensions

4.3.3.1 Nonlinear Classification

One salient property of SVM / TSVM is that the only operation it requires is the computation of dot products between pairs of examples. One may therefore replace the dot product with a Mercer kernel [7], implicitly mapping feature vectors in Ω into a higher dimensional space $\tilde{\Omega}$, and applying the original algorithm in this new space. Using a non-linear kernel (e.g., polynomial, rbf or sigmoid) enables SVM / TSVM to get non-linear classification in Ω, thus greatly promoting the power of SVM / TSVM.

Suppose for SVM / TSVM we use a non-linear kernel $k = \langle \phi(\mathbf{x}_1) \bullet \phi(\mathbf{x}_2) \rangle$, where ϕ is a non-linear map from Ω to $\tilde{\Omega}$. The idea of Cluster Shrinkage in $\tilde{\Omega}$ is to replace each feature vector $\phi(\mathbf{x})$ in category S with $\phi'(\mathbf{x}) = \lambda \tilde{\mathbf{c}} + (1-\lambda)\phi(\mathbf{x})$, where $\tilde{\mathbf{c}} = \frac{1}{|S|} \sum_{\mathbf{x} \in S} \phi(\mathbf{x})$ is the center of S in $\tilde{\Omega}$. We are usually unable to explicitly express a feature vector $\phi(\mathbf{x})$ in $\tilde{\Omega}$, because the dimension of $\tilde{\Omega}$ is extremely large or even infinite. However, CS in $\tilde{\Omega}$ can still be achieved implicitly by replacing the kernel k with the following one.

$$k'(\mathbf{x}_1, \mathbf{x}_2) = \langle \phi'(\mathbf{x}_1) \bullet \phi'(\mathbf{x}_2) \rangle$$

$$= \left\langle \left(\lambda \tilde{\mathbf{c}}_1 + (1-\lambda)\phi(\mathbf{x}_1) \right) \bullet \left(\lambda \tilde{\mathbf{c}}_2 + (1-\lambda)\phi(\mathbf{x}_2) \right) \right\rangle$$

$$\approx \left\langle \left(\lambda \phi(\mathbf{c}_1) + (1-\lambda)\phi(\mathbf{x}_1) \right) \bullet \left(\lambda \phi(\mathbf{c}_2) + (1-\lambda)\phi(\mathbf{x}_2) \right) \right\rangle$$

$$= \lambda^2 \langle \phi(\mathbf{c}_1) \bullet \phi(\mathbf{c}_2) \rangle + (1-\lambda)^2 \langle \phi(\mathbf{x}_1) \bullet \phi(\mathbf{x}_2) \rangle$$

$$+\lambda(1-\lambda)\left(\langle\phi(\mathbf{c}_1)\bullet\phi(\mathbf{x}_2)\rangle+\langle\phi(\mathbf{c}_2)\bullet\phi(\mathbf{x}_1)\rangle\right)$$
$$=\lambda^2 k(\mathbf{c}_1,\mathbf{c}_2)+(1-\lambda)^2 k(\mathbf{x}_1,\mathbf{x}_2)$$
$$+\lambda(1-\lambda)\left(k(\mathbf{c}_1,\mathbf{x}_2)+k(\mathbf{c}_2,\mathbf{x}_1)\right).$$

Note that in the above formula, we have approximated the center of category S in $\tilde{\Omega}$, $\tilde{\mathbf{c}}=\frac{1}{|S|}\sum_{\mathbf{x}\in S}\phi(\mathbf{x})$, with $\phi(\mathbf{c})=\phi\left(\frac{1}{|S|}\sum_{\mathbf{x}\in S}\mathbf{x}\right)$. Although it is possible to derive a strict formula of $k'(\mathbf{x}_1,\mathbf{x}_2)$ without this approximation, it would be computationally more expensive. In this way, we are able to implement CS for non-linear SVM / TSVM efficiently.

4.3.3.2 Hierarchical Classification

As mentioned before, taxonomies are often organized as hierarchies. Although it is possible to flatten the hierarchy to a single level [2], past studies have shown that exploiting the hierarchical structure can lead to better classification results [5, 9]. We think the technique of hierarchical SVM proposed in [9] can be easily extended to hierarchical TSVM and then incorporate the hierarchical version of CS. For instance, consider a two-level taxonomy $\mathcal{H}$ where S_{jk} is a sub-category of S_j, suppose the center of S_{jk} is $\mathbf{c}_{jk}$ and the center of S_j is $\mathbf{c}_j$, for each $\mathbf{x}\in S_{ik}\subset S_i$, one reasonable way to achieve hierarchical CS is as follows: first compute $\mathbf{c}'_{ik}=\mu\mathbf{c}_i+(1-\mu)\mathbf{c}_{ik}$ using a parameter $0\le\mu\le 1$, and then replace $\mathbf{x}$ with $\mathbf{x}'=\lambda\mathbf{c}'_{ik}+(1-\lambda)\mathbf{x}$ using a parameter $0\le\lambda\le 1$.

4.4 Comparison with ENB

Although ENB [2] has been shown to work well for taxonomy integration, we think an approach based on SVM but not NB is still attractive.

In contrast to NB, SVM is a discriminative classification method, i.e., SVM does not posit a generative model but attempt to find the best classifying function directly. It is generally believed that SVM is more promising than NB for text classification [10, 26], and SVM has been successfully applied to many other kinds of data such as images [7].

Both ENB and CS-TSVM 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), CS-TSVM has the ability to adjust the direction of the classification hyperplane of its base TSVM classifier. Moreover, CS-TSVM has the potential to be extended to achieve non-linear and hierarchical classifications.

Although CS-TSVM looks more effective, ENB still has the advantage in efficiency.

5. EXPERIMENTS

We conduct experiments with real-world web data, to demonstrate the advantage of our proposed CS-TSVM approach to taxonomy integration.

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.

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

Table 2: The number of categories.

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

In each category, we take all items listed on the corresponding directory page and its sub-directory pages as its objects. An object (listed 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, as illustrated in Figure 6.

Figure 6: An object (listed item) corresponds a website on the world wide web, which is usually described by its URL, its title, and optionally a short annotation about its content.

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 confirms our previous statement in §2 that an object may belong to multiple categories, and justifies our

strategy to build a binary classifier for each category in the master taxonomy.

Table 3: The number of objects.

	Google	Yahoo	G ∪ Y	G ∩ 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

The category distributions in all theses datasets are highly skewed. For example, in Google's Book taxonomy, the most common category contains 21% objects, but 88% categories contain less than 3% objects and 49% categories contain less than 1% objects, as shown in Figure 7. In fact, skewed category distributions have been commonly observed in real-world applications [26].

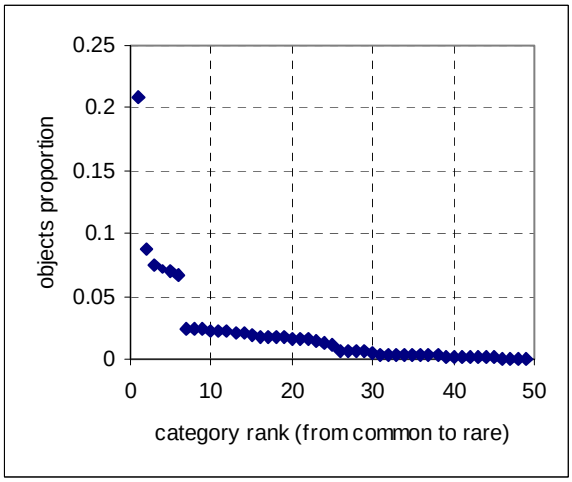

Figure 7: The category distribution of Google's Book taxonomy.

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 ∩ 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 Features

For each object, we assume that the title and annotation of its corresponding website summarizes its content. So each object can be considered as a text document composed of its title and annotation.

The most commonly used feature extraction technique for text data is to treat a document as a bag-of-words [13, 14]. For each document d in a collection of documents D, its bag-of-words is first pre-processed by removal of stop-words and stemming. Then it is represented as a feature vector $\mathbf{x} = (x_1, x_2, ..., x_m)$, where x_i indicates the importance weight of term w_i (the *i*-th distinct word occurred in D). Following the TF×IDF weighting scheme, we set the value of x_i to the product of the term frequency $TF(w_i, d)$ and the inverse document frequency $IDF(w_i)$, i.e., $TF(w_i, d) \times IDF(w_i)$. The term frequency $TF(w_i, d)$ means the number of occurrences of w_i in d. The inverse document frequency is defined as $IDF(w_i) = \log\left(\frac{|D|}{DF(w_i)}\right)$, where $|D|$ is the total number of documents in D, and $DF(w_i)$ is the number of documents in which w_i occur. Finally all feature vectors are normalized to have unit length.

5.4 Measures

As stated in §2, it is natural to accomplish a taxonomy integration task via an ensemble of binary classifiers, each for one category in $\mathcal{M}$. To measure classification performance, 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* [26]. 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 [26]. Since the category distributions are highly skewed (see §5.1), providing both kinds of scores is more informative than providing either alone.

5.5 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 SVMlight[6] for the implementation of SVM / TSVM [13, 14]. We take linear kernel, and accept all default values of parameters except "j" and "p". The parameter "j" is set to the ratio of negative training examples over positive training examples, thus balance the cost of training errors on positive and negative examples. The parameter "p" used in TSVM means the fraction of test examples to be classified into the positive class. To estimate the value of "p", we first run SVM and get $\hat{p}$ (the fraction of test examples predicted to be positive by SVM), then we set the value of "p" to a smoothed version of $\hat{p}$: $\sigma \times \hat{p} + (1-\sigma) \times 0.5$, where σ is set to 99% in our experiments.

The CS algorithm is simple to implement and executes quickly. It only requires one sequential scan to compute the cluster centers and another sequential scan to reposition the examples. In all our CS-SVM and CS-TSVM experiments, the CS parameter λ is set to 0.5. Fine-tuning λ using tune sets would decisively generate better results than sticking with a pre-fixed value. In other words, the performance superiority of applying CS technique is under-estimated in our experiments.

5.6 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 SVM and TSVM.

		SVM		TSVM	
		maF	*miF*	*maF*	*miF*
G←Y	Book	0.2032	0.4916	0.2945	0.5089
	Disease	0.6546	0.7466	0.6844	0.7686
	Movie	0.2563	0.5104	0.3350	0.5290
	Music	0.4774	0.5856	0.4604	0.5921
	News	0.3413	0.5349	0.4408	0.5778
Y←G	Book	0.3284	0.4267	0.4284	0.4666
	Disease	0.5842	0.7470	0.6362	0.7701
	Movie	0.3731	0.5503	0.3995	0.5648
	Music	0.5175	0.6649	0.5236	0.6670
	News	0.4948	0.5848	0.5110	0.6035

[6] http://svmlight.joachims.org/

Table 6: Experimental Results of CS-SVM & CS-TSVM

		CS-SVM		CS-TSVM	
		maF	*miF*	*maF*	*miF*
G←Y	Book	0.1450	0.5084	0.3564	0.6160
	Disease	0.6494	0.7807	0.7288	0.8038
	Movie	0.2070	0.5361	0.3686	0.6182
	Music	0.3917	0.6268	0.5108	0.6768
	News	0.1837	0.5072	0.5504	0.6459
Y←G	Book	0.2421	0.4219	0.4936	0.5706
	Disease	0.3933	0.6846	0.6704	0.7919
	Movie	0.2522	0.5425	0.4434	0.6481
	Music	0.5205	0.7793	0.6181	0.8053
	News	0.4090	0.5391	0.5572	0.6456

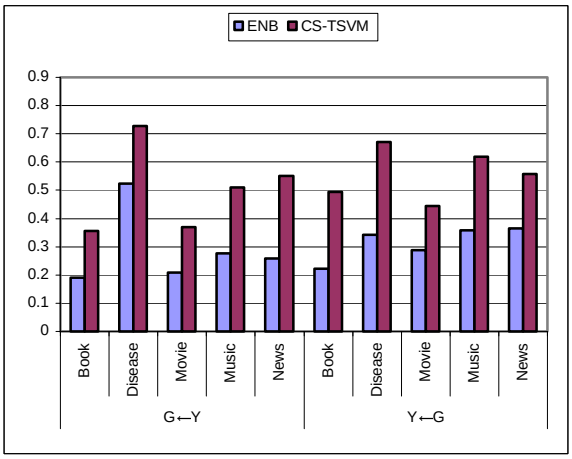

Figure 8: Comparing the macro-averaged F-scores of ENB and CS-TSVM.

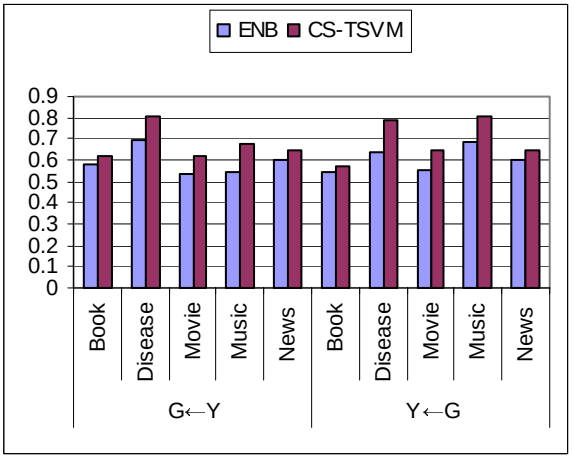

Figure 9: Comparing the micro-averaged F-scores of ENB and CS-TSVM.

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 SVM and TSVM are shown in Table 5. We see that TSVM works better than SVM for taxonomy

integration tasks. We think this is because TSVM makes effective use of the objects in $\mathcal{N}$ to enhance classification.

The experimental results of CS-SVM and CS-TSVM are shown in Table 6. Comparing the experimental results of CS-SVM and SVM, it turns out that in inductive learning setting the CS technique can not provide much help to taxonomy integration. In contrast, CS-TSVM greatly improves TSVM in the performance of taxonomy integration. This implies that the real power of CS-TSVM comes from the marriage of CS and TSVM but not either alone.

The experimental results of ENB and CS-TSVM are compared in Figure 8 and 9. It is clear that CS-TSVM 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 [6, 17, 19, 21]. Recently machine learning techniques have been introduced to further automate the ontology mapping process [8, 12, 15, 20, 23]. Some of them derive similarities between concepts (categories) based on their extensions (objects) [8, 12, 15], 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 [15]; and the Naïve Bayes (NB) algorithm [18] has been applied to this problem in [8], 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.

7. CONCLUSION

Our main contribution is to show that the implicit knowledge in the source taxonomy can be effectively exploited to boost taxonomy integration by marrying Cluster Shrinkage (CS) and Transductive Support Vector Machines (TSVM).

The future work may include: looking for methods to accelerate the proposed CS-TSVM approach, incorporating commonsense knowledge and domain constraints into the taxonomy integration process, extending to full-functional ontology mapping systems, 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] Chakrabarti, S., Dom, B., Agrawal, R. and Raghavan, P. Using Taxonomy, Discriminants, and Signatures for Navigating in Text Databases. in *Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB)*, Athens, Greece, 1997, 446-455.

[6] 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.

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

[8] 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.

[9] Dumais, S. and Chen, H. Hierarchical Classification of Web Content. in *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, Athens, Greece, 2000, 256-263.

[10] Dumais, S., Platt, J., Heckerman, D. and Sahami, M. Inductive Learning Algorithms and Representations for Text Categorization. in *Proceedings of the 7th ACM International Conference on Information and Knowledge Management (CIKM)*, Bethesda, MD, 1998, 148-155.

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

[12] 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.

[13] Joachims, T. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. in

Proceedings of the 10th European Conference on Machine Learning (ECML), Chemnitz, Germany, 1998, 137-142.

[14] Joachims, T. Transductive Inference for Text Classification using Support Vector Machines. in *Proceedings of the 16th International Conference on Machine Learning (ICML)*, Bled, Slovenia, 1999, 200-209.

[15] 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.

[16] 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.

[17] 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.

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

[24] Vapnik, V.N. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, NY, 2000.

[25] Vapnik, V.N. *Statistical Learning Theory*. Wiley, New York, NY, 1998.

[26] 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.