On Web Taxonomy Integration

Dell Zhang and Wee Sun Lee
Singapore-MIT Alliance, 4 Engineering Drive 3, National University of Singapore, Singapore-117576

***Abstract*— We address the problem of integrating objects from a source taxonomy into a master taxonomy. This problem is not only pervasive on the nowadays 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 categorization of the master and source taxonomies often have some semantic overlap, we propose a new 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.**

***Index Terms*—Web Taxonomy Integration, Classification, Support Vector Machines, Transductive Learning**

I. 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 pervasive on the nowadays web, given that many websites are aggregators of information from various other websites [1]. A few examples will illustrate the scenario. A web marketplace like Amazon (http:// *www. amazon. com/*) may want to combine goods from multiple vendors' catalogs into its own. A web portal like DBLP (*http:// dblp. uni-trier. de/*) 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 (*http:// web. mit. edu/ sma/*), 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 [2], where data has structure 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 [3].

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 Google (*http:// www. google. com/*) and Yahoo (*http:// www. yahoo. com/*): what is "Arts/ Music/ Styles/" in one may be "Entertainment/ Music/ Genres/" in the other, "Computers_and_Internet/ Software/ Freeware" and "Computers/ Open_Source/ Software" have similar contents but meanwhile 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. These standards, moreover, are far from static.

Manually 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 [4]. In this paper, we attempt to use a powerful classification method, Support Vector Machine (SVM) [5], 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 categorization of the master and source taxonomies often have some semantic overlap, we propose a new method, Cluster Shrinkage (CS), to further

Dell Zhang is with the Computer Science Program in Singapore-MIT Alliance, National University of Singapore, Singapore 117543 (phone: 65-6874-4251; fax: 65-6779-4580; e-mail: dell.z@ieee.org).

Wee Sun Lee is with the Computer Science Program in Singapore-MIT Alliance, and Department of Computer Science in National University of Singapore, Singapore 117543 (e-mail: leews@comp.nus.edu.sg).

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 detailed problem statement. In §3, we review Support Vector Machines. In §4, we describe transductive learning and explain why it is more suitable to our task. In §5, we present our proposed Cluster Shrinkage method and analyze its effect. In §6, we conduct empirical evaluation that show the promise of our approach. In §7, we discuss the related work. In §8, we make concluding remarks.

II. Problem Statement

Now we formally define the taxonomy integration problem 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 approach.

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

III. Support Vector Machines

Support Vector Machine (SVM) [5] is a powerful classification method which has shown outstanding classification performance in practice. It has a solid theoretical foundation called *structural risk minimization* [6].

In its simplest linear form, an SVM is a hyperplane that separates the positive and negative training examples with maximum margin, as shown in Fig. 1.

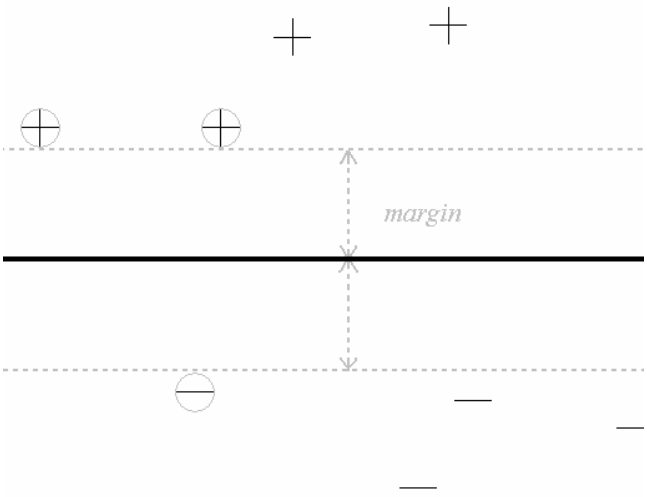

Fig. 1. An SVM is a hyperplane that separates the positive and negative training examples with maximum margin. The examples closest to the hyperplane are called support vectors (marked with circles).

The formula for the output of a linear SVM is $f(x) = \langle \mathbf{w} \bullet \mathbf{x} \rangle + b$, 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 maximal 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. Large margin between positive and negative examples has been proven to lead to good generalization capacity [6].

IV. 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 [7].

Transductive SVM (TSVM) introduced by Joachims [8] 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 Fig. 2.

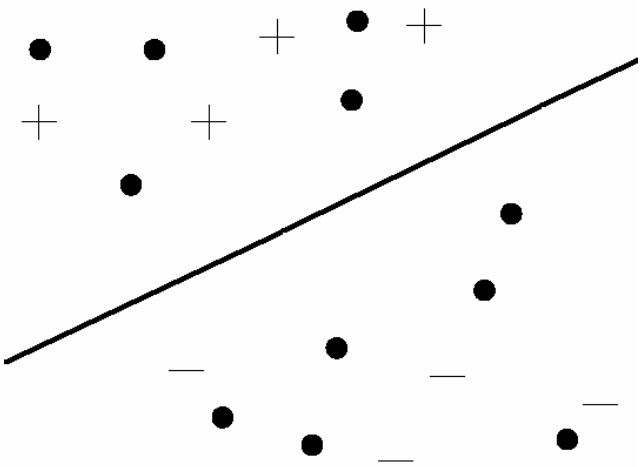

Fig. 2. A TSVM is essentially a hyperplane that separates the positive and negative training examples with maximum margin on both training and test examples (cf. Fig. 1).

Why TSVM can 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 [8], 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.

V. Cluster Shrinkage

Applying TSVM, we can effectively use the objects in the source taxonomy (test examples) to boost classification performance. However, thus far we have completely ignored the categorization of the source taxonomy.

Although the master and source taxonomies are usually not identical, their categorizations often have some semantic overlap. Therefore the categorization of the source taxonomy contains valuable implicit knowledge about the categorization of the master taxonomy. 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 here propose a new method, Cluster Shrinkage (CS), to further enhance the classification by exploiting such implicit knowledge.

A. Algorithm

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 as a cluster and shrinking it.

Fig. 3. presents our proposed Cluster Shrinkage algorithm, and Fig. 4. depicts its process.

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 $0 \le \lambda \le 1$;

}

}

Fig. 3. The Cluster Shrinkage algorithm.

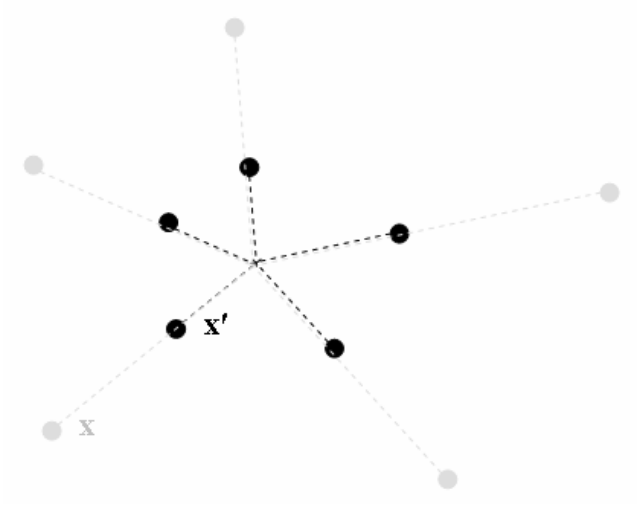

Fig. 4. The Cluster Shrinkage process.

The formula $\mathbf{x}' = \lambda\mathbf{c} + (1-\lambda)\mathbf{x}$ is actually a linear interpolation of the example $\mathbf{x}$ and the 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 as follows: $\mathbf{x}' = \lambda\left(\frac{1}{g}\sum_{h=1}^{g}\mathbf{c}^{(h)}\right) + (1-\lambda)\mathbf{x}$.

Before run TSVM, we do Cluster Shrinkage on source categories $S_1, S_2, ..., S_N$ as well as on master categories $C_1, C_2, ..., C_M$. We name this approach CS-TSVM.

CS-TSVM not only makes effective use of the objects from the source taxonomy like TSVM, but also makes effective use of the categorization of the source taxonomy in addition.

B. Analysis

Denoting the distance between two examples $\mathbf{u}$ and $\mathbf{v}$ by function $d(\mathbf{u}, \mathbf{v})$, we can get the following theorems about Cluster Shrinkage.

THEOREM 1. For any example $\mathbf{x}\in S$, suppose the center of S is $\mathbf{c}$, $\mathbf{x}$ becomes $\mathbf{x}'$ after Cluster Shrinkage, 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
$0 \le 1-\lambda \le 1,\ (1-\lambda)d(\mathbf{x},\mathbf{c}) \le d(\mathbf{x},\mathbf{c})$.

THEOREM 2. For any pair of examples $\mathbf{x}_1 \in S$ and $\mathbf{x}_2 \in S$, suppose the center of S is $\mathbf{c}$, $\mathbf{x}_1$ and $\mathbf{x}_2$ become $\mathbf{x}_1'$ and $\mathbf{x}_2'$ respectively after Cluster Shrinkage, 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
$d(\mathbf{x}_1', \mathbf{x}_2') = \|\mathbf{x}_1' - \mathbf{x}_2'\| = \|(\lambda\mathbf{c} + (1-\lambda)\mathbf{x}_1) - (\lambda\mathbf{c} + (1-\lambda)\mathbf{x}_2)\|$
$= \|(1-\lambda)(\mathbf{x}_1 - \mathbf{x}_2)\| = (1-\lambda)\|\mathbf{x}_1 - \mathbf{x}_2\| = (1-\lambda)d(\mathbf{x}_1, \mathbf{x}_2)$
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 theorems, we see that Cluster Shrinkage let all examples in a category move closer to the center, meanwhile become closer to each other. Because TSVM finds the maximum margin hyperplane (i.e. the thickest slab) in both training and test examples, making the examples in source category S closer to each other directs TSVM to avoid splitting S. In other words, Cluster Shrinkage guides TSVM to reserve the original categorization of the source taxonomy to some degree when do classification, as shown in Fig. 5.

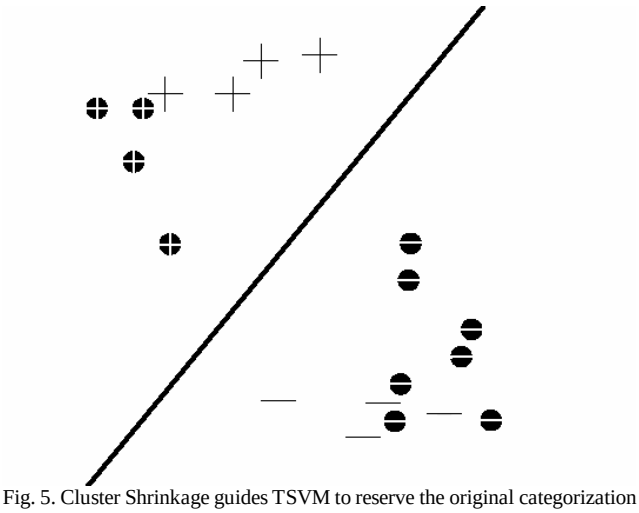

Fig. 5. Cluster Shrinkage guides TSVM to reserve the original categorization of the source taxonomy in some degree when do classification (cf. Fig. 2.).

On the other hand, we also do Cluster Shrinkage for master categories, to reduce TSVM's dependence on training examples and thus put more emphasis on taking advantage of the source taxonomy. If the number of training examples is small, we think this operation could help reduce the variance of the learned classifying functions.

Note that in inductive learning (e.g. SVM) setting, Cluster Shrinkage is not likely to work well, because the test examples (objects in the source taxonomy) have no influence on the classifying hyperplane. This thought has been confirmed by our experiments.

The 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 examples in one source category as a whole into a specific master category. 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 [1].

Another way to incorporate the categorization of the source taxonomy into classification 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 Cluster Shrinkage. 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 will get $\langle \mathbf{x}_1'' \bullet \mathbf{x}_2'' \rangle = 0$, while Cluster Shrinkage will get a more reasonable dot product function $\langle \mathbf{x}_1' \bullet \mathbf{x}_2' \rangle = \langle \mathbf{c}_1' \bullet \mathbf{c}_2' \rangle$.

VI. EXPERIMENTS

We conduct experiments with real-world web data, to demonstrate the advantage of TSVM over SVM as well as the advantage of CS-TSVM over TSVM, for taxonomy integration.

A. Datasets

We have collected 5 datasets from Google (*http:// www. google. com/*) and Yahoo (*http:// www. yahoo. com/*). 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 I
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 II
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 Fig. 6.

Association for Computing Machinery (**ACM**)
the world's first educational and scientific computing society for professionals and students.
www.acm.org/

Fig. 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, Yahoo, either of them (G ∪ Y), and both of them (G ∩ Y) respectively. The set of objects in G ∩ 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 [1].

TABLE III
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

B. *Tasks*

For each dataset, we pose 2 symmetric taxonomy integration tasks: G←Y (taking Google as the master taxonomy and Yahoo as the source taxonomy) and Y←G (taking Yahoo as the master taxonomy and Google as the source taxonomy).

For each task, we use the objects in G ∩ Y for experiments, because we know their categorization in both taxonomies [1]. We randomly split this set of objects into a training set and a test set. For all objects in the training set, we discard their source categories. For all objects in the test set, we hide their master categories from the learning algorithm during the training phase, and then compare their hidden master categories with the predictions of the learning algorithm during the test phase. For each task, we do such random split 5 times, each time allocating 20% objects to the training set and the rest 80% objects to the test set. It is common in practice that the training set is much smaller than the test set. Then we formulate each task as a classification problem, as described in §2.

C. *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 [8, 9]. 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.

D. *Measures*

As stated in §2, each task is formulated as a classification problem. To measure classification performance, we use the standard F-score (F_1 measure) [10]. 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. For each task, the F-scores are first computed on each individual class (master category), then averaged over all classes, finally averaged over all random train-test splits. In order to ensure the meaningfulness of evaluation, we do not take into account the F-scores on the classes with inadequate (less than 10) training or test examples.

E. *Settings*

We use SVMlight (available at *http:// svmlight. joachims. org/*) for the implementation of SVM / TSVM. We take linear

kernel, and accept all default values of parameters except "-j". We set the parameter "-j" to balance the cost of training errors on positive and negative examples.

The Cluster Shrinkage algorithm is very simple and very fast. It only requires one sequential scan to compute the cluster centers and another sequential scan to reposition the examples.

In all our CS-TSVM experiments, the CS parameter λ was fixed at 0.6. 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 CS-TSVM would be under-estimated.

F. Results

For each taxonomy integration task on each dataset, we try three approaches (SVM, TSVM, and CS-TSVM) and compare their performances measured by average *F*-scores.

The experimental results for G←Y tasks (integrating objects from Yahoo taxonomy into Google taxonomy) are shown in Table 4 and Fig. 7.

TABLE IV
EXPERIMENTAL RESULTS FOR G←Y TASKS

	SVM	TSVM	CS-TSVM
Book	0.4152	0.6026	0.7793
Disease	0.6065	0.7175	0.7576
Movie	0.3615	0.4664	0.5513
Music	0.4637	0.5482	0.6079
News	0.4844	0.6260	0.7345

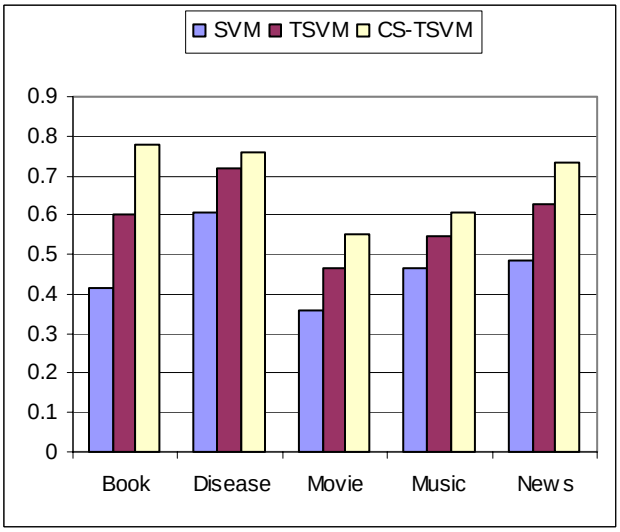

Fig. 7. Experimental results for G←Y tasks.

The experimental results for Y←G tasks (integrating objects from Google taxonomy into Yahoo taxonomy) are shown in Table 5 and Fig. 8.

TABLE V
EXPERIMENTAL RESULTS FOR Y←G TASKS

	SVM	TSVM	CS-TSVM
Book	0.4022	0.5777	0.7630
Disease	0.6044	0.7361	0.7725
Movie	0.3658	0.4883	0.5857
Music	0.5391	0.6077	0.7526
News	0.4863	0.6338	0.7158

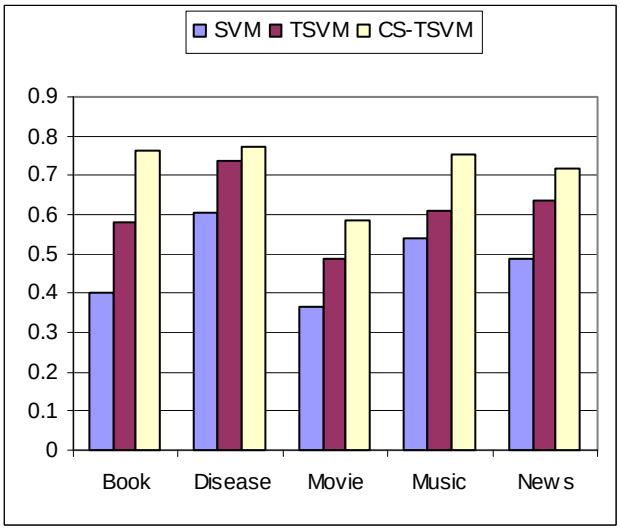

Fig. 8. Experimental results for Y←G tasks.

These experimental results show TSVM outperforms SVM consistently and significantly, which implies that making effective use of objects from the source taxonomy is helpful. Moreover, these experimental results also show CS-TSVM outperforms TSVM consistently and significantly, which implies that exploiting categorization of the source taxonomy is a plus.

VII. RELATED WORK

Our work is inspired by the exploration of Agrawal and Srikant. They have proposed an enhanced Naïve Bayes algorithm for taxonomy integration in [1].

The Naïve Bayes (NB) algorithm is a well-known text classification technique [4]. NB tries to fit a generative model for documents using training examples and apply this model to classify test examples.

Although the enhanced NB algorithm has been shown to work well for taxonomy integration, we think an approach based on SVM but not NB is still interesting and attractive. In contrast to NB, SVM is a discriminative classification method, i.e., SVM does not posit a generative model but seek to find the best classifying function directly. It is generally believed that SVM is more promising than NB for text classification [11, 12], and SVM has been successfully applied to many other kinds of data such as images [5]. Moreover, our proposed CS-TSVM

approach has the potential to be extended to achieve non-linear and hierarchical classifications.

The empirical comparison between the enhanced NB algorithm and CS-TSVM is left for future work.

VIII. Conclusion

We have presented a new technique, CS-TSVM, for integrating objects from a source taxonomy into a master taxonomy. Our technique based on transductive learning enhances the standard SVM classifiers by exploiting the implicit knowledge in the source taxonomy. Our experiments using real-world web data indicate that the proposed approach can result in large improvements in taxonomy integration performance.

Our work suggests several natural research directions. What is the best way to find the optimal value of parameter λ for Cluster Shrinkage? How can other transductive learning algorithms exploit the implicit knowledge in the source taxonomy? And which transductive learning algorithm is most suitable for this task?

References

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

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

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

[4] T. Mitchell, *Machine Learning*, international ed. Singapore: McGraw Hill, 1997.

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

[6] V. N. Vapnik, *The Nature of Statistical Learning Theory*, 2nd ed. New York, NY: Springer-Verlag, 2000.

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

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

[9] T. Joachims, "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, pp. 137-142.

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

[11] S. Dumais, J. Platt, D. Heckerman, and M. Sahami, "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, pp. 148-155.

[12] Y. Yang and X. Liu, "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, pp. 42-49.

Dell Zhang is a research fellow in the National University of Singapore under the Singapore-MIT Alliance (SMA). He has received his BEng and PhD in Computer Science from the Southeast University, Nanjing, China. His primary research interests include machine learning, data mining, and information retrieval.

Wee Sun Lee is an Assistant Professor at the Department of Computer Science, National University of Singapore, and a Fellow of the Singapore-MIT Alliance (SMA). He obtained his Bachelor of Engineering degree in Computer Systems Engineering from the University of Queensland in 1992, and his PhD from the Department of Systems Engineering at the Australian National University in 1996. He is interested in computational learning theory, machine learning and information retrieval.