Web Taxonomy Integration with Hierarchical Shrinkage Algorithm and Fine-Grained Relations

Chia-Wei Wu[a], Richard Tzong-Han Tsai[a], Cheng-Wei Lee[ab],Wen-Lian Hsu[ab]

{cwwu,thtsai,aska,hsu}@iis.sinica.edu.tw

[a]*Institute of Information Science, Academia Sinica, Taipei, Taiwan*

[b]*Department of Computer Science, National Tsing-Hua University, Hsingchu, Taiwan*

Elsevier use only: Received date here; revised date here; accepted date here

Abstract

We address the problem of integrating web taxonomies from different real Internet applications. Integrating web taxonomies is to transfer instances from a source to target taxonomy. Unlike the conventional text categorization problem, in taxonomy integration, the source taxonomy contains extra information that can be used to improve the categorization. The major existing methods can be divided in two types: those that use neighboring categories to smooth the document term vector and those that consider the semantic relationship between corresponding categories of the target and source taxonomies to facilitate categorization. In contrast to the first type of approach, which only uses a flattened hierarchy for smoothing, we apply a hierarchy shrinkage algorithm to smooth child documents by their parents. We also discuss the effect of using different hierarchical levels for smoothing. To extend the second type of approach, we extract fine-grain semantic relationships, which consider the relationships between lower-level categories. In addition, we use the cosine similarity to measure the semantic relationships, which achieves better performance than existing methods. Finally, we integrate the existing approaches and the proposed methods into one machine learning model to find the best feature configuration. The results of experiments on real Internet data demonstrate that our system outperforms standard text classifiers by about 10 percent.

Keywords: Web Taxonomy Integration, Shrinkage Algorithm, Text Categorization.

1. Introduction

Web taxonomy is a hierarchical collection of classes and documents. A number of taxonomies have been developed for various services, such as electronic auctions, online book stores, electronic libraries, and search engine crawlers. Yahoo! and Google Directories are two good examples of these applications. Such taxonomies encourage serendipitous searching for information, improve

navigation among related topics, and enhance full-text searching.

Information sharing between taxonomies is becoming an increasingly important activity on the Internet. Google News [1] is a good example, shown in Fig. 1, because, instead of offering news content, it provides an algorithm that collects news articles from various web sites and categorizes them based on its own classification method. The value of Google news is not necessarily the information per se, but the fact that the integration process allows the information to be read easily and improves access to more news sources.

Fig 1 An example of information sharing between taxonomies, Google News integrates other news web site to its navigate categories.

The B2B and B2C commercial web site is another typical applications which needs intelligent solution for sharing information between existing structures and personal data[2]. For example, Shopzilla[1], a B2C web site, integrates the products of many commercial web sites into its own schema for selling purpose. For such web sites, integrating product schemas is a crucial process in their business service.

In this paper, to integrate web taxonomies, we select one taxonomy as the source and another as the target, and then transfer web pages from the source to the target. Under this scenario, the web taxonomy integration problem can be simplified to a document categorization task which categorizes documents from source to target given the source taxonomy information. [3] Unlike standard text categorization tasks, in web taxonomy integration, more information can be used, such as the structure of the taxonomy and the source class label of documents. How to exploit such information effectively in order to achieve more accurate categorization is a crucial issue that has been addressed by many web taxonomy integration approaches [3-6].

Several approaches, such as the enhanced Naïve Bayes classifier [3], Co-Bootstrapping [4], and SVM-based methods [5], exploit the semantic overlap of corresponding categories to improve the categorization performance. The basic idea of these approaches is that if the topics of classes A and B are known to be very similar, then there should be a large number of documents in A that also belong to B. For example, if an article belongs to the ***Movie*** category of BBC news, it probably also belongs to the ***Movie*** category of Google news. To measure the distance between categories, we use the cosine similarity to determine the relationships and incorporate the information into a discriminative machine learning model. Another method of web taxonomy integration uses the term vectors of neighboring categories [5] to smooth a document's term vector with proper weights.

However, the above approaches do not consider the effect of the hierarchical structure, which could provide valuable information for web taxonomy integration. In this paper, we extend existing methods by adding information about hierarchies. Specifically, we apply a hierarchical shrinkage algorithm that smoothes the term frequencies by considering the hierarchy of the classes. We also discuss the impact of using categories in different levels for smoothing. For example, suppose a web page is an instance of the ***professional_sport*** category and ***professional_sport*** is a child of the ***sports*** category. Both categories are this web page's parents and can be used for smoothing. The advantage of using an upper-level category is that more terms can be used for smoothing, while the advantage of using a lower-level category is that the topic is more coherent with the web page itself. We discuss this point in detail in Section 4.

Finally, we integrate all the proposed features with those of previous works into one system to find the best configuration. The results demonstrate that the system with multiple features outperforms system with single feature type.

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

The remainder of this paper is organized as follows: In Section 2, we discuss related works. In Section 3, we present our approach and the machine learning model. In Section 4, we describe our experiments, including the dataset, experiment design, and results. We then close the paper with some concluding remarks in Section 5.

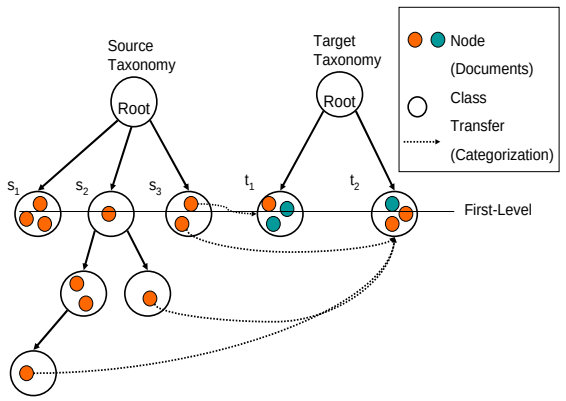

Fig 2 The concept of web taxonomy integration

1.1. Web taxonomy integration

We first define the web taxonomy integration process. The concept of web taxonomy integration, illustrated in Fig.1 [3-5], can be formulated as the assignment of documents from a source taxonomy to a target taxonomy. We use first-level categorizes as our categorization targets; therefore, as evaluate the categorization accuracy, the categorization target is the first-level categories of the taxonomy and the interior documents will be treated as the leaf of first category no matter which level they located. The terms used in this task are as follows:

A source taxonomy, ***S***, with a set of classes, $s_1, s_2, \ldots, s_i, \ldots, s_n$, each of which contains a set of documents.

A target taxonomy, ***T***, with a set of classes, $t_1, t_2, \ldots, t_i, \ldots, t_m$, each of which contains a set of documents.

For each document x in ***S***, our task is to assign x to the appropriate target category in ***T***. Thus, in Figure 2, documents from a source taxonomy are categorized into a target taxonomy.

2. Related Work

We now introduce some existing web taxonomy integration approaches, many of which use information from the source taxonomy, or the relationships between the source and target taxonomies to enhance the categorization accuracy. The Enhanced Naïve Bayes algorithm (ENB), proposed by Agrawal and Srikant [3], involves two steps. First, ENB uses a Naïve Bayes (NB) classifier [7] to estimate the degree of overlap between the source and target categories. For example, if the categorization result of an NB classifier shows that 60% of the documents are categorized from class A to class B, then the semantic overlap score between A and B is 0.60. In the second step, ENB classifier combines the categorization score of a NB classifier and the score computed in the first step with a parameter w. The formula of the ENB algorithm can be written as follows:

$$p(c_i \mid d, s) = \frac{p(c_i \mid s)\, p(d \mid c_i)}{p(d \mid s)} \tag{1}$$

$$p(c_i \mid s) = \frac{|c_i| \times N(s, c_i)^{\omega}}{\sum_{j=1}^{n} \left(|c_j| \times N(s, c_j)^{\omega} \right)} \tag{2}$$

where $p(c_i|d,s)$ denotes the probability of category c_i, given document d and source category s; $p(c_i|s)$ denotes the probability of a document's target class label given its source category. The taxonomy and $p(c_i|s)$ can be estimated by an NB classifier, i.e., formula two. If w is set to zero, then the categorization result of a ENB classifier will be exactly the same with the result of a NB classifier. In essence, ENB measures the relationships between categories and uses that information to enhance the categorization result.

Ichise [8], proposed a category-based integration method that merges similar category pairs, but it does not categorize documents individually. The integration performance can be enhanced by fully exploiting the category information. The disadvantage of this approach is that it completely ignores the influence of specific documents and the discrepancy between categories in the source and target taxonomies. Sarawagi et. al. [6] use an iterative co-training approach based on the Naïve Bayes classifier to obtain more robust term probability distributions. The co-training procedure can be seen as an EM

algorithm. In the E-step, the documents in the source taxonomy are labeled by initial classifiers. Then, in the M-step, the labeled documents are used to retune the parameters and train new classifiers.

The above approaches are based on generative model; however, a discriminative model usually performs better in text categorization. Hence, several works extend web taxonomy integration to use discriminative models. Co-Bootstrapping (Sarawagi et. al. [6] and Zhang and Lee [4]) uses a classifier to predict a document's source class label and then encodes that information as a feature in machine learning models. This approach can be thought of as using a classifier to measure the similarity between two corresponding categories. The underlying ideas of these discriminative classifier-based approaches are similar in measuring the similarity between classes for improving document categorization.

Another method exploits the term vector of neighboring nodes for smoothing. For example, suppose there is a source taxonomy instance in the "Action movie" category and the class label of its target taxonomy is "Movie". In this case, it would be reasonable to expect that other instances in "Action movie" could provide some information that would help categorize that instance. To this end, the cluster shrinkage algorithm (Zhang and Lee [5]) smoothes the word vector of a document by using the word vectors of neighboring nodes in the same class. A flattened hierarchy structure is used as the information source for smoothing.

In addition to automatic integration approaches, there are some semi-automatic integration methods, such as QOM [9] and PROMPT [10].

3. Method

First, we introduce the features used to exploit information in the source and target taxonomies to enhance the categorization performance, and then present the machine learning model used for training and testing.

3.1. Features

3.1.1. Term Frequency (TF)

In text categorization classifiers, the term frequency feature is usually used as the baseline system. A distinct term frequency feature is initiated for each term-category combination. If a term *w* occurs most frequently in a category, *t*, we would expect the weight corresponding to the *w*-*t* pair to be higher than that of the term-category combination, According to Nigam et. al. [11], in text classification, using real numbers for the feature values to represent the frequency achieves a better performance than only using binary feature values to represent the appearance of terms. For each term, *w*, and a category, *t*', in the target taxonomy, ***T***, we formally define the term frequency feature as:

$$f_{w,t'}(x,t) = \begin{cases} \dfrac{N(w,x)}{N(x)} & if\ t = t' \\ 0 & otherwise \end{cases} \quad (3)$$

where $N(w, x)$ is the number of times a term w appears in a document x, and $N(x)$ is the number of terms in x.

3.1.2. Shrinkage Term Frequency (STF)

In practice, the number of terms contained in a document varies, and is relatively small compared to that of a category. Therefore, the values of most term frequency features will be zero. Smoothing is a popular statistical technique used to alleviate this problem since it provides a more robust term frequency distribution for sparse data distribution. In a taxonomy, a leaf (a document or an item) often have one or more ancestor categories which can be used for smoothing. For utilizing the hierarchy for smoothing, we adopt the hierarchical shrinkage algorithm [7] to smooth the terms in a document by its ancestors. The algorithm was originally proposed for the generative probabilistic model. In this paper, we apply it to the discriminative model. For each word *w* and category *t* in the target taxonomy, we define the shrinkage term frequency (STF) feature as follows:

$$f_{w,t'}(\boldsymbol{x},t,\boldsymbol{c})=\begin{cases}\alpha\dfrac{N(w,\boldsymbol{x})}{N(\boldsymbol{x})}+ & \\ (1-\alpha)\sum_{l=1}^{\mathrm{L}}\beta^{l}\dfrac{N(w,\boldsymbol{c}^{l})}{N(\boldsymbol{c}^{l})} & if\ t=t' \\ 0 & otherwise\end{cases} \quad (4)$$

where $N(w, \boldsymbol{x})$ denotes the number of times a term w appears in x's category t; β^{l} denotes the interpolation weights between levels and the sum of β^{l} for all is one. l denoted as the level number, for example, l equal to 1 means the top-level and l equal to L means the bottom-level. $N(w, c^{l})$ is the number of times a word w appears in c^{l}, the source category in level l. α denotes the weight that controls the strength of the smoothing effect.

For discussing the smoothing effect result from different levels, we apply bottom-level categories and first-level categories in smoothing respectively. For only using bottom-level in STF, we let $\beta^{l}=0$ if $l \neq$ L (bottom-level) while for applying first-level for STF let $\beta^{l}=0$ if $l \neq 1$ (first-level).

3.1.3. Cosine Similarity

As mentioned in Section 2, using the similarity of document terms to facilitate categorization is based on the idea that, if the term vectors of two corresponding categories are very similar, then more their documents could overlap. The cosine similarity measure is one of the most popular algorithms for measuring the distance between documents' term vectors. We use the cosine similarity to measure the distance of documents, which is then used as a feature in our machine learning model. The cosine similarity feature function can be written as follows:

$$f_{t',s'}(t,s)=\begin{cases}\mathrm{Sim}(t,s) & if\ t=t'\quad s=s' \\ 0 & otherwise\end{cases} \quad (5)$$

where Sim(t,s) is the cosine similarity function used to compute the cosine value of the term vectors of t and s.

Like shrinkage term frequency features, for discussing the effect of applying different levels in cosine similarity features, we apply bottom-level categories and first-level categories respectively.

3.1.4. Category Labels of the Source Taxonomy (CLST)

The information in the source and the target taxonomies can also be exploited by cross-training, as proposed by Sarawagi et al.[6] and Zhang [4]. In the training phase, documents in the source taxonomy are used to train a multi-class classifier, which is then used to tag the documents in the target taxonomy. Next, each target taxonomy document is tagged with a source class label to indicate the category label feature. For each category t' in the target taxonomy, and each category s' in the source taxonomy, we define the CLST feature function as:

$$f_{t',s'}(t,s)=\begin{cases}1 & if\ t=t'\quad s=s' \\ 0 & otherwise\end{cases} \quad (8)$$

The CLST feature allows us to measure how many documents appear in both two categories. If most documents of the target category t are classified into a certain source category s, then the documents in s may be more likely to be classified into t.

3.2. Maximum Entropy Model

To implement our approach, we use the Maximum Entropy (ME) model [12], a statistical modeling technique for estimating the conditional probability of a target label based on the given information. ME computes the probability, $p(o|h)$, where o denotes all possible category labels from the outcome space, and h denotes all possible distinct features from the feature space. In the web taxonomy integration task, h can be viewed as all information related to the current document that can be derived from documents in the taxonomy, and the outcome can be viewed as the target category label. The computation of $p(o|h)$ in ME depends on a set of features that are helpful for making predictions about the outcome.

Given a set of features and a training set, the ME estimation process produces a model, in which every feature f_i has a weight λ_i. Following Berger [12], we compute the conditional probability as:

$$P(o \mid h) = \frac{1}{Z(h)} \exp\left(\sum_i \lambda_i f_i(h,o) \right), \quad (9)$$

$$z(h) = \sum_c \exp\left(\sum_i \lambda_i f_i(o,h) \right). \quad (10)$$

The probability is derived by multiplying the weights of the active features (i.e., those f_i $(h,o) = 1$). The weight, λ_i, is estimated by a procedure called limited-memory BFGS [13], a quasi-newton algorithm improves the estimation of weights iteratively. The ME estimation technique guarantees that, for every f_i, the expected value of λ_i will equal the empirical expectation of λ_i in the training corpus.

ME has a proven competitive performance in various tasks, including part-of-speech tagging [14], named entity recognition [15], English parser [16], prepositional phrase attachment [17], and text classification [11]. As noted in [15], ME allows users to focus on finding features that characterize the problem, while leaving feature weight assignment to the ME estimation routine.

Although the ME model is our choice in this paper, other machine learning algorithms, such as Support Vector Machine [18], Conditional Random Fields [19], or Boosting [20], could also be used in our approach to improve taxonomy integration. In other words, the machine learning model is only used as a platform to integrate various methods.

3.3. System Overview

We show the data processing flows of integrating taxonomies in Fig 3. The document text and other information will be sent to the feature generator to generate feature value for the machine classifiers. There are seven feature types in the feature generator, including TF, hierarchical STF, bottom-level STF, first-level STF, bottom-level cosine similarity, first-level cosine similarity, and CLST. There are two types of features, cosine similarity and CLST, refer to the information of target and source taxonomy. Therefore, there are two information flows link from both target and source to them. Other features only need information of source. A trained machine learning classifier will generate the categorization results (the category labels in target taxonomy) based on the input features.

Fig 3 Data Processing Flows

4. Experiment & Results

In this section, we describe the data used in the experiments and the experimental settings, followed by an evaluation of the experimental results.

4.1. Datasets

We collected five datasets from the Google and Yahoo! directories to evaluate our approach. Each dataset contained a sub-taxonomy of the Google Directory and the corresponding sub-taxonomy in the Yahoo! Directory. In Table 1, each row shows the dataset name, the number of links (the web pages) within each sub-taxonomy, and the number of shared links as they exist in both Google and Yahoo! directories.

Shared links were identified by their URLs and used as test data, while the remaining links were used as training data. Only a small number of links are shared by the two web taxonomies.

Table 1. The Datasets

	Google Directory	# of links	# of classes	Yahoo! Directory	# of links	# of classes	# of Shared links
Books	Shopping/ Publications/ Books/	5544	42	Business_and_economy /shopping_and_services /books/	7348	39	626
Music	Top/Arts/ Music/Styles	9903	50	Entertainment/Music/ Genres	1787	25	1308
Garden	Shopping/Home _and_Garden	10048	37	Business_and_economy /shopping_and_services /home_and_garden/	2912	18	601
Outdoor	Top/Recreation/ Outdoors/	10137	37	Recreation/Outdoors	5009	65	853
Finance	Business/ Financial_Services	10446	43	Business and Economy Business_to_Business/ Financial_Services	3016	20	946
Travel	Top/Recreation/ Travel/ Specialty_Travel/	9421	50	Recreation/Travel	6864	49	1981
Total #		55499	259		33251	216	6315

In the Google and Yahoo! directories, each instance (document) contains the web page's title, URL, and description, as shown by the following example:

Title: Spider-Man 3

Snippet: Official site for the motion picture.

Link*:* ww.sonypictures.com/movies/**spiderman**3/site/

Fig 4 Google Directory page and the Information of an instance.

We used the titles and snippets for experiment. All documents were pre-processed by removing the stop words and stemming.

4.2. Experiment design and setting

We use the documents from Yahoo! (excluding the shared links) for training and categorize documents from Google into Yahoo! and vice versa.
We define the classification accuracy as follows:

$$\text{Acc.} = \frac{\#\text{of documents correctly classified}}{\#\text{of documents}}$$

In the NB and ENB experiments, we implement the NB and ENB modules. The parameter w with the best performance in formula 2 of ENB is selected from a series of numbers: {0, 1, 3, 10, 30, 100, 300, and 1,000}, and the smoothing parameter [3] of the NB and ENB classifier is set to 0.1. We use the Maximum Entropy Toolkit [21] to implement the ME-based approaches. To compare our approach with normal text classification methods, we implement the ME-based text classification algorithm proposed in [11]. The parameter α used in the shrinkage-term frequency (STF) is set to 0.5 and β, another parameter in STF, is trained by randomly selecting one-tenth of the documents of the test dataset.

4.3. Experiment Results

Table 2 Experiment results of Naïve Bayes, Enhanced Naïve Bayes, Maximum entropy text classifier (ME-Basic), and our approach using the best feature configuration (ME-Best).

		NB	**ENB**	**ME-Basic**	**ME-Best**
G to Y	Books	36.22	58.20	60.12	75.05
	Music	17.38	32.10	62.19	79.26
	Garden	54.79	70.7	82.30	87.17
	Outdoor	39.38	60.62	71.54	76.90
	Finance	37.25	46.63	72.32	75.47
	Travel	29.15	35.90	73.17	82.92
	Average	**35.69**	**50.69**	**70.27**	**79.46**
Y to G	Books	34.79	49.78	53.18	67.37
	Music	34.25	41.76	46.30	68.45
	Garden	55.87	60.88	71.18	76.35
	Outdoor	50.81	66.08	68.8	75.26
	Finance	32.29	35.41	58.64	66.67
	Travel	40.35	38.17	67.32	70.29
	Average	**41.39**	**48.68**	**60.90**	**70.73**

We implement ME-Basic following Nigam et. al.'s work[11] using TF feature in ME. ME-basic can provide the performance of an ordinary text classifier without any taxonomy integration features. We implement ME-basic ME-Best uses the best feature configuration set, including TF, STF, and cosine similarity. From the results listed in Table 2, we observe that ME-Best outperforms the other three approaches. In comparison of ME-Best and ME-Text, we can observe that the improvement brought by our designed features for taxonomy integration is significant about 10%. The improvement of ME-Best over NB is almost 45% percent. This improvement is not only brought by the extra features used by ME-Best, but also affected by the intrinsic difference between the NB and ME models.

Table 3. Experimental results of the shrinkage algorithms. STF first-level only uses first level categories for smoothing, while STF bottom-level only uses categories in the bottom-level. Hierarchy STF integrates the categories of all levels with different weights.

		ME Basic	**STF First-Level**	**STF Bottom-Level**	**Hierarchy STF**
G to Y	Books	60.12	60.12	67.89	69.12
	Music	62.19	62.80	71.34	72.12
	Garden	82.30	84.87	85.89	85.34
	Outdoor	71.54	71.2	74.3	76.29
	Finance	72.32	67.92	71.06	71.37
	Travel	73.17	74.39	79.26	84.12
	Average	**70.27**	**70.22**	**74.96**	**76.39**
Y to G	Books	53.18	54.16	57.91	58.08
	Music	46.30	53.02	58.38	64.42
	Garden	71.18	72.41	77.09	76.79
	Outdoor	68.8	69.39	74.67	73.4
	Finance	58.64	60.49	66.67	64.02
	Travel	67.32	63.36	71.28	71.28
	Average	**60.90**	**62.14**	**67.67**	**67.99**

In Table 3, in comparison of the ME-Basic results and those of other models using STF features, we can see that using STF-Bottom-level and Hierarchy-STF increases the classification accuracy by at least 4%. This finding suggests that smoothing with ancestor categories term vectors could provide valuable information for assigning documents.

The results in the second and third columns of Table 3 show that bottom-level STF outperforms the first-level. The topics of two corresponding categories in directories are not always exactly the same. In fact, topics often only overlap partially. This result can show that although upper-level categories provide more terms for smoothing, the classification performance is undermined because the topics that the terms represent are often too general. In contrast, the terms in the bottom-level are more coherent, so they are better for smoothing than upper-levels.

Hierarchical STF achieves the best performance between using different levels STF. The parameter β, the interpolation weights between different level

categories, can successful enhance the utility of terms in levels for smoothing.

Table 4. The results of using the cosine similarity feature. Cosine similarity first-level uses the cosine value between the first-level categories as a feature, while Cosine similarity bottom-level uses bottom-level categories.

		ME Basic	**Cosine Similarity First-Level**	**Cosine Similarity Bottom-Level**
G to Y	Books	60.12	70.55	71.78
	Music	62.19	70.73	71.95
	Garden	82.30	85.12	85.79
	Outdoor	71.54	77.07	75.93
	Finance	72.32	73.58	77.98
	Travel	73.17	76.82	74.39
	Average	**70.27**	**75.65**	**76.30**
Y to G	Books	53.18	62.97	66.23
	Music	46.30	65.10	64.42
	Garden	71.18	71.67	75.61
	Outdoor	68.8	72.23	74.23
	Finance	58.64	67.28	69.75
	Travel	67.32	63.36	65.34
	Average	**60.90**	**67.10**	**69.26**

Table 4 shows the results of using cosine similarity features. Both the cosine similarity first-level and the bottom-level features outperform ME-Basic for all categories.

In the Cosine similarity First-Level configuration, we only use the cosine similarity values between the first–level source categories and their corresponding target categories as features, while in the Cosine similarity Bottom-Level, we use the bottom-level categories. The results show that using the similarity between the bottom-level source categories and target categories as features is more effective than using the first-level categories. These results are similar to those in Table 3 that using the bottom-level yields better results than the top-level.

Table 5 Experiment results using predicted category label features

		ME-Basic	**CLST**
G to Y	Books	60.12	67.08
	Music	62.19	71.95
	Garden	82.30	86.15
	Outdoor	71.54	76.1
	Finance	72.32	69.25
	Travel	73.17	78.04
	Average	**70.27**	**74.76**
Y to G	Books	53.18	60.03
	Music	46.30	51.67
	Garden	71.18	71.92
	Outdoor	68.8	71.15
	Finance	58.64	71.69
	Travel	67.32	67.32
	Average	**60.90**	**65.63**

Table 5 shows the results of using category label features for classification. Compared with ME-Basic, the average improvement of applying category label features is about 4~5%. Both category label features and ENB initially use a classifier to measure how many documents appear in both categories in the source and target taxonomies. Like the ENB result, this result shows that measuring the document overlap between categories provides useful information for web taxonomy integration.

Table 6. The results of different feature configurations

		Books	Music	Garden	Outdoor	Finance	Travel	Average
G to Y	Basic (Term Frequency)	60.12	62.19	82.30	71.54	72.32	73.17	**70.27**
	+Hierarchical Shrinkage (STF)	69.12	70.12	85.34	76.29	71.37	84.12	**76.39**
	+Cosine Similarity Bottom-Level	75.05	79.26	87.17	76.90	75.47	82.92	**79.46**
	+Category Label	73.82	75.0	85.64	76.45	72.95	76.82	**76.78**
Y to G	Basic (Term Frequency)	53.18	46.30	71.18	68.8	58.64	67.32	**60.90**
	+Hierarchical Shrinkage (STF)	58.08	64.42	76.79	73.4	64.02	71.28	**67.99**
	+Cosine Similarity Bottom-Level	67.37	68.45	76.35	75.26	66.67	70.29	**70.73**
	+Category Label	67.70	68.45	75.37	75.40	63.58	66.33	**69.47**

Table 6 details the results of different feature configuration sets and the cumulative performance of the features listed in the second column. The "+" sign following a feature name indicates that the feature is added incrementally. The performance of each row is contributed by the feature name of that row and the features showing above. With the baseline feature, TF, we add the STF feature, cosine similarity bottom-level feature, and CLST feature incrementally. Adding hierarchy shrinkage and cosine similarity features improves the accuracy by almost 10%. Nevertheless, the accuracies are not improved on all datasets after adding category label features. The cosine similarity relies on term vectors; meanwhile, the category labels rely on the predicted category labels which are also decided by term vectors. Therefore, the information used by these two features is duplicated. For example, if all the documents in a category of the target taxonomy are tagged as one specific category of the source taxonomy, then it would be reasonable to assume that their term vectors would also have a high cosine value.

The result of integrating the best feature set is better than single feature. This demonstrates that each feature uses different types of information and makes distinct contribution to the categorization performance. Finally, we list the performance of each feature and their combinations in Fig 5.

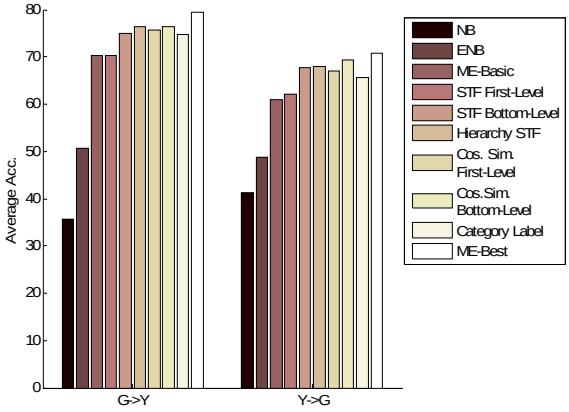

Fig 5 The average accuracy of features and their configurations.

5. Conclusion

We have proposed an approach that uses two types of information referred to source taxonomies to improve taxonomy integration. Unlike previous works, which only consider a flattened hierarchy as the information source, we use a hierarchical shrinkage algorithm to smooth the term vector of a child document by its ancestor's category. Another approach is using category information, the semantic relationships between categories. To obtain category information, we not only implement the previous approaches that use predicted class labels, but also propose a feature

with better performance that uses the cosine similarity to measure the semantic relationships between categories. Finally, we integrate previous and our proposed features into one model, and then select the best configuration for classification

For evaluating the feasibility to real applications, the proposed approach was tested using real Internet data. The experiment results show that our model, which incorporates source taxonomy as its features, outperforms the baseline system that only uses term frequency of documents in the target taxonomy to train classifiers. The results also demonstrate that lower-level categories provide more precise information than upper-level categories for shrinkage algorithm.

In the future, more information, such as web resources, a third taxonomy, or existing taxonomies could be incorporated into our approach. It is also interesting to assess whether the proposed approach could be used in other applications, especially the real internet applications.

References

[1] "Google News http://news.google.com/."

[2] D. Fensel, Y. Ding, B. Omelayenko, E. Schulten, G. Botquin, M. Brown, and A. Flett, "Product data integration in B2B e-commerce," *Intelligent Systems, IEEE*, vol. 16, pp. 54- 59, 2001.

[3] R. Agrawal and R. Srikant, "On Integrating Catalogs," *Proceedings of the Tenth International Conference on World Wide Web*, pp. 603 - 612, 2001.

[4] D. Zhang and W. S. Lee, "Machine Learning for IR: Web Taxonomy Integration through Co-bootstrapping," *Proceedings of the 27th Annual International Conference on Research and Development in Information Retrieval* pp. 410 - 417 2004.

[5] D. Zhang and W. S. Lee, "Web Taxonomy Integration Using Support Vector Machines," *Proceedings of the Thirteenth International Conference on World Wide Web*, pp. 472 - 481, 2004.

[6] S. Sarawagi, S. Chakrabarti, and S. Godbole, "Cross-Training: Learning Probabilistic Mappings between Topics.," presented at Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2003.

[7] A. McCallum, R. Rosenfeld, T. M. Mitchell, and A. Y. Ng, "Improving Text Classification by Shrinkage in a Hierarchy of Classes," *Proceedings of the Fifteenth International Conference on Machine Learning*, pp. 359 - 367 1998.

[8] R. Ichise, H. Takeda, and S. Honiden, "Integrating Multiple Internet Directories by Instance-Based Learning," presented at International Joint Conference on Artificial Intelligence, 2003.

[9] M. Ehrig and S. Staab, "QOM - Quick Ontology Mapping," presented at International Semantic Web Conference, 2004.

[10] N. F. Noy and M. A. Musen, "PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment," *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, 2000.

[11] K. Nigam, J. Lafferty, and A. McCallum, "Using Maximum Entropy for Text Classification," presented at In IJCAI-99 Workshop on Machine Learning for Information Filtering, 1999.

[12] A. Berger, S. A. D. Pietra, and V. J. D. Pietra, "A Maximum Entropy Approach to Natural Language Processing," *Computer Linguistic*, vol. 22, pp. 39-71, 1996.

[13] D. C. Liu and J. Nocedal, "On The Limited Memory Bfgs Method For Large Scale Optimization," *Math. Programming*, vol. 45, pp. 503 - 528, 1989.

[14] A. Ratnaparkhi, "A Maximum Entropy Model for Part-Of-Speech Tagging," *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 133-142, 1996.

[15] A. Borthwick, " A Maximum Entropy Approach to Named Entity Recognition," *New York University*, 1999.

[16] E. Charniak, "A Maximum-Entropy-Inspired Parser," *Proceedings of the First Conference on North American Chapter of the Association for Computational Linguistics*, pp. 132 - 139, 2000.

[17] A. Ratnaparkhi, "Statistical Models for Unsupervised Prepositional Phrase Attachment," *Proceedings of the Thirty-sixth Conference on Association for Computational Linguistics*, vol. 2, pp. 1079 - 1085 1998.

[18] T. Joachims, "Text Categorization with Suport Vector Machines: Learning with Many Relevant Features," *Proceeding of Tenth European Conference on Machine Learning*, pp. 137-142, 1998.

[19] J. Lafferty, A. McCallum, and F. Pereira, "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data," presented at Proceeding

Eighteenth International Conference on Machine Learning, 2001.

[20] R. E. Schapire and Y. Singer, "BoosTexter: A Boosting-based System for Text Categorization," *Machine Learning*, vol. 39, pp. 135-168 2000.

[21] Z. Le, "Maximum Entropy Toolkit."