Computational Linguistics and Chinese Language Processing
Vol. 13, No. 4, December 2008, pp. 387-404

Feature Weighting Random Forest for Detection of Hidden Web Search Interfaces

Yunming Ye[*], Hongbo Li[*], Xiaobai Deng[*] and

Joshua Zhexue Huang[+]

Abstract

Search interface detection is an essential task for extracting information from the hidden Web. The challenge for this task is that search interface data is represented in high-dimensional and sparse features with many missing values. This paper presents a new multi-classifier ensemble approach to solving this problem. In this approach, we have extended the random forest algorithm with a weighted feature selection method to build the individual classifiers. With this improved random forest algorithm (*IRFA*), each classifier can be learned from a weighted subset of the feature space so that the ensemble of decision trees can fully exploit the useful features of search interface patterns. We have compared our ensemble approach with other well-known classification algorithms, such as SVM, C4.5, Naïve Bayes, and original random forest algorithm (*RFA*). The experimental results have shown that our method is more effective in detecting search interfaces of the hidden Web.

Keywords: Search Interface Detection, Random Forest, Hidden Web, Form Classification

1. Introduction

Hidden Web refers to the Web pages that are dynamically generated from searchable structured or unstructured databases. Different from the Publicly Indexable Web that is accessible through static hyperlinks, the pages in a hidden Web can only be obtained through queries submitted via the search interface to the databases containing data about the hidden Web. Search interfaces are usually encoded as HTML forms that need to be filled out and

[*] Shenzhen Graduate School, Harbin Institute of Technology, China
E-mail: yeyunming@hit.edu.cn; {wave118, dawndeng}@gmail.com
[+] E-Business Technolgy Institute, The University of Hong Kong, Pokfulam Road, Hong Kong
E-mail: jhuang@eti.hku.hk

[Received August 31, 2008; Revised February 23, 2009; Accepted February 16, 2009]

submitted by users to obtain information. On the Web, there are many different HTML forms, and many of them are not search interfaces (He, Patel, Zhang, & Chang, 2007). To extract useful information from hidden Web pages, effectively detecting the search interfaces is an essential step since the interface is the only entrance to the hidden Web. Therefore, we first need to find the entrance to the hidden database. The entrance (*i.e.*, search forms) is mixed with lots of non-search forms in HTML pages. Thus, it is very important to distinguish the two types of forms in order to enable the Hidden Web crawler to locate the entrance and extract information further.

Information extraction from the hidden Web has been a hot research topic (BrightPlanet.com, 2000) since the term "Hidden Web" was first coined (Florescu, Levy, & Mendelzon, 1998). Most previous work, however, has been focused on the problems of automatic query generation (Ntoulas, Zerfos, & Cho, 2005), form filling (Cavelee, Liu, & Probe, 2004), and wrapper generation for extracting structured information (Wang & Lochovsky, 2003), where detecting search interface was performed manually or by some heuristic methods. Using heuristic rules to find search forms is the simplest and most effective method (Raghavan & Garcia-Molina, 2001; Lage, Silva, Golgher, & Laender, 2004). As hidden Web sites adopt different search forms, though, it is time-consuming to compose different rules for different search forms. Employing machine learning and information retrieval techniques to learn classification models from the content of search forms and using the models to classify different forms automatically is a more desirable approach with respect to scalability and robustness. This approach regards search interface detection as a two-class classification problem. One example of this is using decision trees to build form classification models to detect search interface (Cope, Craswell, & Hawking, 2003).

Automatic search interface detection was first explored by Raghavan and Garcia-Molina in their hidden Web crawler HiWE (Hidden Web Exposer) (Raghavan & Garcia-Molina, 2001). Their crawling system used heuristic rules to detect the search entrance to the hidden database. Juliano P. Lage (Lage, Silva, Golgher, & Laender, 2004) used two heuristic rules to perform detection tasks. The first heuristic was the same as HiWE. The second one was to check whether the form contains the "password" HTML element or not. The disadvantage of this method, however, lies in that it does not have auto-leaning capability. Moreover, it is not robust and scalable to diverse hidden Web databases because the rules are too simple to match different form structures.

Cope *et al.* (2003)used a decision tree classification algorithm to detect search interfaces. This method usually generates long rules due to the large size of the feature space in the training set (the number of training samples is too small compared to the number of features). Therefore, it is prone to overfitting, and the classification precision is not satisfying.

Zhang *et al.* (2004) presented a best-effort parsing framework to address the problem of understanding Web search interfaces. The authors transformed search interfaces into a visual language under the hypothesis that automatic construction of search interfaces is guided by a hidden syntax. This hypothesis enables parsing as a principled framework to understand the semantic model of the visual language. The experimental results testified the effectiveness of their approach.

To summarize, little previous work has addressed the special characteristics of the search interface detection domain, for instance, large and diverse features, small size of training samples with many missing values, *etc*. The high dimension and sparse data of search interfaces present a tricky problem for the traditional single classifier approach. As collecting training samples (*i.e.*, HTML forms) is costly, the training data set is usually small, while the number of features in the learning space is relatively large, due to multi-type features existing in forms. It's difficult for a single classifier to fully exploit the rich feature space (very sparse in the data matrix). For many classification methods, the single classifier tends to be overfitting. To attack this problem, we propose a multi-classifier ensemble approach in this paper.

Our method is based on the random forest algorithm. A random forest model consists of a set of decision trees that are constructed by bootstrapping the training data. In our approach, we develop a weighted feature selection method to select a subset of features for each decision tree in the tree induction process. Classification is made by aggregating predictions of individual decision trees in the forest. Since each classifier is learned from a subset of the feature space, the ensemble approach can fully exploit the useful features in search forms. We have conducted experiments on several real data sets. The experimental results have shown that our random forest approach improves the classification accuracy in search interface detection.

The contributions of this paper can be summarized as follows:

1. We explored the random forest approach to attacking the problem of detecting search interfaces from the sparse feature space of hidden Webs where specific feature extraction and representation techniques were used.
2. We extended the random forest algorithm with a weighted feature selection method to select a subset of features for each decision tree. The new algorithm can automatically remove the noisy features in search forms so that decision tree classifiers can be learned from more representative subsets of the feature space.
3. We conducted experiments on real data sets to compare the improved random forest algorithm with other well-known classification algorithms, such as SVM and C4.5. The experimental results have shown that the new method is more effective in detecting search interfaces of the hidden Web.

The rest of this paper is organized as follows. In Section 2, we formalize the detecting search interface problem as a form classification problem and present the feature extraction techniques. Section 3 describes the improved random forest algorithm for form classification. Experimental results and analysis are presented in Section 4. Section 5 concludes this paper and presents our future work.

2. Feature Extraction for Form Classification

Search interface detection is a process of distinguishing the search forms of the hidden Web from non-search forms. It is a two-class classification problem in machine learning. This section describes how to extract form features from HTML pages and discusses the characteristics of the data matrix for form classification.

2.1 Feature Extraction Rules

An HTML form usually begins with the tag $< FORM >$ and ends with the tag $< /FORM >$. According to this rule, HTML forms can be extracted by searching the $< FORM >$ tag in HTML pages. Each extracted form is a sample in the training set. The features of each form are generated by parsing the corresponding $< FORM >$ HTML block.

HTML forms contain two kinds of features: one is the attributes of forms and elements, and the other is the statistics of those attributes. A form mainly contains four kinds of elements, that is, “INPUT”, “SELECT”, “LABEL” and “TEXTAREA”, which are the children elements of “FORM” element. Element “INPUT” contains several types, such as “text”, “hidden”, *etc*. The hierarchy of a form is shown in Figure 1. All of these elements contain a set of attributes, such as “name”, “value”, “size”, *etc*. The attributes of “form” elements are “method”, “action”, and “name”. Attribute “method” indicates the method for the form to submit query data, such as “POST” or “GET”. Attribute “action” indicates the address of the corresponding server of the form, and attribute “name” indicates the name of the form. Some elements and attributes can be removed because they are not useful for form classification, for instance “option”, “size”, “width”, *etc*. Besides, the statistics about the number of elements or attributes in each element can also be computed as important features.

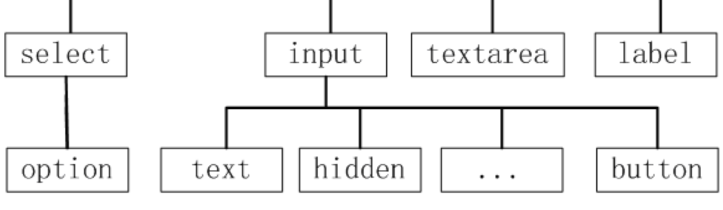
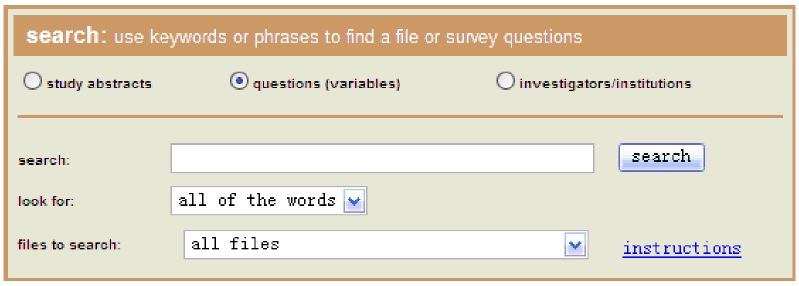


Figure 1. The hierarchy of form elements

Figure 2 shows an example of a search form. The form contains three elements: one “SELECT” element and two “INPUT” elements. There are three attributes in the “SELECT” element : “name” with value “and”, “size” with value “1”, and “width” with value “50”. It also contains several “OPTION” elements. The “size” and “width” attributes, along with the “OPTION” elements are not useful for form classification, so they can be removed. The corresponding HTML codes are shown in Figure 3.

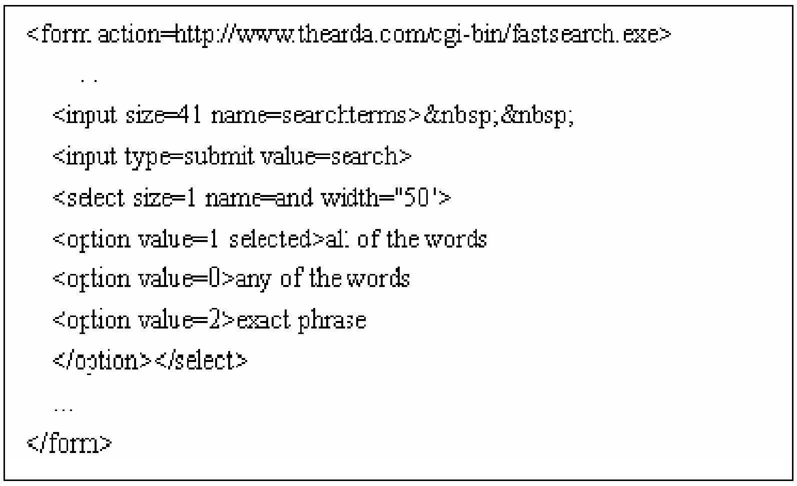


search: use keywords or phrases to find a file or survey questions

○ study abstracts [● questions (variables)] ○ investigators/institutions

search:		search
look for:	all of the words	
files to search:	all files	instructions

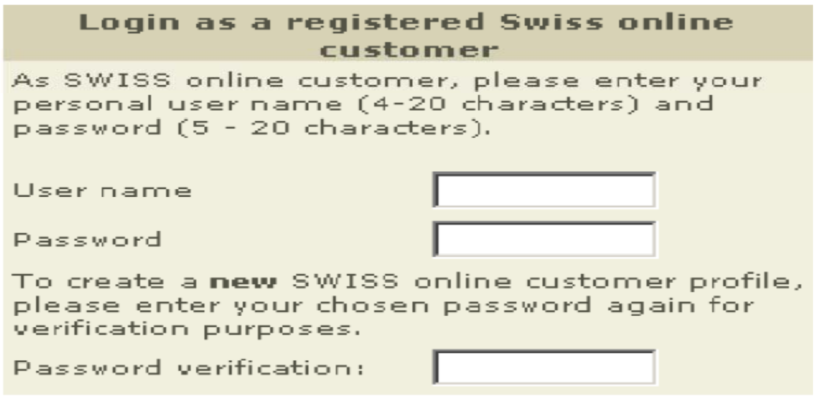
Figure 2. A search form



```
<form action=http://www.thearda.com/cgi-bin/fastsearch.exe>
      . .
   <input size=41 name=searchterms>&nbsp;&nbsp;
   <input type=submit value=search>
   <select size=1 name=and width="50">
   <option value=1 selected>all of the words
   <option value=0>any of the words
   <option value=2>exact phrase
   </option></select>
   ...
</form>
```

Figure 3. The HTML codes of the form in Figure 2

Figure 4 shows an example of a non-search form. The corresponding HTML codes are shown in Figure 5. According the feature extraction rules, the useful form elements in this form include “INPUT”, “LABEL”, and “FORM”, which can be used to compose the feature space. Elements “TABEL”, “FONT”, and “TR” can be removed because they are not useful.



Login as a registered Swiss online customer

As SWISS online customer, please enter your personal user name (4-20 characters) and password (5 - 20 characters).

User name	
Password	

To create a **new** SWISS online customer profile, please enter your chosen password again for verification purposes.

Password verification:	

Figure 4. A non-search form

```
<form name="DATES"
action="/servlet/crossair_w_login" method=post>
<table>
<tr>
  <td align=center class="tabletitle" >
  Login as a registered Swiss online customer
</td></tr>
...
<tr>
  <td >User name</td>
  <td><input type=text name="LOGIN"
   value="" size=10 ></td></tr>
<tr>
   <td>Password</td>
   <td><input type=password
    name="PASSWORD"value=""
    size="10"></td> </tr>
...
</table>
</form>
```

Figure 5. The HTML codes of the form in Figure 4

There are some important differences between the features of search form and non-search form. First, the number of "INPUT", "SELECT", and "TEXTAREA" elements in search forms is larger than that in non-search forms. Second, the value of the "method" attribute in "FORM" elements is always set as "POST" in search forms, while it is always set as "GET" in non-search forms. Moreover, the elements' values of search forms often contain some keywords such as "search", "find", or other words that have the same meaning as "search". These differences, however, are not the only decisive factors. There are other features that can be explored by classification algorithms.

According to the major differences, six kinds of rules are used in the feature extraction process as follows:

1. Extract the "name" attribute values from "input", "select", "textarea", and "label" elements;
2. Extract the "value" attribute value from "input", "textarea", and "label" elements;
3. Extract the "name" and "method" attribute values from "form" elements;
4. Extract the words that appear between slashes(/) in the "action" attributes of the "form" elements;
5. Extract the words that appear between slashes(/) in the "src" and "alt" attributes of the "input-image" element;
6. Calculate the number of "input", "select", "label", and "textarea" elements in each form.

The next step is to standardize the value of the features that are extracted from the forms.

First, all strings are transformed into lowercases; then the string type values are aggregated and mapped to specific enumerating values. For instance, the values of "search", "find", or "srch" are mapped to "search".

2.2 The Sparse Data Matrix

The extracted features and the labels of forms are used to compose the data matrix for the classification algorithm. The formalized data matrix is shown in Table 1.

Table 1. The data matrix for form classification

class	t_1	t_2	$\cdots$	t_i	$\cdots$	t_m
c_1	a_{11}	a_{12}	$\cdots$	a_{1i}	$\cdots$	a_{1m}
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
c_j	a_{j1}	a_{j2}	$\cdots$	a_{ji}	$\cdots$	a_{jm}
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
c_n	a_{n1}	a_{n2}	$\cdots$	a_{ni}	$\cdots$	a_{nm}

The set $T = \{t_1, t_2 \cdots, t_m\}$ in Table 1 represents the names of the form features. For form classification, the label of a form is represented as an element in the set $C = \{yes, no\}$, while "yes" indicates that the form is a search interface of hidden Web and "no" indicates a non-search interface. Each row is a sample form. a_{ji} represents the value of feature t_i in the j th form, and c_j indicates the class of the j th form. Table 2 illustrates two examples of a search form and a non-search form as shown in Figure 2 and Figure 4.

The expression of t_i is a four-tuple of "element name"-"type"-"attribute name"-"sequence number". The "element name" contains six values: "FORM", "SELECT", "INPUT", "TEXT AREA", "LABEL", and their statistics. For element, "INPUT", the value of "type" can be "text", "hidden", and so on. "attribute name" has six options: "name", "value", "src", "alt", "method", and "action". Sequence number represents the sequence of the features in the form.

As illustrated in Table 2, the combination of "element name","type","attribute name", and "sequence number" has many unique alternatives. This will result in a high-dimensional feature space for form classification. Furthermore, since each form has just a few features, the data matrix for classification is very sparse and there are many missing values and noisy features. This problem presents a big challenge for search form detection.

Table 2. Two examples of form vectors

class	form-action-1	form-action-2	form-action-3	$\cdots$	input-text-number	input-submit-number
yes	www.thearda.com	cgi-bin	search	$\cdots$	1	1
no	servlet	login	?	$\cdots$	3	0

3. Feature Weighting Random Forest Algorithm

This section presents an improved random forest algorithm, which extends the classical random forest method with a feature weighting technique. We describe the basic random forest classification approach in Subsection 3.1 and our new algorithm in Subsection 3.2.

3.1 Random Forest Algorithm

Random Forest (*RFA*) (Ho, 1998; Breiman, 2001) is an ensemble of unpruned classification or regression trees, which is induced from bootstrapping samples of the training set, using random feature selection in the tree induction process. Prediction is made by aggregating the predictions of the ensemble. Random Forest grows many classification trees. To classify a new object from an input vector, it passes the sample vector to each of the trees in the forest. Each tree gives a classification decision. All the classification results of individual trees are combined to choose the classification having the most votes over all the classification trees in the forest.

Random forest generally exhibits a substantial performance improvement over single tree classifiers, such as CART (Breiman, Friedman, & Olshen, 1984) and C4.5 (Quinlan, 1993). It presents a good solution for classification of sparse data sets. Since basic *RFA* selects features randomly, it's easy to select unimportant or noisy features, especially when there are many noisy features in the training data. This may lead to bad classification results. As discussed in previous sections, the data matrix for form classification contains many missing values. It's necessary to enhance basic *RFA* so that the performance can be improved in search form classification.

3.2 Improved Random Forest with Weighted Feature Selection

Due to the sparse feature space, there are a lot of missing values in the training data set. The features with too many missing values become less important and can be treated as noisy features. Random selection of features often obtains many unimportant or noisy features, which leads to bad trees in the forest. To avoid this, we extend basic *RFA*, using a weighting scheme in feature selection to replace random selection. We use χ^2 statistic to measure the importance of features (Larson, 1982). The χ^2 statistic of a feature A against the class feature is computed as follows.

$$\chi^2 = \sum_{i=1}^{m} \sum_{j=1}^{2} \frac{(o_{ij} - e_{ij})^2}{e_{ij}} \tag{1}$$

where

- m is the number of values in feature A
- o_{ij} is the count of joint event (A_i, C_j), defined as:

$$o_{ij} = count(A = a_i \cap C = c_j) \tag{2}$$

- e_{ij} is the expected value of joint event (A_i, C_j), defined as:

$$e_{ij} = \frac{count(A = a_i) \times count(C = c_j)}{N} \tag{3}$$

  where N is the number of the samples in the training data, $count(A = a_i)$ is the number of samples whose value of feature A is a_i, and $count(C = c_j)$ is the number of samples whose value of the class feature is c_j.

An χ^2 statistic weight is calculated for each feature. From the weights, we select only different subsets of features with high weights to build individual decision trees.

Given a set of decision trees built from different subsets of features, we use a probability estimation technique to combine the results of individual classifiers. Assume x is a test instance and is given to each classifier h_j $(j = 1...k)$ for deciding a possible class c_i. The output of an individual classifier can be computed as $P(I(x) = c_i \mid h_j)$. The final classification result is achieved by combining the probability values as:

$$P(I(x) = c_i) = \frac{1}{k} \sum_{j=1}^{k} P(I(x) = c_i \mid h_j) \tag{4}$$

If class c_i has the highest probability, c_i is the class of x. Kittler has provided a more profound explanation of this method (Kittler, Hatef, Duin, & Matas, 1998). The pseudo-code of the new algorithm (*IRFA*) is given in *Algorithm* 1.

Step 1 is to compute a weight for each feature according to (1). Step 2 sorts the features in descending order of feature weights. Step 3 selects n' features from the entire feature set according to a given feature selection rate β. Step 4 learns individual classifiers from the selected training samples (and selected features). Selection of training samples employs the bootstrapping method. The method of sampling without replacement is used to select t features from n' features, where $t = \lfloor \log_2 n + 1 \rfloor$. After each iteration, the learned decision tree classifier will be added to forest M^*. After forest M^* is grown, Step 5 classifies the unlabeled instances based on (4).

Algorithm 1 The pseudo-code of the feature weighting random forest algorithm

```
Input:
- $D$ : the training database (its number is $d$ ),
- $N$ : the features of the forms (its number is $n$ ),
- $C$ : the target class attribute $C = yes, no$ ,
- $k$ : the number of decision trees,
- $\beta$ : the selection rate of features.
Output: the decision forest $M^*$ .
Process:
1. Compute the weight $W$ based on Formula (1);
2. Sort $N$ on the descending order of weight $W$ ;
3. Let $n' = \lfloor \beta \cdot n \rfloor$, and select $\lfloor \beta \cdot n \rfloor$ features with larger weights as the training samples;
4. for $i = 1$ to $k$ do
    (a) Select $d'$ samples from the training samples by bootstrapping;
    (b) Randomly select $t$ features; where $t = \lfloor \log_2 n' + 1 \rfloor$ and the selection is biased towards
        the features with larger weights;
    (c) Build a decision tree from the $d'$ samples with selected features;
    (d) Add the learned decision tree to $M^*$ ;
   endfor
5. Using $M^*$ to do classification based on Formula (4).
```

3.3 The computational complexity

The computational complexity of *RFA* (Breiman, 2001) is $O(ktd \log d)$, where k is the number of decision trees, t is the number of attributes, d is the number of training samples. In *IRFA*, the enumerating number of the feature attribute is constant (Formula (1)). The computational complexity of all feature weights is $O(n)$. Using the bucket sorting method, the weights can be sorted in linear time. Therefore, the computational complexity of the *IRFA* is $O(ktd \log d + \alpha n)$, where $t = \lfloor \log_2 n + 1 \rfloor$. Therefore, the computational complexity of *IRFA* is very close to the complexity of *RFA*.

The computational cost depends on three factors: the number of decision trees k, the number of features n, and the number of training samples d. We will discuss how to select the number of decision trees k and the number of features n to balance between classification accuracy and computational cost in Section 4.

4. Experiments

4.1 Data Sets

We used two Web page collections in our experiments. One was taken from project Metaquerier[1], and the other was created by crawling the website Search Engine Guide[2] with a Web crawler implemented in Java. The two collections represent a pseudo-random crawling of the Web. A HTML parser was developed to extract the HTML forms and their context features from these two collections. The extracted forms were used to compose the final data sets for experiments.

Table 3. The three data sets used in the experiments

	search	non-search	features	content of data sets
Data set 1	46	43	198	Extracted from the website collection crawled from Search Engine Guide by crawler
Data Set 2	65	116	208	Extracted from artificial website collection of Metaquerier project
Data Set 3	51	96	202	The forms are selected from data set 1 and data set 2.

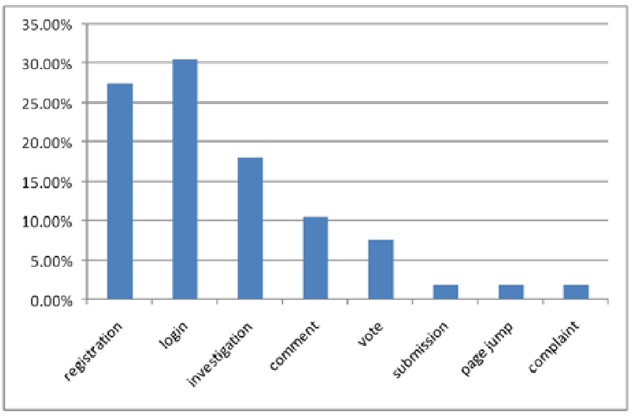


Figure 6. The domain distribution of non-search forms

We manually classified the extracted forms into search forms (*i.e.*, real search interface of hidden Web) and non-search forms. Three classification data sets were constructed from the classified forms, as shown in Table 3. The three data sets have a variety of sample distributions. Data set 1 and Data set 2 cover different domains, while Data set 3 is a mixture

[1]http://metaquerier.cs.uiuc.edu/repository/ training data set

[2]http://www.searchengineguide.com

of the two domains. The three data sets also have different feature types and feature numbers. The average number of features in the data sets is over 200, while the average number of samples is less than 140. The matrices of the three data sets were quite sparse, and the number of features was quite large.

To further test the robustness of our method, the non-search forms in the data sets were made of a variety of forms, including registration forms, login forms, network investigation forms, *etc*. Figure 6 shows the distribution of different non-search forms.

4.2 Comparison Experiments

We first carried out experiments to compare our random forest method with four well-known classification algorithms, *i.e.*, Support Vector Machine (SVM), C4.5, Naïve Bayes,and Random Forest Algorithm (*RFA*) implemented in Weka[3]. We also implemented our algorithm (*IRFA*) as a plug-in of Weka and conducted all experiments in this environment to make a fair comparison. We conducted the standard 10-fold classification experiments on the three data sets. The evaluation metrics used were precision and computation time. The number of trees was set to 100 for *IRFA* and parameter β set to 0.5. The final experimental results are shown in Table 4.

Table 4. The results of comparison experiments

		Bayes	C4.5 Decision Tree	SVM	RFA	*IRFA*
Data Set 1	Precision Time Cost(s)	82.02% <0.005	80.90% 0.05	79.78% 0.52	88.76% 3.16	91.01% 7.08
Data Set 2	Precision Time Cost(s)	83.43% <0.005	88.40% 0.03	82.87% 0.88	91.71% 5.22	92.27% 17.86
Data Set 3	Precision Time Cost(s)	84.35% <0.005	89.79% 0.02	85.71% 0.88	91.84% 3.58	93.88% 15.13

We can see that *IRFA* showed significant improvement over the other four algorithms. The result of C4.5 was better than SVM and Naïve Bayes. This was due to the fact that there were a lot of missing values in the data sets and SVM and Naïve Bayes did not perform well in this kind of sparse data. The high dimensionality in the training sets, however, causes an overfitting problem to C4.5 because the single decision tree could become very complex. *RFA* and *IRFA* can avoid this problem by selecting different subsets of features to build individual decision trees. Compared with *RFA*, *IRFA* uses features that are more correlated to the class label feature, so the accuracy of each individual tree is improved. Therefore, our method got better performance than *RFA*. Since *IRFA* needed to compute the χ^2 values for features, it

[3]http://www.cs.waikato.ac.nz/ml/weka/

took more computational time. This extra overhead, however, is worthwhile and acceptable in real applications because the training process is offline and not executed frequently.

4.3 Selection of the Number of Features

When building each decision tree, *IRFA* only selects a subset of the original features. The number of selected features is controlled by the selection rate β. Different selection rates result in different classification precisions and computational costs. We carried out experiments on the three data sets with $\beta = 0.1, 0.2, \ldots, 1.0$. Figure 7 plots selection rate β against precision, while Figure 8 is β against computational cost.

Figure 7 shows that when $\beta < 0.5$, the precision increases greatly as the selection rate increases. The reason is that a larger selection rate increases the number of features to be selected. When $0.5 \le \beta \le 0.8$, the classification performance becomes relatively stable. This means that the forest has selected enough discriminative features. When $\beta > 0.8$, the precision will decrease as the selection rate increases. This can be explained by the idea that having too large a selection rate will increase the possibility of selecting noisy features. Most experiments have shown that $\beta = 0.5$ was a good setting.

Figure 8 shows that the computational time of *IRFA* increases linearly as the feature selection rate increases. This property indicates that *IRFA* is scalable to large high-dimensional data.

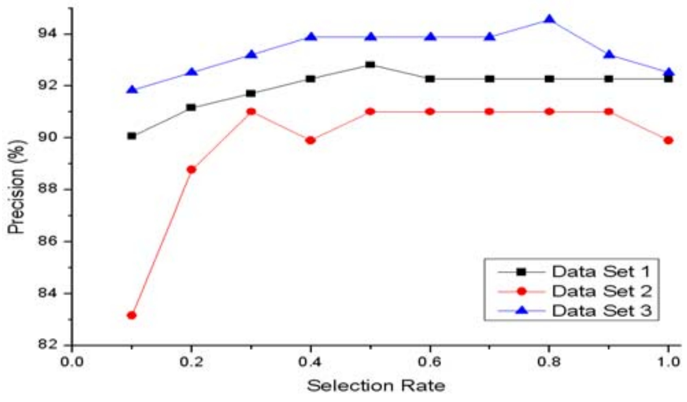


Figure 7. Influence of the number of features on precision

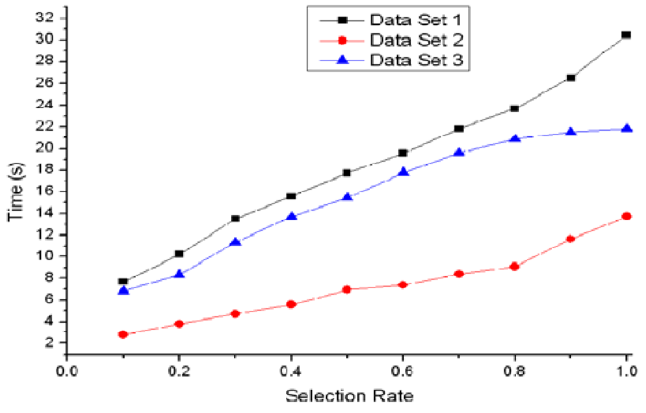


Figure 8. Influence of the number of features on time cost

4.4 Selection of the Number of Trees

Another interesting issue is whether the performance of *IRFA* highly depends on the size of a forest (number of decision trees in the forest). Since a large number of trees lead to a considerable computational cost, we need to find a good tradeoff between classification precision and computational cost. We performed experiments on the three data sets with different tree numbers $(k = 10, 25, 50, 75, 100, 125, 150, 200)$. The feature selection rate β was set to 0.5 in all experiments. The experimental results are shown in Figure 9 and Figure 10.

Figure 9 plots the precision against the number of trees k. The results show that if the number of trees is too small, the classification performance of the forest will be unstable. If the number of trees is too large, however, the computational cost for generating a forest will be very high. The classification precision becomes stable when $75 < k < 150$. The near-optimal precision can be obtained when k is set to 100. Moreover, as shown in Figure 10, with the increase of the number of trees, the computational time increases linearly as well. Therefore, in most situations, $k = 100$ is a good balance between classification accuracy and computational cost.

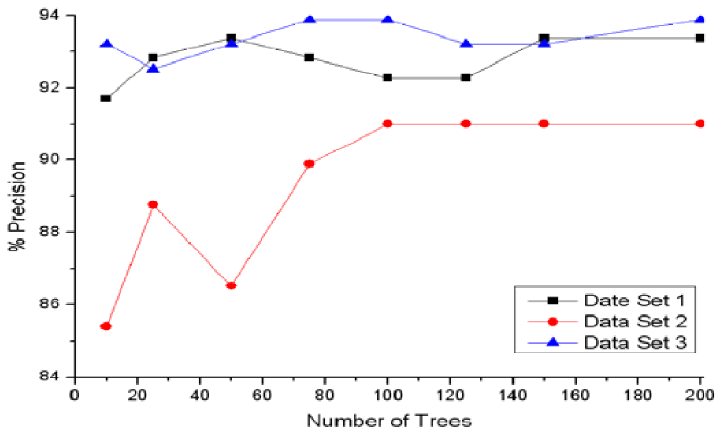


Figure 9. Influence of the number of trees on precision

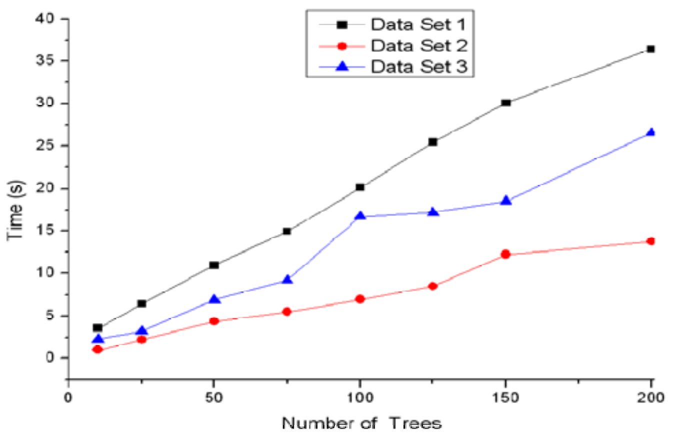


Figure 10. Influence of the number of trees on time cost

4.5 Comparison in Different Domains

The representation of search forms in different domains varies, so it is necessary to investigate if the performance of *IRFA* can be consistent in different domains. An experiment was designed to classify search interfaces from four different domains. We used the Web collection provided by Metaquerier[4], which contains the information about Books, Movies, Airfares, and Jobs. The number of trees for *IRFA* was set to 100, and parameter β was set to 0.5.

The experimental results are shown in Table 5. We can see the obvious variance of accuracy with regard to different domains. The detection accuracies for Books and Jobs domains are higher than those of Movies and Airfares. This was due to the different HTML structures of the search interfaces, as the search interfaces of Movies and Airfares domains are more complex and some of them require more than one step to get the content. The results imply that more formalized and simpler interfaces are more easily recognized. From the results, we also observe that the classification performance of *IRFA* was very stable in various domains. Even though the precision of SVM on the Movies data set was a little better than IRFA, it became the worst in other domains. Compared with other algorithms, *IRFA* was the most stable.

Table 5. Results of comparison experiments on different domains

	Dataset Size	Bayes	C4.5	SVM	RFA	IRFA
Books	251	92.83%	96.41%	89.24%	92.03%	**96.81%**
Movies	126	91.27%	91.27%	**92.06%**	87.30%	91.27%
Airfares	265	89.06%	90.94%	87.92%	91.69%	**91.70%**
Jobs	104	88.46%	94.23%	78.85%	**96.15%**	**96.15%**

4.6 Comparison with Different Feature Selection Schemes

We conducted experiments to demonstrate the performance of *IRFA* with different feature selection schemes. In this section, we used four commonly used feature selection functions mentioned in (Dash & Liu, 1997).

1. Random selection, which randomly selects the features using sampling without replacement, is the simplest feature selection method. Random feature selection is the method used in the original random forest (Breiman, 2001).
2. Information gain is defined as the difference between the original information requirement and the new requirement (Han & Kamber, 2007). The information gain value of a feature X

[4]http://metaquerier.cs.uiuc.edu/repository/training data set

is attained by computing the difference between the prior uncertainty and expected posterior uncertainty using X. Feature X is preferred to feature Y if the information gain from feature X is greater than that from feature Y (Dash & Liu, 1997). In this experiment, the information gain from feature X is set to feature X as its weight instead of Chi-square value in *IRFA*.

3. Gain Ratio is an extension of information gain. It attempts to overcome the problem that the information gain measure is biased toward tests with many outcomes (*i.e.*, it prefers to select attributes having a large number of values) (Han & Kamber, 2007). The process of embedding gain ratio into *IRFA* is similar to information gain in our experiment.
4. Chi-square χ^2 that has been explained in Section 3.2. The experiments in the prior sections were all based on this feature selection method.

Table 6. Comparison with different feature selection schemes

	Random	Information Gain	Gain Ratio	Chi-square χ^2
Data Set 1	88.76%	88.76%	89.89%	**91.01%**
Data Set 2	91.71%	91.16%	92.27%	**92.27%**
Data Set 3	91.84%	93.20%	93.20%	**93.88%**

We implemented the four feature selection methods in Weka's random forest package, and carried out experiments on the data sets described in Section 4.1. The results are shown in Table 6. From the results, we can see that the Chi-square method is better than the others. The reason that Information Gain and Gain Ratio scheme did not attain better performance can be explained as follows: since both of them use the same feature evaluation criterion in feature sampling and tree construction (for node splitting), the information of features cannot be fully exploited.

5. Conclusions

This paper has proposed *IRFA*, an improved random forest algorithm, for detecting search interfaces of the hidden Web. We extend the original random forest algorithm with a weighted feature selection method to automatically select a more representative subset of features for building each decision tree. The new method can overcome the problem of classifying high-dimensional and sparse search interface data through the ensemble of decision trees, each learned from a different subset of the original feature space. We have implemented the new algorithm and compared it with SVM, C4.5, Naïve Bayes, and original random forest algorithm (*RFA*). The experimental results have shown that our method is more accurate and robust. We have also observed that the new method is scalable to high dimensional data.

In the future, we plan to investigate more feature weighting methods for construction of random forests. Currently, we just use the features in the search forms. It is expected that using contextual information near the search forms may improve detection performance.

Acknowledgements

This research is supported in part by NSFC under grant No.60603066 and China National High-tech Program under grants No.2007AA01Z436 and No.2006AA01A124. Part of Joshua Huang’s research was supported by the 863 project matching fund from The University of Hong Kong.

References

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5-32.

Breiman, L., Friedman, J. H., & Olshen, R. A. (1984). *Classification and regression trees*. New York: Chapman & Hall.

BrightPlanet.com (2000). The deep web: Surfacing hidden value. URL http://www.brightplanet.com

Caverlee, J., Liu, L., & Probe, D. B. (2004). Cluster and discover: focused extraction of qa-pagelets from the deep web. *Proceeding of the 20th International Conference of Data Engineering.*

Cope, J., Craswell, N., & Hawking, D. (2003). Automated discovery of search nterfaces on the web. *Fourteenth Australasian Database Conference*. Adelaide, Australia: Fourteenth Australasian Database Conference.

Dash, M., & Liu, H. (1997). Feature selection for classification. *Intelligent Data nalysis*, 1(3), 131-156.

Florescu, D., Levy, A. Y., & Mendelzon, A. O. (1998). Database techniques for the world wide web: A survey. *SIGMOD Record*, 27(3), 59-74.

Han, J., & Kamber, M. (2007). *Data Mining: Concepts and Techniques(2nd version)*. China Machine Press.

He, B., Patel, M., Zhang, Z., & Chang, K. C. (2007). Accessing the deep web. *Communications Of The ACM*, 50(5).

Ho, T. K. (1998). The random subspace method of constructing decision forests. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20(8), 832-844.

Kittler, J., Hatef, M., Duin, R., & Matas, J. (1998). On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20, 226-239.

Lage, J. P., Silva, D., Golgher, P. B., & Laender, A. H. F. (2004). Automatic generation of agents for collecting hidden web pages for data extraction. *Data and Knowledge Engineering*, 49, 177-196.

Larson, H. J. (1982). *Introduction to probability theory and statistical inference*. New York: Wiley, 3 ed.

Ntoulas, A., Zerfos, P., & Cho, J. (2005). Downloading textual hidden web content through keyword queries.

Quinlan, J. R. (1993). *C4.5: Programs for Machine LearningMachine Learning*. Morgan Kaufmann.

Raghavan, S., & Garcia-Molina, H. (2001). Crawling the hidden web. *Proceedings of the 27th International Conference on Very Large Data Bases*. Roma, Italy.

Wang, J., & Lochovsky, F. (2003). Data extraction and label assignment for web databases. *Proceedings of the 12th International World Wide Web Conference.* Budapest, Hungary.

Zhang, Z., He, B., & Chang, K. C. (2004). Understanding web query interfaces: best-effort parsing with hidden syntax. *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*. Paris, France.

Computational Linguistics and Chinese Language Processing
Vol. 13, No. 4, December 2008, pp. 405-420

Corpus Cleanup of Mistaken Agreement Using Word Sense Disambiguation

Liang-Chih Yu[*], **Chung-Hsien Wu**[+], **Jui-Feng Yeh**[#], **and Eduard Hovy**[‡]

Abstract

Word sense annotated corpora are useful resources for many text mining applications. Such corpora are only useful if their annotations are consistent. Most large-scale annotation efforts take special measures to reconcile inter-annotator disagreement. To date, however, nobody has investigated how to automatically determine exemplars in which the annotators agree but are wrong. In this paper, we use OntoNotes, a large-scale corpus of semantic annotations, including word senses, predicate-argument structure, ontology linking, and coreference. To determine the mistaken agreements in word sense annotation, we employ word sense disambiguation (WSD) to select a set of suspicious candidates for human evaluation. Experiments are conducted from three aspects (*precision*, *cost-effectiveness ratio*, and *entropy*) to examine the performance of WSD. The experimental results show that WSD is most effective in identifying erroneous annotations for highly-ambiguous words, while a baseline is better for other cases. The two methods can be combined to improve the cleanup process. This procedure allows us to find approximately 2% of the remaining erroneous agreements in the OntoNotes corpus. A similar procedure can be easily defined to check other annotated corpora.

[*] Department of Information Management, Yuan-Ze University, Chung-Li, Taiwan, R.O.C.
E-mail: lcyu@saturn.yzu.edu.tw
[+] Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan, R.O.C.
E-mail: chwu@csie.ncku.edu.tw
[#] Department of Computer Science and Information Engineering, National Chiayi University, Chiayi, Taiwan, R.O.C.
E-mail: ralph@mail.ncyu.edu.tw
[‡] Information Sciences Institute, University of Southern California, 4676 Admiralty Way, Marina del Rey, CA 90292, U.S.A.
E-mail: hovy@isi.edu

[Received September 11, 2008; Revised February 3, 2009; Accepted February 5, 2009]

Keywords: Corpus Cleanup, Word Sense Disambiguation, Semantic Analysis, Entropy

1. Introduction

Word sense annotated corpora are useful resources for many text mining applications, such as thesaurus construction (Tseng, 2002; Yeh, 2004; 2008), paraphrase extraction (Zhao *et al.*, 2008; Bhaget & Ravichandran, 2008), opinion mining (Ku & Chen, 2007; Kim & Hovy, 2007), and medical information extraction (Wu *et al.*, 2005; Yu *et al.*, 2008). Various machine learning algorithms can then be trained on these corpora to improve the applications' effectiveness. Lately, many such corpora have been developed in different languages, including SemCor (Miller *et al.*, 1993), LDC-DSO (Ng & Lee, 1996), Hinoki (Kasahara *et al.*, 2004), and the sense annotated corpora with the help of Web users (Chklovski & Mihalcea, 2002). The SENSEVAL[1] (Kilgarriff & Palmer, 2000; Kilgarriff, 2001; Mihalcea & Edmonds, 2004) and SemEval-2007[2] evaluations have also created large amounts of sense tagged data for word sense disambiguation (WSD) competitions.

The OntoNotes (Pradhan *et al.*, 2007a; Hovy *et al.*, 2006) project has created a multilingual corpus of large-scale semantic annotations, including word senses, predicate-argument structure, ontology linking, and coreference[3]. In word sense creation, sense creators generate sense definitions by grouping fine-grained sense distinctions obtained from WordNet and dictionaries into more coarse-grained senses. There are two reasons for using this grouping instead of using WordNet senses directly. First, people have trouble distinguishing many of the WordNet-level distinctions in real text and make inconsistent choices; thus, the use of coarse-grained senses can improve inter-annotator agreement (ITA) (Palmer *et al.*, 2004; 2006). Second, improved ITA enables machines to more accurately learn to perform sense tagging automatically. Sense grouping in OntoNotes has been calibrated to ensure that ITA averages at least 90%. Table 1 shows the OntoNotes sense tags and definitions for the word *arm* (noun sense). The OntoNotes sense tags have been used for many applications, including the SemEval-2007 evaluation (Pradhan *et al.*, 2007b), sense merging (Snow *et al.*, 2007), sense pool verification (Yu *et al.*, 2007), and class imbalance problems (Zhu & Hovy, 2007).

[1] http://www.senseval.org

[2] http://nlp.cs.swarthmore.edu/semeval

[3] Year 1 and Year 2 of the OntoNotes corpus has been released by Linguistic Data Consortium (LDC) (http://www.ldc.upenn.edu) in 2007 and 2008, respectively.

Table 1. OntoNotes sense tags and definitions. The WordNet version is 2.1.

Sense Tag	Sense Definition	WordNet sense
arm.01	The forelimb of an animal	WN.1
arm.02	A weapon	WN.2
arm.03	A subdivision or branch of an organization	WN.3
arm.04	A projection, a narrow extension of a structure	WN.4 WN.5

In creating Onto Notes, each word sense annotation involves two annotators and an adjudicator. First, all sentences containing the target word along with its sense distinctions are presented independently to two annotators for sense annotation. If the two annotators agree on the same sense for the target word in a given sentence, then their selection is stored in the corpus. Otherwise, this sentence is double-checked by the adjudicator for the final decision. The major problem of the above annotation scheme is that only the instances where the two annotators disagree are double-checked, while those showing agreement are stored directly without any adjudication. Therefore, if the annotators happen to agree but are both wrong, the corpus becomes polluted by the erroneous annotations. Table 2 shows an actual occurrence of an erroneous instance (sentence) for the target word *management*. In this example sentence, the actual sense of the target word is *management.01*, but both of our annotators made a decision of *management.02*. (Note that there is no difficulty in making this decision; the joint error might have occurred due to annotator fatigue, habituation after a long sequence of *management.02* decisions, *etc*.)

Table 2. Example sentence for the target word management along with its sense definitions.

Example sentence:

The 45-year-old Mr. Kuehn, who has a background in crisis **management**, succeeds Alan D. Rubendall, 45.	
management.01:	*Overseeing or directing. Refers to the act of managing something.* He was given overall management of the program. I'm a specialist in risk management. The economy crashed because of poor management.
management.02:	*The people in charge. The ones actually doing the managing.* Management wants to start downsizing. John was promoted to Management. I spoke to their management, and they're ready to make a deal.

Although most annotations in OntoNotes are correct, there is still a small (but unknown) fraction of erroneous annotations in the corpus. Therefore, a cleanup procedure is necessary to produce a high-quality corpus. It is, however, impractical for human experts to evaluate the whole corpus for cleanup. Given that we are focusing on word senses, this study proposes the use of WSD to facilitate the corpus cleanup process. WSD has shown promising accuracy in recent SENSEVAL and SemEval-2007 evaluations.

The rest of this work is organized as follows. Section 2 describes the corpus cleanup procedure. Section 3 presents the features for WSD. Section 4 summarizes the experimental results. Conclusions are drawn in Section 5.

2. Corpus Cleanup Procedure

Figure 1 shows the cleanup procedure (dashed lines) for the OntoNotes corpus. As mentioned earlier, each word, along with its sentence instances, is annotated by two annotators. The annotated corpus, thus, can be divided into two parts according to the annotation results. The first part includes the annotation with disagreement among the two annotators, which is double-checked by the adjudicator. The final decisions made by the adjudicator are stored into the corpus. Since this part is double-checked by the adjudicator, it will not be evaluated by the cleanup procedure.

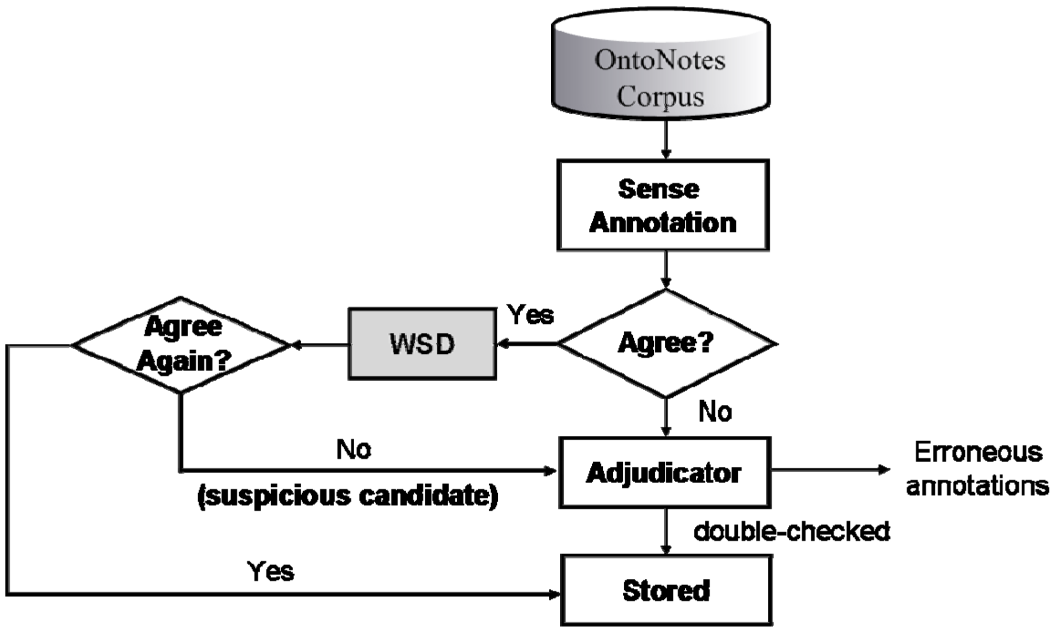


Figure 1. Corpus cleanup procedure.

The second part of the corpus is the focus of the cleanup procedure. The WSD system evaluates each instance in the second part. If the output of the WSD system disagrees with the two annotators, the instance is considered to be a suspicious candidate, otherwise it is considered to be clean and is stored into the corpus. The set of suspicious candidates is collected and subsequently evaluated by the adjudicator to identify erroneous annotations.

3. Word Sense Disambiguation

This study takes a supervised learning approach to build a WSD system from the OntoNotes corpus. The feature set used herein is similar to several state-of-the-art WSD systems (Lee & Ng, 2002; Ando, 2006; Tratz *et al.*, 2007; Cai *et al.*, 2007; Agirre & Lopez de Lacalle, 2007; Specia *et al.*, 2007), which is further integrated into a Naïve Bayes classifier (Lee & Ng, 2002; Mihalcea, 2007). In addition, a new feature, predicate-argument structure, provided by the OntoNotes corpus is integrated as well. The feature set includes:

Part-of-Speech (POS) tags: This feature includes the POS tags in the positions (P_{-3}, P_{-2}, P_{-1}, P_0, P_1, P_2, P_3), relative to the POS tag of the target word. For instance, the POS sequence of the constituent "…mediator in an attempt to break the…" is "NN NN IN DT TO VB DT".

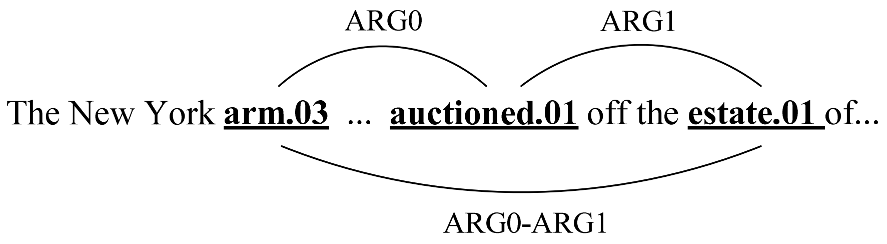
Local Collocations: This feature includes single words and multi-word n-grams. The single words include (W_{-3}, W_{-2}, W_{-1}, W_0, W_1, W_2, W_3), relative to the target word W_0. Similarly, the multi-word n-grams include ($W_{-2,-1}$, $W_{-1,1}$, $W_{1,2}$, $W_{-3,-2,-1}$, $W_{-2,-1,1}$, $W_{-1,1,2}$, $W_{1,2,3}$). For instance, the multi-word n-grams of the above example constituent include {in_an, an_to, to_break, mediator_in_an, in_an_to, an_to_break, to_break_the}.

Bag-of-Words: This feature can be considered as a global feature, consisting of 5 words prior to and after the target word, without regard to position.

Predicate-Argument Structure: The predicate-argument structure captures the semantic relations between the predicates and their arguments within a sentence. Consider the following example sentence.

[$_{\textbf{Arg0}}$ The New York arm of the London-based firm] auctioned off [$_{\textbf{Arg1}}$ the **estate** of John T. Dorrance Jr., the Campbell's Soup Co. heir,] [$_{\textbf{ArgM-TMP}}$ last week].

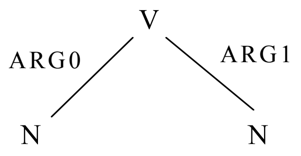
The argument label Arg0 is usually assigned to the *agent*, *causer*, and *experiencer*, while Arg1 is usually assigned to the *patient*. The ArgM-TMP represents a temporal modifier (Babko-Malaya, 2006; Palmer *et al.*, 2005). The predicate-argument structure of the above sentence is illustrated in Figure 2. The semantic relations can be either *direct* or *indirect*. A direct relation is used to model a verb-noun (VN), whereas an indirect relation is used to model a noun-noun (NN) relation. Additionally, an NN-relation can be built from the combination of two VN-relations with the same predicate. Table 3 presents some examples. For instance, NN1 can be built by combining VN1 and VN2. Therefore, the two features, VN1 and NN3, can be used to disambiguate the noun *arm* [4].


ARG0
ARG1
The New York **arm.03** ... **auctioned.01** off the **estate.01** of...
ARG0-ARG1

VN1: (auction.01, ARG0, arm.03)
VN2: (auction.01, ARG1, estate.01)
NN1: (arm.03, ARG0-ARG1, estate.01)

Figure 2. Example of predicate-argument structure.

Table 3. VN and NN-relations. <DATE> is a named entity identified by the IdentiFiner.

Relation Type	Example
VN relation	VN1: (auction.01, Arg0, arm.03) VN2: (auction.01, Arg1, **estate.01**) VN3: (auction.01, ArgM-TMP, <DATE>)
NN relation: 	NN1: (arm.03, Arg0-Arg1, **estate.01**) NN2: (**estate.01**, Arg1-ArgM-TMP, <DATE>) NN3: (arm.03, Arg0-ArgM-TMP, <DATE>)

4. Experimental Results

4.1 Experiment Setup

The experiment data used herein consisted of the 35 nouns from the SemEval-2007 English Lexical Sample Task (Pradhan *et al.*, 2007b). All sentences containing the 35 nouns were selected from the OntoNotes corpus, resulting in a set of 16,329 sentences. This data set was

[4] Our WSD system does not include the sense identifier (except for the target word) for word-level training and testing.

randomly split into training and test sets using different proportions (1:9 to 9:1, 10% increments). The WSD systems (described in Section 3) were then built from the different portions of the training set, called WSD_1 to WSD_9, respectively, and applied to their corresponding test sets. In each test set, the instances with disagreement among the annotators were excluded, since they have already been double-checked by the adjudicator. A baseline system was also implemented using the principle of *most frequent sense* (MFS), where each word sense distribution was retrieved from the OntoNotes corpus. Table 4 shows the accuracy of the baseline and WSD systems.

Table 4. Accuracy of the baseline and WSD systems with different training portions.

	Baseline (MFS)	WSD								
		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Accuracy	0.696	0.751	0.798	0.809	0.819	0.822	0.824	0.831	0.836	0.832

The output of WSD may agree or disagree with the annotators. The instances with disagreement were selected from each WSD system as suspicious candidates. This experiment randomly selected at most 20 suspicious instances for each noun then unified these instances to form a suspicious set of 687 instances. An adjudicator who is a linguistic expert then evaluated the suspicious set, and agreed in 42 instances with the WSD systems, indicating about 6% (42/687) truly erroneous annotations. This corresponds to 2.6% (42/16329) erroneous annotations in the corpus as a whole, which we verified by an independent random spot check.

In the following sections, we examine the performance of WSD from three aspects: precision, cost-effectiveness ratio, and entropy. In addition, we summarize a general cleanup procedure for other sense-annotated corpora.

4.2 Cleanup Precision Analysis

The cleanup precision for a single WSD system can be defined as the number of erroneous instances identified by the WSD system, divided by the number of suspicious candidates selected by the WSD system. An erroneous instance refers to an instance where the annotators agree with each other but disagree with the adjudicator. Table 5 lists the cleanup precision of the baseline and WSD systems. The experimental results show that WSD_7 (trained on 70% training data) identified 17 erroneous instances out of 120 selected suspicious candidates, thus yielding the highest precision of 0.142. Another observation is that the upper bound of WSD_7 was 0.35 (42/120) under the assumption that it identified all erroneous instances. This low precision discourages the use of WSD to automatically correct erroneous annotations.

Table 5. Cleanup precision of the baseline and WSD systems with different training portions.

	Baseline (MFS)	WSD								
		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Prec	0.090 (17/188)	0.113 (20/177)	0.112 (16/143)	0.113 (17/150)	0.124 (16/129)	0.123 (15/122)	0.127 (16/126)	**0.142** (17/120)	0.130 (14/108)	0.125 (14/112)

4.3 Cleanup Cost-Effectiveness Analysis

The cleanup procedure used herein is a semi-automatic process; that is, WSD is applied in the first stage to select suspicious candidates for human evaluation in the later stage. Obviously, we would like to minimize the number of candidates the adjudicator has to examine. Thus, we use the metric cost-effectiveness (CE) ratio, which is defined as *effectiveness* divided by *cost*, to measure the performance of WSD. The *cost* rate is defined as the number of suspicious instances selected by a single WSD system, divided by the total number of suspicious instances in the suspicious set. The *effectiveness* rate is defined as the number of erroneous instances identified by a single WSD system, divided by the total number of erroneous instances in the suspicious set. On the other hand, the *missing* rate can be defined as 1-*effectiveness* rate. In this experiment, the baseline value of the cost-effectiveness ratio is 1, which means that the human expert needs to evaluate all 687 instances in the suspicious set to identify the 42 erroneous instances. Figure 3 illustrates the CE ratio of the WSD systems. The most cost-effective WSD system was WSD_7. The CE ratios of the baseline and WSD_7 are listed in Table 6. The experimental results indicate that 17.5% of all suspicious instances were required to be evaluated to identify about 40% of the erroneous annotations when using WSD_7.

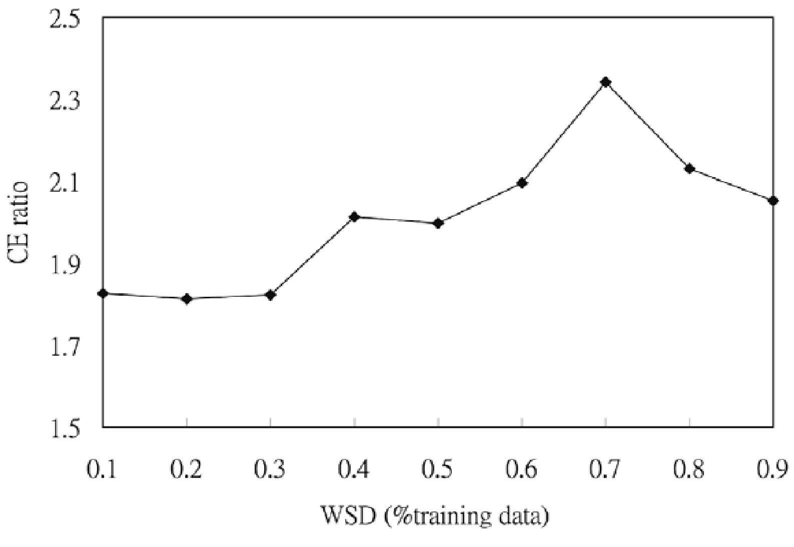


Figure 3. CE ratio of WSD systems with different training portions.

Table 6. CE ratio of the baseline and WSD_7.

	Cost	Effectiveness	CE Ratio
Baseline (MFS)	0.274 (188/687)	0.405 (17/42)	1.48
WSD_7	0.175 (120/687)	0.405 (17/42)	2.31

4.4 Entropy Analysis

So far, the experimental results show that the best WSD system can help human experts identify about 40% erroneous annotations, but it still missed the other 60%. To improve performance, we conducted experiments to analyze the effect of word entropy with respect to WSD performance on identifying erroneous annotations.

For the SemEval 35 nouns used in this experiment, some words are very ambiguous and some words are not. This property of ambiguity may affect the performance of WSD systems in identifying erroneous annotation. To this end, this experiment used *entropy* to measure the ambiguity of words (Melamed, 1997). The entropy of a word can be computed by the word sense distribution, defined as:

$$H(W) = -\sum_{ws_i \in W} P(ws_i)\log_2 P(ws_i), \tag{1}$$

where $H(W)$ denotes the entropy of a word *W*, and $P(ws_i)$ denotes the probability of a word sense. A high entropy value indicates a high ambiguity level. For instance, the noun *defense* has 7 senses (see Table 8) in the OntoNotes corpus, occurring with the distribution {.14, .18, .19, .08, .04, .28, .09}, thus yielding a relative high entropy value (2.599). Conversely, the entropy of the noun *rate* is low (0.388), since it has only two senses with very skewed distribution {.92, .08}.

Consider the two groups of the SemEval nouns: the nouns for which at least one (Group 1) or none (Group 2) of their erroneous instances can be identified by the machine. The use of the criteria "at least one" and "none" is to distinguish whether or not the machine can identify the erroneous instances in these two groups of nouns. The average entropy of these two groups of nouns was computed, as shown in Table 7. An independent *t-test* was then used to determine whether or not the difference of the average entropy among these two groups was statistically significant. The experimental results show that WSD_7 was more effective on identifying erroneous annotations occurring in highly-ambiguous words ($p<0.05$), while the baseline system has no such tendency ($p=0.368$).

Table 7. Average entropy of two groups of nouns for the baseline and WSD_7.

	Group 1	Group 2	Difference	p-value
Baseline (MFS)	1.226	1.040	0.186	0.368
WSD_7	1.401	0.932	0.469*	0.013

*p<0.05

Table 8 shows the detailed analysis of WSD performance on different words. As indicated, WSD_7 identified the erroneous instances (7/7) occurring in the two top-ranked highly-ambiguous nouns, *i.e.*, *defense* and *position*, but missed all those (0/12) occurring in the two most unambiguous words, *i.e.*, *move* and *rate*. The major reason is that the sense distribution of unambiguous words is often skewed, thus, WSD systems built from such imbalanced data tend to suffer from the over-fitting problem; that is, they tend to over-fit the predominant sense class and ignore small sense classes (Zhu & Hovy, 2007). Fortunately, the over-fitting problem can be greatly reduced when the entropy of words exceeds a certain threshold (*e.g.*, the dashed line in Table 8), since the word sense has become more evenly distributed.

Table 8. Entropy of words versus WSD performance. The dashed line denotes a cut-point for the combination of the baseline and WSD_7.

Noun	#sense	Major Sense	Entropy	#err. instances	WSD_7	MFS	WSD_7+ MFS
defense	7	0.28	2.599	5	5	4	5
position	7	0.30	2.264	2	2	2	2
base	6	0.35	2.023	1	1	0	1
system	6	0.54	1.525	2	1	0	1
chance	4	0.49	1.361	1	1	1	1
order	8	0.72	1.348	4	1	0	1
part	5	0.70	1.288	1	1	1	1
power	3	0.51	1.233	3	1	3	3
area	3	0.72	1.008	2	1	2	2
management	2	0.62	0.959	2	1	0	0
condition	3	0.71	0.906	1	0	1	1
job	3	0.78	0.888	1	0	0	0
state	4	0.83	0.822	1	0	0	0
hour	4	0.85	0.652	1	1	1	1
value	3	0.90	0.571	2	1	1	1
plant	3	0.88	0.556	1	0	0	0
move	4	0.93	0.447	6	0	0	0
rate	2	0.92	0.388	6	0	1	1
Total	—	—	—	42	17	17	21

Nouns without erroneous instances: *authority, bill, capital, carrier, development, drug, effect, exchange, future, network, people, point, policy, president, share, source, space*

4.5 Combination of WSD and MFS

Another observation from Table 8 is that WSD_7 identified more erroneous instances when the word entropy exceeded the cut-point, since the over-fitting problem was reduced. Conversely, MFS identified more instances when the word entropy was below the cut-point. This finding encourages the use of a combination of WSD_7 and MFS for corpus cleanup; that is, different strategies can be used with different entropy intervals. For this experimental data, MFS and WSD_7 can be applied below and above the cut-point, respectively, to select the suspicious instances for human evaluation. Therefore, the final suspicious set can be generated by combining the suspicious instances suggested by MFS and WSD_7. As illustrated in Figure 4, when the entropy of words increased, the accumulated effectiveness rates of both WSD_7 and MFS increased accordingly, since more erroneous instances were identified. Additionally, the difference of the accumulated effect rate of MFS and WSD_7 increased gradually from the beginning until the cut-point, since MFS identified more erroneous instances than WSD_7 did in this stage. When the entropy exceeded the cut-point, WSD_7 was more effective and, thus, its effectiveness rate kept increasing, while that of MFS increased slowly, thus, their difference was decreased with the rise of the entropy. For the combination of MFS and WSD_7, its effectiveness rate before the cut-point was the same as that of MFS, since MFS was used in this stage to select the suspicious set. When WSD was used after the cut-point, the effectiveness rate of the combination system increased continuously, and finally reached 0.5 (21/42).

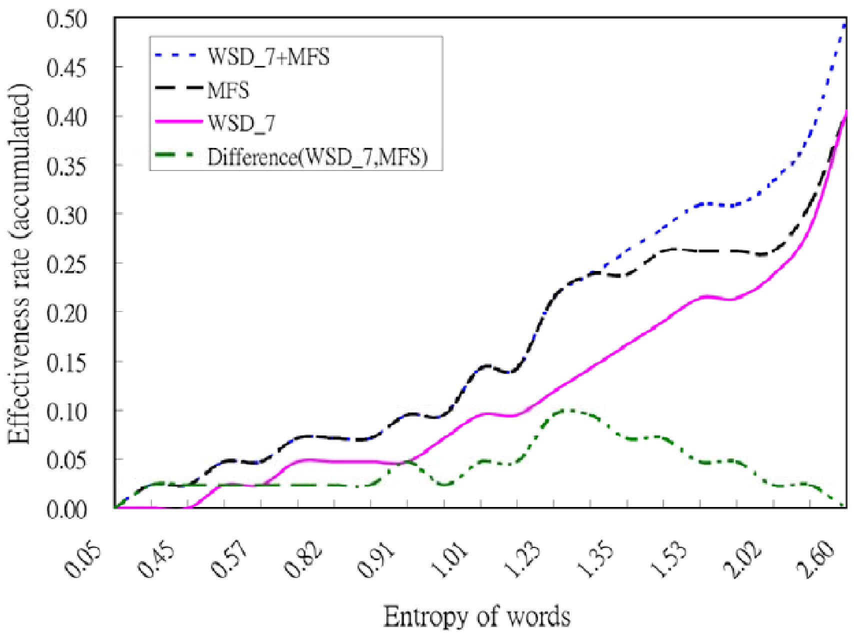


Figure 4. Effectiveness rate against word entropy.

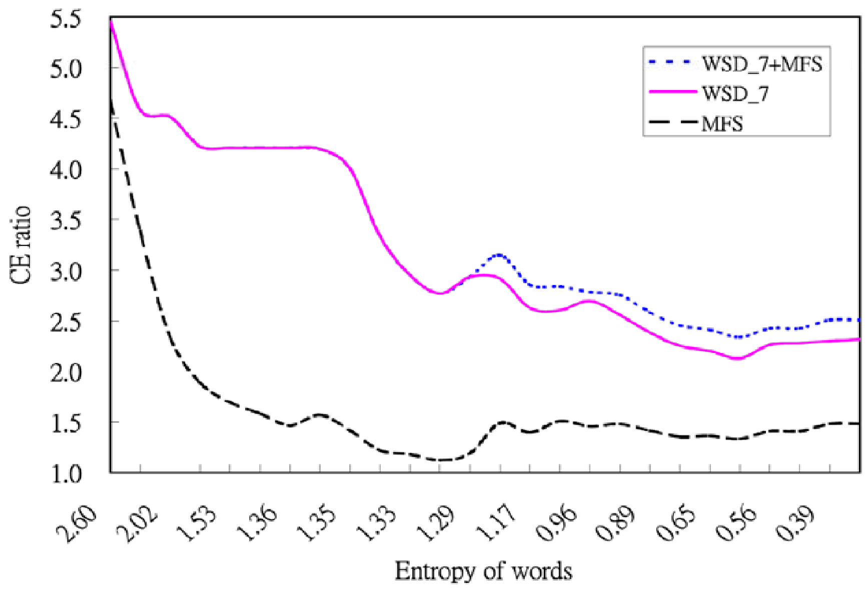


Figure 5. CE ratio against word entropy.

Based on the above experimental results, the most cost-effective method for corpus cleanup is to use the combination method and begin with the most ambiguous words, since the WSD system in the combination method is more effective in identifying erroneous instances occurring in highly-ambiguous words and these words are also more important for many applications. Figure 5 shows the curve of the CE ratios of the combination method by starting with the most ambiguous word. The results indicate that the CE ratio of the combination method decreased gradually after more words with lower entropy were involved in the cleanup procedure. Additionally, the CE ratio of the combination method was improved by using MFS after the cut-point and finally reached 2.50, indicating that 50% (21/42) erroneous instances can be identified by double-checking 20% (137/687) of the suspicious set. This CE ratio was better than 2.31 and 1.48, reached by WSD_7 and MFS, respectively.

The proposed cleanup procedure can be applied to other sense annotated corpora by the following steps:

- Build the baseline (MFS) and WSD systems from the corpus.
- Create a suspicious set from the WSD systems.
- Calculate the entropy for each word in terms of it sense distribution in the corpus.
- Choose a cut-point value. Select a small portion of words with entropy within a certain interval (*e.g.*, 1.0 ~ 1.5 in Table 8) for human evaluation to decide an appropriate cut-point value. The cut-point value should not be too low or too high, since WSD systems may suffer from the over-fitting problem if the value is too low, and the performance would be dominated by the baseline system if the value is too high.

- Combine the baseline and best single WSD system through the cut-point.
- Start the cleanup procedure in the descending order of word entropy until the CE ratio is below a predefined threshold.

5. Conclusion

This study has presented a cleanup procedure to identify incorrect sense annotation in a corpus. The cleanup procedure incorporates WSD systems to select a set of suspicious instances for human evaluation. The experiments are conducted from three aspects: precision, cost-effectiveness ratio, and entropy, to examine the performance of WSD. The experimental results show that the WSD systems are more effective on highly-ambiguous words. Additionally, the most cost-effective cleanup strategy is to use the combination method and begin with the most ambiguous words. The incorrect sense annotations found in this study can be used for SemEval-2007 to improve the accuracy of WSD evaluation.

The absence of related work on (semi-) automatically determining cases of erroneous agreement among annotators in a corpus is rather surprising. Variants of the method described here, replacing WSD for whatever procedure is appropriate for the phenomenon annotated in the corpus (sentiment recognition for a sentiment corpus, *etc*.), are easy to implement and may produce useful results for corpora in current use. Future work will focus on devising an algorithm to perform the cleanup procedure iteratively on the whole corpus.

References

Agirre, E., & Lopez de Lacalle, O. (2007). UBC-ALM: Combining k-NN with SVD for WSD. In *Proc. of the 4th International Workshop on Semantic Evaluations (SemEval-2007) at ACL-07*, 342-345.

Ando, R.K. (2006). Applying Alternating Structure Optimization to Word Sense Disambiguation. In *Proc. of CoNLL*, 77-84.

Babko-Malaya, O. (2006). PropBank Annotation Guidelines.

Bhagat, R., & Ravichandran, D. (2008). Large Scale Acquisition of Paraphrases for Learning Surface Patterns. In *Proc. of the 46th Annual Meeting of the Association of Computational Linguistics (ACL-08)*, 574-682.

Cai, J.F., Lee, W.S., & The, Y.W. (2007). Improving Word Sense Disambiguation Using Topic Features. In *Proc. of EMNLP-CoNLL*, 1015-1023.

Chklovski, T., & Mihalcea, R. (2002). Building a Sense Tagged Corpus with Open Mind Word Expert. In *Proc. of the Workshop on Word Sense Disambiguation: Recent Successes and Future Directions at ACL-02*, 116-122.

Fellbaum, C. (1998). *WordNet: An Electronic Lexical Database*. Cambridge, MA: MIT Press.

Hovy, E.H., Marcus, M., Palmer, M., Ramshaw, L., & Weischedel, R. (2006). OntoNotes: The 90% Solution. In *Proc. of HLT/NAACL-06*, 57-60.

Kasahara, K., Sato, H., Bond, F., Tanaka, T., Fujita, S., Kanasugi, T., *et al.* (2004). Construction of a apanese Semantic Lexicon: Lexeed. In *IPSG SIG: 2004-NLC-159*, Tokyo, 75-82.

Kilgarriff, A. (2001). English Lexical Sample Task Description. In *Proc. of the SENSEVAL-2 Workshop*, 17-20.

Kilgarriff, A., & Palmer, M. editors. (2000). SENSEVAL: Evaluating Word Sense Disambiguation Programs, *Computer and the Humanities*, 34(1-2),1-13.

Kim, S.M., & Hovy, E.H. (2007). CRYSTAL: Analyzing Predictive Opinions on the Web. In *Proc. of EMNLP-CoNLL*, Prague, Czech Republic.

Ku, L.W., & Chen, H.H. (2007). Mining Opinions from the Web: Beyond Relevance Retrieval. *Journal of American Society for Information Science and Technology*, 58(12), 1838-1850.

Lee, Y.K., & Ng, H.T. (2002). An Empirical Evaluation of Knowledge Sources and Learning Algorithms for Word Sense Disambiguation. In *Proc. of EMNLP*, 41-48.

Melamed, I.D. (1997). Measuring Semantic Entropy. In *Proc. of ACL-SIGLEX Workshop*, 41-46.

Mihalcea, R. (2007). Using Wikipedia for AutomaticWord Sense Disambiguation. In *Proc. of NAACL/HLT-07*, 196-203.

Mihalcea, R., & Edmonds, P. editors. (2004). In *Proc. of SENSEVAL-3*.

Miller, G., Leacock, C., Tengi, R., & Bunker, R. (1993). A Semantic Concordance. In *Proc. of the 3rd DARPA Workshop on Human Language Technology*, 303-308.

Ng, H.T., & Lee, H.B. (1996). Integrating Multiple Knowledge Sources to Disambiguate Word Sense: An Exemplar-based Approach. In *Proc. of the 34th Meeting of the Association for Computational Linguistics (ACL-96)*, 40-47.

Palmer, M., Babko-Malaya, O., & Dang, H.T. (2004). Different Sense Granularities for Different Applications. In *Proc. of the 2nd International Workshop on Scalable Natural Language Understanding at HLT/NAACL-04*.

Palmer, M., Dang, H.T., & Fellbaum, C. (2006). Making Fine-grained and Coarse-grained Sense Distinctions, Both Manually and Automatically. *Journal of Natural Language Engineering*, 13, 137-163.

Palmer, M., Gildea, D., & Kingsbury, P. (2005). The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1), 71-106.

Pradhan, S., Hovy, E.H., Marcus, M., Palmer, M., Ramshaw, L., & Weischedel, R. (2007a). OntoNotes: A Unified Relational Semantic Representation. In *Proc. of the First IEEE International Conference on Semantic Computing (ICSC-07)*, 517-524.

Pradhan, S., Loper, E., Dligach, D., & Palmer, M. (2007b). SemEval-2007 Task 17: English Lexical Sample, SRL and All Words. In *Proc. of the 4th International Workshop on*

Semantic Evaluations (SemEval-2007) at ACL-07, 87-92.

Tseng, Y.S. (2002). Automatic Thesaurus Generation for Chinese Documents. *Journal of the American Society for Information Science and Technology*, 53(13), 1130-1138.

Snow, R., Prakash, S., Jurafsky, D., & Ng, A.Y. (2007). Learning to Merge Word Senses. In *Proc. of EMNLP-CoNLL*, 1005-1014.

Specia, L., Stevenson, M., &. das Gracas V. Nunes, M. (2007). Learning Expressive Models for Word Sense Disambiguation. In *Proc. of the 45th Annual Meeting of the Association of Computational Linguistics (ACL-07)*, 41-48.

Tratz, S., Sanfilippo, A., Gregory, M., Chappell, A., Posse, C., & Whitney, P. (2007). PNNL: A Supervised Maximum Entropy Approach to Word Sense Disambiguation. In *Proc. of the 4th International Workshop on Semantic Evaluations (SemEval-2007) at ACL-07*, 264-267.

Wu, C.H., Yu, L.C., & Jang, F.L. (2005). Using Semantic Dependencies to Mine Depressive Symptoms from Consultation Records. *IEEE Intelligent Systems*, 20(6), 50-58.

Yeh, J.F., Wu, C.H., Chen, M.J., & Yu, L.C. (2004). Automated Alignment and Extraction of Bilingual Domain Ontology for Cross-Language Domain-Specific Applications. *In Proc. of the 20th COLING*, 1140-1146.

Yeh, J.F., Wu, C.H., & Chen, M.J. (2008). Ontology-based Speech Act Identification in a Bilingual Dialog System Using Partial Pattern Trees. *Journal of the American Society for Information Science and Technology*, 59(5), 684-694.

Yu, L.C., Wu, C.H., Philpot, A., & Hovy, E.H. (2007). OntoNotes: Sense Pool Verification Using Google N-gram and Statistical Tests. In *Proc. of the OntoLex Workshop at the 6th International Semantic Web Conference (ISWC 2007).*

Yu, L.C., Wu, C.H., Yeh, J.F., & Jang, F.L. (2008). HAL-based Evolutionary Inference for Pattern Induction from Psychiatry Web Resources. *IEEE Trans. Evolutionary Computation*, 12(2), 160-170.

Zhao, S., Wang, H., Liu, T., & Li, S. (2008). Pivot Approach for Extracting Paraphrase Patterns from Bilingual Corpora. In *Proc. of the 46th Annual Meeting of the Association of Computational Linguistics (ACL-08)*, 780-788.

Zhu, J., & Hovy, E.H. (2007). Active Learning for Word Sense Disambiguation with Methods for Addressing the Class Imbalance Problem. In *Proc. of EMNLP-CoNLL*, 783-790.

Computational Linguistics and Chinese Language Processing
Vol. 13, No. 4, December 2008, pp. 421-442

Hierarchical Taxonomy Integration Using Semantic Feature Expansion on Category-Specific Terms

Cheng-Zen Yang*, **Ing-Xiang Chen***, **Cheng-Tse Hung***, **and**

Ping-Jung Wu*

Abstract

In recent years, the hierarchical taxonomy integration problem has obtained considerable attention in many research studies. Many types of implicit information embedded in the source taxonomy are explored to improve the integration performance. The semantic information embedded in the source taxonomy, however, has not been discussed in previous research. In this paper, an enhanced integration approach called SFE (Semantic Feature Expansion) is proposed to exploit the semantic information of the category-specific terms. From our experiments on two hierarchical Web taxonomies, the results show that the integration performance can be further improved with the SFE scheme.

Keywords: Hierarchical Taxonomy Integration, Semantic Feature Expansion, Category-Specific Terms, Hierarchical Thesauri Information

1. Introduction

In many daily information processing tasks, merging two classified information sources to create a larger taxonomy with abundant information is in great demand. For example, an e-commerce service provider may merge various catalogs from other vendors into its local catalog to provide customers with versatile contents. A Web user may also want to integrate different blog catalogs from Web 2.0 portals to organize a personal information management library. In these examples, people may need an efficient automatic integration approach to process the huge amount of information.

In recent years, the taxonomy integration problem has obtained much attention in many research studies (*e.g.* Agrawal & Srikan, 2001; Sarawagi, Chakrabarti, & Godbole, 2003;

* Dept. of Computer Sci. and Eng., Yuan Ze University, 135 Yuan-Tung Rd., Chungli, 320, Taiwan.
Tel.: +886-3-4638800 ext: 2361 Fax: +886-3-4638850.
E-mail: {czyang,sean,chris,pjwu}@syslab.cse.yzu.edu.tw

[Received September 12, 2008; Revised December 11, 2008; Accepted February 19, 2009]

Zhang & Lee, 2004a; Zhang & Lee, 2004b; Zhu, Yang, & Lam, 2004; Chen, Ho, & Yang, 2005; Wu, Tsai, & Hsu, 2005; Ho, Chen, & Yang, 2006; Chen, Ho, & Yang, 2007; Cheng & Wei, 2008; Wu, Tsai, Lee, & Hsu, 2008). As pointed out in these studies, the integration work is more subtle than traditional classification work because the integration accuracy can be further improved with different kinds of implicit information embedded in the source or destination taxonomy. A taxonomy, or catalog, usually contains a set of objects divided into several categories according to some classified characteristics. In the taxonomy integration problem, the objects in a taxonomy, the *source* taxonomy *S*, are integrated into another taxonomy, the *destination* taxonomy *D*. As shown in earlier research, this problem is more than a traditional document classification problem because different kinds of implicit information in the source taxonomy are explored to greatly help integrate source documents into the destination taxonomy. For example, a Naive Bayes classification approach (Agrawal & Srikan, 2001) with the classification relationship information implicitly existing in the source catalog can achieve integration accuracy improvement. Several SVM (Support Vector Machines) approaches (Chen, Ho, & Yang, 2005) can also have similar improvement with other implicit source information.

The implicit source information studied in previous enhanced approaches generally includes the following features: (1) co-occurrence relationships of source objects (Agrawal & Srikan, 2001; Zhu, Yang, & Lam, 2004; Chen, Ho, & Yang, 2005), (2) latent source-destination mappings (Sarawagi, Chakrabarti, & Godbole, 2003; Zhang & Lee, 2004b; Cheng & Wei, 2008), (3) inter-category centroid information (Zhang & Lee, 2004a), and (4) parent-children relationships in the source hierarchy (Wu, Tsai, & Hsu, 2005; Ho, Chen, & Yang, 2006; Wu, Tsai, Lee, & Hsu, 2008). In our survey, however, the semantic information embedded in the source taxonomy has not been discussed. Since different applications have shown that the semantic information can benefit the task performance (Krikos, Stamou, Kokosis, Ntoulas, & Christodoulakis, 2005; Hsu, Tsai, & Chen, 2006), such information should be able to achieve similar improvements for taxonomy integration. In addition, we further study the hierarchical taxonomy integration problem because many taxonomies, such as Web catalogs, existing in the real world are hierarchical.

In this paper, we propose an enhanced integration approach by exploiting the implicit semantic information in the source taxonomy with a semantic feature expansion (SFE) mechanism. The basic idea behind SFE is that some semantically related terms can be found to represent a source category, and these representative terms can be further viewed as the additional common category labels for all documents in the category. Augmented with these additional semantic category labels, the source documents should be more precisely integrated into the correct destination category. The semantic expanding scheme, however, needs to consider the polysemy situation to avoid introducing many topic-irrelevant features. Therefore, SFE employs an efficient correlation coefficient method to select representative semantically-related terms.

To study the effectiveness of SFE, we implemented it based on a hierarchical taxonomy integration approach (EHCI) proposed in Ho *et al.* (2006) and Chen *et al.* (2007) with the Maximum Entropy (ME) model classifiers. We have conducted experiments with real-world Web catalogs from Yahoo! and Google, and measured the integration performance with precision, recall, and F_1 measures. The results show that the SFE mechanism consistently can improve the integration performance of the EHCI approach.

The rest of the paper is organized as follows. Section 2 describes the problem definition and Section 3 reviews previous related research. Section 4 elaborates the proposed semantic feature expansion approach and the hierarchical integration process. Section 5 presents the experimental results, and discusses the factors that influence the experiments. Section 6 concludes the paper and discusses some future directions of our work.

2. Problem Statement

Following the definitions in Ho *et al.* (2006), we assume that two *homogeneous* hierarchical taxonomies, the source taxonomy *S* and the destination taxonomy *D*, participate in the integration process. The taxonomies are said to be *homogeneous* if the topics of the two taxonomies are similar. In addition, the taxonomies under consideration are required to overlap with a significant number of common documents. For example, in our experimental data sets, 20.6% of the total documents (436/2117) in the Autos directory of Yahoo! also appear in the corresponding Google directory.

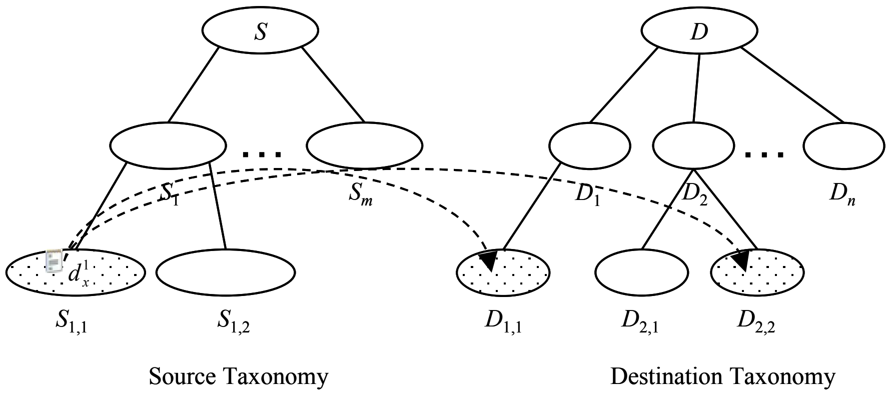


Figure 1. A typical integration scenario for two hierarchical taxonomies.

The source taxonomy *S* has a set of *m* categories, or directories, S_1, S_2, …, S_m. These categories may have subcategories, such as $S_{1,1}$ and $S_{2,1}$. Similarly, the destination catalog *D* has a set of *n* categories. The integration process is to directly decide the destination category in D for each document d_x in S. In this study, we allow that d_x can be integrated into multiple destination categories because a document commonly appears in several different directories in a real-world taxonomy.

Figure 1 depicts a typical scenario of the integration process on two hierarchical taxonomies. For illustration, we assume that the source category $S_{1,1}$ has a significant number of overlapped documents with the destination categories $D_{1,1}$ and $D_{2,2}$. This means that the documents appearing in $S_{1,1}$ should have similar descriptive information as the documents in $D_{1,1}$ and $D_{2,2}$. Therefore, a non-overlapped document d_x^1 in category $S_{1,1}$ should be intensively integrated into both two destination categories $D_{1,1}$ and $D_{2,2}$.

3. Previous Work

3.1 Integration Techniques

In previous studies, different sorts of implicit information embedded in the source taxonomy are explored to help the integration process. These implicit source features can be mainly categorized into four types: (1) co-occurrence relationships of source objects, (2) latent source-destination mappings, (3) inter-category centroid information, and (4) parent-children relationships in the source hierarchy. The co-occurrence relationships of source objects are first studied to enhance a Naive Bayes classifier based on the concept that if two documents are in the same source category, they are more likely to be in the same destination category (Agrawal & Srikan, 2001). The enhanced Naïve Bayes classifier (ENB) is shown to have more than 14% accuracy improvement on average. The work in Chen *et al.* (2005) also has the similar concept in its iterative pseudo relevance feedback approach. As reported in Chen *et al.* (2005), the enhanced SVM classifiers consistently achieve improvement.

Latent source-destination mappings are explored in Sarawagi *et al.* (2003) and Zhang and Lee (2004b). The cross-training (CT) approach (Sarawagi, Chakrabarti, & Godbole, 2003) extracts the mappings from the first semi-supervised classification phase using the source documents as the training sets. Then, the destination documents are augmented with the latent mappings for the second semi-supervised classification phase to complete the integration. The co-bootstrapping (CB) approach (Zhang & Lee, 2004b) exploits the predicted source-destination mappings to repeatedly refine the classifiers. The experimental results show that both CT and CB outperform ENB (Sarawagi, Chakrabarti, & Godbole, 2003; Zhang & Lee, 2004b).

In Zhang and Lee (2004a), a cluster shrinkage (CS) approach, in which the feature weights of all objects in a document category are shrunk toward the category centroid, is proposed. Therefore, the cluster-binding relationships among all documents of a category are strengthened. The experimental results show that the CS-enhanced Transductive SVMs give significant improvement to the original T-SVMs and consistently outperform ENB.

In Wu *et al.* (2005) and Ho *et al.* (2006), the parent-children information embedded in hierarchical taxonomies is intentionally extracted. Based on the hierarchical characteristics, Wu *et al*. extend the CS and CB approach to improve the integration performance. In Ho *et al.* (2006), an enhanced approach called EHCI is proposed to further extract the hierarchical relationships as a conceptual thesaurus. Their results show that the implicit hierarchical information can be effectively used to boost the accuracy performance.

The semantic information embedded in the source taxonomy has not been discussed in past studies. This observation motivates us to study the embedded taxonomical semantic information and its effectiveness.

3.2 Overview of the Maximum Entropy Model Classifiers

In our proposed SFE scheme, we use the Maximum Entropy (ME) model classifiers to perform the main integration task. Here, we provide a brief overview of the ME model as the background of our work. More details can be found in Berger *et al.* (1996). In ME, the entropy $H(p)$ for a conditional distribution $p(y|x)$ is used to measure the uniformity of $p(y|x)$, where y is an instance of all outcomes Y in a random process and x denotes a contextual environment of the contextual space X, or the history space. To express the relationship between x and y, we can have an indicator function $f(x,y)$ (usually known as feature function) defined as:

$$f(x,y)=\begin{cases}1 & \text{if } (x,y) \text{ has the defined relationship}\\ 0 & \text{else}\end{cases} \tag{1}$$

The entropy $H(p)$ is defined by:

$$H(p)=-\sum_{x\in X} p(y|x)\log p(y|x) \tag{2}$$

The Maximum Entropy Principle is to find a probability model $p^{*}\in C$ such that:

$$p^{*}=\arg\max_{p\in C} H(p) \tag{3}$$

where C is a set of allowed conditional probabilities. There are, however, two constraints:

$$E_{\tilde{p}}\{f\}=E_{p}\{f\} \tag{4}$$

and

$$\sum_{y\in Y} p(y|x)=1 \tag{5}$$

where $E_{\tilde{p}}\{f\}$ is the expected value of f with the empirical distribution $\tilde{p}(x,y)$ as defined in Equation 6 and $E_{p}\{f\}$ is the observed expectation of f with the observed distribution $\tilde{p}(x)$ from the training data as defined in Equation 7.

$$E_{\tilde{p}}\{f\} \equiv \sum_{x,y} \tilde{p}(x,y) f(x,y) \tag{6}$$

$$E_{p}\{f\} \equiv \sum_{x,y} \tilde{p}(x) p(y \mid x) f(x,y) \tag{7}$$

As indicated in [10], the conditional probability $p(y \mid x)$ can be computed by:

$$p(y \mid x) = \frac{1}{z(x)} \exp\left(\sum_{i} \lambda_i f_i(x,y) \right) \tag{8}$$

where λ_i is the Lagrange multiplier for feature f_i, and $z(x)$ is defined as

$$z(x) = \sum_{y} \exp\left(\sum_{i} \lambda_i f_i(x,y) \right) \tag{9}$$

With the improved iterative scaling (IIS) algorithm (Darroch & Ratcliff, 1972; Berger, Pietra, & Pietra, 1996), the λ_i values can be estimated. Then, the classifiers are built according to the ME model and the training data.

3.3 Hierarchical Taxonomy Integration

Previous integration research for hierarchal taxonomy integration mainly can be classified into two categories: clustering-based (Cheng & Wei, 2008) and classification-based (Ho, Chen, & Yang, 2006; Zhu, Yang, & Lam, 2004; Chen, Ho, & Yang, 2007). The clustering-based approach has the advantage in handling manifold taxonomies which may even have small overlaps and in performing integration without *a priori* training work. Therefore, the application of the clustering-based approach is much more general. The effectiveness of the clustering-based approach, however, depends on the clustering parameters. For inexperienced users, finding optimal clustering parameters will be very challenging.

Although the classification-based approach is more appropriate for handling taxonomies which have significant overlaps, it cannot handle the subtle relationships embedded in categories. For example, CatRelate uses five types of hierarchical relationships in a taxonomy to help catalog integration (Zhu, Yang, & Lam, 2004), and an integration scheme called EHCI uses a hierarchical weighting mechanism to strengthen the integration effectiveness (Ho, Chen, & Yang, 2006; Chen, Ho, & Yang, 2007). Nonetheless, CatRelate only discusses the hierarchical relationships on a category basis with a set of simple rules. It may suffer from complicated hierarchical relationships when handling large taxonomies. In contrast, EHCI's hierarchical weighting mechanism considers the influences of category labels of more comprehensive neighboring levels on a document basis. The experimental results reported in Ho *et al.* (2006) and Chen *et al.* (2007) also show that EHCI is effective for handling large taxonomies. Therefore, we use EHCI as our baseline to study the effectiveness of the proposed SFE approach. The following gives a brief overview for EHCI.

Table 1. The label weights assigned for different levels.

Hierarchical Level	Label Weight
Document Level (L_0)	$1/2^0$
One Level Upper (L_1)	$1/2^1$
Two Levels Upper (L_2)	$1/2^2$
…	…
n Levels Upper (L_n)	$1/2^n$

In EHCI, the conceptual relationships (category labels) are first extracted from the hierarchical taxonomy structure as a thesaurus (Ho, Chen, & Yang, 2006; Chen, Ho, & Yang, 2007). Then, the features of each document are extended with the thesaurus by adding the weighted label features. A weighting formula is designed to control the impact of the semantic concepts of each hierarchical level. Equation 10 calculates the EHCI feature weight $f_{x,d}^{e}$ of each term *x* in document *d*, where L_i is the relevant label weight assigned as $1/2^i$ with an *i*-level depth, $f_{x,d}$ is the original weight, and λ is used to control the magnitude relation. The weight $f_{x,d}$ is assigned by $TF_x/\sum TF_i$, where TF_x is the term frequency of *x*, and *i* denotes the number of the stemmed terms in each document. The label weight L_i of each thesaurus is exponentially decreased and accumulated based on the increased levels.

$$f_{x,d}^{e} = \lambda \times \frac{L_x}{\sum_{i=0}^{n} L_i} + (1-\lambda) \times f_{x,d} \tag{10}$$

Table 1 shows the label weights of different levels, where L_0 is the document level, L_1 is one level upper, and so on to L_n for *n* levels upper. The label weighting scheme uses a power-law distribution to avoid over-emphasis on the least related hierarchical levels. To build the enhanced classifiers for destination categories, the same enhancement on hierarchical label information is also applied to the destination taxonomy to strengthen the discriminative power of the classifiers.

Although the EHCI approach employs only the embedded hierarchical information with a simple power-law distribution, the integration accuracy performance can be effectively improved. As reported in Chen *et al.* (2007), the EHCI approach outperforms a straightforward classification scheme that does not employ any embedded information to help hierarchical taxonomy integration.

4. Hierarchical Taxonomy Integration with Semantic Feature Expansion

The proposed semantic feature expansion (SFE) approach is to use extracted representative terms of a category as the implicit semantic information to help the corresponding integration process. In the following, the overall processing flow of SFE is presented first. Related approaches incorporated in the integration process are then described. Finally, the SFE approach is elaborated.

4.1 Integration Process

To apply SFE to hierarchical taxonomies, a hierarchical taxonomy integration approach (EHCI) (Ho, Chen, & Yang, 2006; Chen, Ho, & Yang, 2007) is considered as the baseline. Currently, classifiers based on the Maximum Entropy (ME) model are used because of its prominent performance in many tasks, such as natural language processing (Berger, Pietra, & Pietra, 1996) and flattened taxonomy integration (Wu, Tsai, & Hsu, 2005). Figure 2 shows the entire integration process flow of the SFE approach.

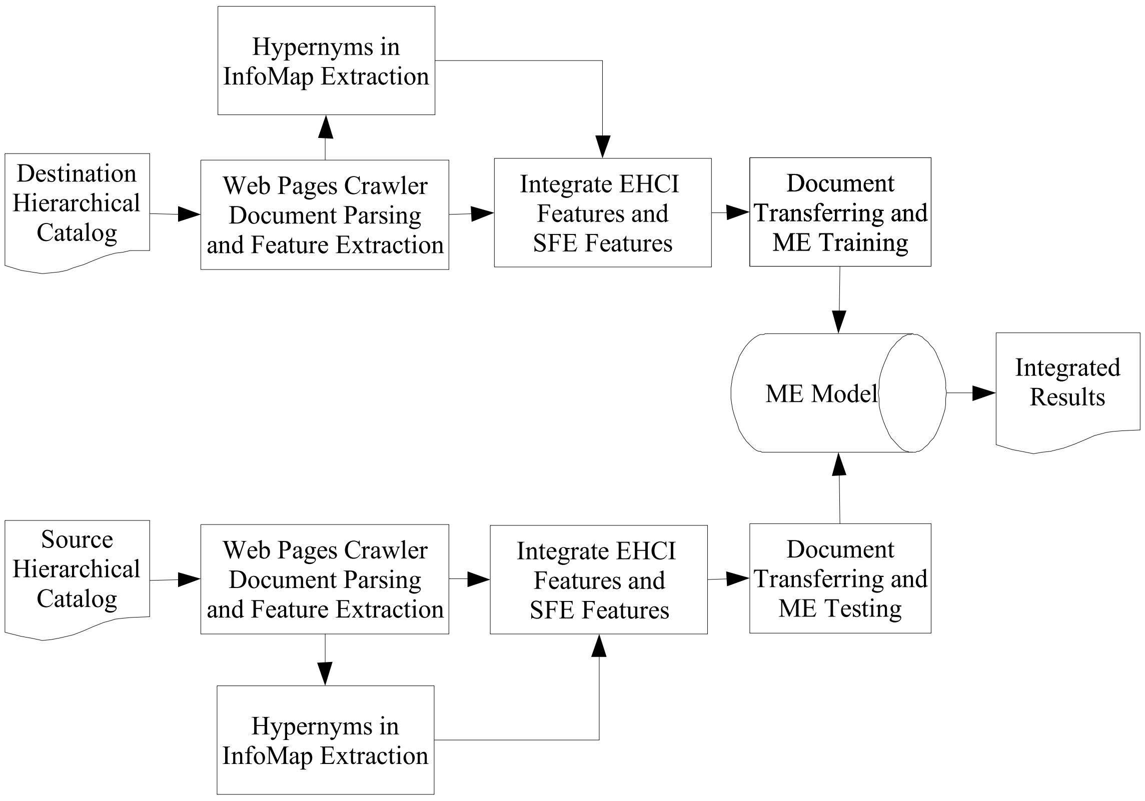

Hypernyms in InfoMap Extraction
Destination Hierarchical Catalog
Web Pages Crawler Document Parsing and Feature Extraction
Integrate EHCI Features and SFE Features
Document Transferring and ME Training
ME Model
Integrated Results
Source Hierarchical Catalog
Web Pages Crawler Document Parsing and Feature Extraction
Integrate EHCI Features and SFE Features
Document Transferring and ME Testing
Hypernyms in InfoMap Extraction

Figure 2. The processing flow for hierarchical taxonomy integration with semantic feature expansion.

4.2 Semantic Feature Expansion

To further improve the integration performance, the semantic information of inter-taxonomy documents is explored in the proposed approach to perform semantic feature expansion (SFE). The main idea is to augment the feature space of each document with representative topic words. As noted in Tseng *et al.* (2006), the hypernyms of documents can be considered as the candidates of the representative topic words for the documents. Hereby, SFE adopts a similar approach to Tseng *et al.* (2006) to first select important term features from the documents and then decide the representative topic terms from hypernyms.

Since feature expansion with hypernyms intends to introduce features that are not related to the document topic, these irrelevant features need to be filtered out before the final integration work. From the aspect of improving integration accuracy, the expanded features that have little discriminative power among categories are considered to be removed. According to previous studies (Ng, Goh, & Low, 1997; Yang & Pedersen, 1997; Tseng, Lin, Chen, & Lin, 2006), although the χ^2-test (chi-square) method is very effective in feature selection for text classification, it cannot differentiate negatively related terms from positively related ones. For a term *t* and a category *c*, their χ^2 measure is defined as:

$$\chi^2(t,c) = \frac{N \times (N_T^+ \times N_T^- - N_F^+ \times N_F^-)^2}{\left(N_T^+ + N_F^-\right)\left(N_F^+ + N_T^-\right)\left(N_T^+ + N_F^+\right)\left(N_F^- + N_T^-\right)} \tag{11}$$

where *N* is the total number of the documents, N_T^+ (N_F^+) is the number of the documents of category *c* (other categories) containing the term *t*, and N_T^- (N_F^-) is the number of the documents of category *c* (other categories) not containing the term *t*.

Therefore, the correlation coefficient (CC) method is suggested to filter out the negatively related terms (Ng, Goh, & Low, 1997; Tseng, Lin, Chen, & Lin, 2006). Since *N* is the same for each term, we can omit it and get the following equation to calculate the CC value for each term:

$$CC(t,c) = \frac{(N_T^+ \times N_T^- - N_F^+ \times N_F^-)}{\sqrt{\left(N_T^+ + N_F^-\right)\left(N_F^+ + N_T^-\right)\left(N_T^+ + N_F^+\right)\left(N_F^- + N_T^-\right)}} \tag{12}$$

Since the categories in a taxonomy are in a hierarchical relationship, SFE only considers the categories of the same parent in the CC method.

Then, the five terms with the highest CC values are selected to perform semantic feature expansion. As indicated by (Ng, Goh, & Low, 1997; Tseng, Lin, Chen, & Lin, 2006), the terms selected with CC are highly representative for a category. The category-specific terms of a source category, however, may not be topic-genetic to the corresponding destination category. Therefore, SFE uses them as the basis to find more topic-indicative terms for each category.

Some lexical dictionaries, such as InfoMap (http://infomap.stanford.edu/) and WordNet (http://wordnet.princeton.edu/), can be used to extract the hypernyms of the category-specific terms to get the topic indicative features of a category. For example, if a category has the following five category-specific terms: *output*, *signal*, *circuit*, *input*, and *frequency*, SFE gets the following hypernyms from InfoMap: *signal*, *signaling*, *sign*, *communication*, *abstraction*, *relation*, *etc*. These hypernyms are more topic-generic than the category specific terms. Then, SFE calculates the weight HW_x of each extracted hypernym *x* by:

$$HW_x = \frac{HF_x}{\sum_{i=1}^{n} HF_i} \tag{13}$$

where HF_x is the term frequency of *x*, and *i* denotes the number of the hypernyms in each category.

For each document d_k, its SFE feature vector $\mathbf{sf}_k$ is changed by extending Equation 10 as follows:

$$\mathbf{sf}_k = \lambda \times \mathbf{l}_k + (1-\lambda) \times \left[\alpha \times \mathbf{h}_k + (1-\alpha) \times \mathbf{f}_k\right] \tag{14}$$

where $\mathbf{l}_k$ denotes the feature vector of the hierarchical thesaurus information computed from the left term of Equation 10, $\mathbf{h}_k$ denotes the feature vector of the topic-generic terms of the category computed from Equation 13, and $\mathbf{f}_k$ denotes the original feature vector of the document derived from the right term of Equation 10.

5. Experimental Analysis

We have conducted experiments with real-world catalogs from Yahoo! and Google to study the performance of the SFE scheme with a Maximum Entropy classification tool from Edinburgh University (ver. 20041229) (Zhang, 2004). Two integration procedures were implemented. The baseline is ME with EHCI (EHCI-ME), and the other is ME with EHCI and SFE (SFE-ME). We measured three scores with different λ and α values: precision, recall, and F_1 measures. Both integration directions were evaluated: from Google to Yahoo! and from Yahoo! to Google. The experimental results show that SFE-ME can effectively improve the integration performance. For recall measures, SFE-ME outperforms EHCI-ME in more than 60% of all cases. For precision measures, SFE-ME outperforms EHCI-ME in more than 90% of all cases. SFE-ME can also achieve the best recall and precision performance. For F_1 measures, SFE-ME outperforms EHCI-ME in nearly 95% of all the cases. The experimental results are detailed in the following.

5.1 Data Sets

In the experiments, five directories from Yahoo! and Google were extracted to form two experimental taxonomies (Y and G). Table 2 shows these directories and the number of the extracted documents after ignoring the documents that could not be retrieved. As in previous studies (Agrawal & Srikan, 2001; Sarawagi, Chakrabarti, & Godbole, 2003; Ho, Chen, & Yang, 2006), the documents appearing in only one category were used as the training data (|Y-G| and |G-Y|), and the common documents were used as the testing data (|Y Test| and |G Test|). Since some documents may appear in more than one category in a taxonomy, |Y Test| is slightly different from |G Test|. For simplicity consideration, the level of each hierarchy was controlled to be at most three in the experiments. If the number of the documents of a certain subcategory was less than 10, the subcategory would be merged upward to its parent category.

Table 2. The experimental categories and the numbers of documents.

Category	Google	|G-Y|	|G Class|	|G Test|	Yahoo!	|Y-G|	|Y Class|	|Y Test|
Autos	/autos/…	1096	12	427	/automotive/…	1681	24	436
Movies	/movies/…	5188	26	1422	/movies_Film/…	7255	27	1344
Outdoors	/outdoors/…	2396	16	208	/outdoors/…	1579	19	210
Photo	/photography/…	615	9	235	/photography/…	1304	23	218
Software	/software/…	5829	27	641	/software/…	1876	25	691
Total		15124	90	2932		13695	108	2918

Before the integration, we used the stopword list in Frakes and Baeza-Yates (1992) to remove the stopwords, and the Porter algorithm (Porter, 1980) for stemming. In the integration process, we allow that each source document d_x can be integrated into multiple destination categories (one-to-many) as we find in real-world taxonomies. Different λ values from 0.1 to 1.0 were applied to the source taxonomy (λ_s) and the destination taxonomy (λ_d). To both taxonomies, the same α value ranging from 0.1 to 1.0 was applied for semantic feature expansion. The lexical dictionary used in the experiments was InfoMap to get hypernyms. As reported in Tseng *et al.* (2006), we believe that WordNet will result in similar hypernym performance.

In the experiments, we measured the integration performance of EHCI-ME and SFE-ME in six scores: macro-averaged recall (MaR), micro-averaged recall (MiR), macro-averaged precision (MaP), micro-averaged precision (MiP), macro-averaged F_1 measure (MaF), and micro-averaged F_1 measure (MiF). The standard F_1 measure is defined as the harmonic mean of recall and precision: $F_1 = 2rp/r + p$, where recall is computed as $r = \frac{\text{correctly integrated documents}}{\text{all test documents}}$ and precision is computed as $p = \frac{\text{correctly integrated documents}}{\text{all predicted positive documents}}$. The micro-averaged scores were measured by computing the scores globally over all categories in five directories. The macro-averaged scores were measured by first computing the scores for each individual category, then averaging these scores. The recall measures are used to reflect the traditional performance measurements on integration accuracy. The precision measures show the degrees of false integration. The standard F_1 measures show the compromised scores between recall and precision.

5.2 Experimental Results and Discussion

Although we have measured the integration performance with different λ values, this paper only lists part of the results in five different λ_d values, which are 0.1, 0.3, 0.5, 0.7, and 0.9. Considering α, we have also measured the integration performance with different values ranging from 0.1 to 1.0. When α is between 0.1 and 0.4, SFE-ME is superior to EHCI-ME. For different integration directions, we found that the optimal α value may be also different. Here, we only report two cases, $\alpha = 0.4$ for integrating documents from Google to Yahoo! and $\alpha = 0.1$ for integrating documents from Yahoo! to Google, in which the SFE approach can show its effectiveness.

Table 3 and Table 4 show the macro-averaged and micro-averaged recall results of EHCI-ME and SFE-ME. The macro-averaged and micro-averaged precision results of EHCI-ME and SFE-ME are listed in Table 5 and Table 6. In Table 7 and Table 8, the macro-averaged and micro-averaged F_1 measure results of EHCI-ME and SFE-ME are listed, respectively.

From Table 3 (a), we can notice that SFE-ME is superior to EHCI-ME in more than 75% of all MaR scores for the integrations from Google to Yahoo!. Although Table 3 (b) shows that SFE-ME can only achieve nearly 40% improvements for the integration from Yahoo! to Google, SFE-ME has consistent MaR performance. Two reasons cause this lower-than-average MaR performance. First, the recall performance of SFE-ME is not as good as EHCI-ME for categories with few positive examples in the Y$\rightarrow$G integration process. This can be justified from the superior MiR performance of SFE-ME. Second, the λ_d weight increasingly mitigates the improvements of SFE in the MaR measures of SFE-ME in a consistent way in the Y$\rightarrow$G integration process. The MiR performance of SFE-ME also has the similar mitigation.

From Table 3, we can also notice that SFE-ME achieves the best MaR of 0.8935 when $\lambda_s = 0.1$ and $\lambda_d = 0.1$ for the G$\rightarrow$Y integration process. Although Table 3 (b) shows that EHCI-ME achieves the best MaR for the Y$\rightarrow$G integration process, SFE-ME indeed achieves higher MaR of 0.9501 in our experiment while $\lambda_s = 0.1, \lambda_d = 0.1$, and $\alpha = 0.4$.

Table 3. The macro-averaged recall (MaR) measures of EHCI-ME and SFE-ME.

	EHCI-ME					SFE-ME ($\alpha = 0.4$)				
λ_d / λ_s	0.10	0.30	0.50	0.70	0.90	0.10	0.30	0.50	0.70	0.90
0.10	0.8023	0.7419	0.7320	0.7334	0.7175	0.8935	0.8618	0.8500	0.8489	0.8678
0.20	0.7636	0.7342	0.7274	0.7331	0.7192	0.7867	0.7845	0.7769	0.7742	0.7950
0.30	0.7481	0.7336	0.7315	0.7333	0.7210	0.7347	0.7501	0.7511	0.7476	0.7539
0.40	0.7422	0.7329	0.7283	0.7313	0.7197	0.7185	0.7367	0.7403	0.7374	0.7398
0.50	0.7362	0.7299	0.7272	0.7301	0.7204	0.7085	0.7310	0.7340	0.7337	0.7346
0.60	0.7317	0.7261	0.7262	0.7292	0.7207	0.7081	0.7284	0.7338	0.7338	0.7338
0.70	0.7262	0.7242	0.7233	0.7263	0.7191	0.6941	0.7227	0.7333	0.7338	0.7338
0.80	0.7231	0.7205	0.7232	0.7253	0.7235	0.6922	0.7208	0.7277	0.7304	0.7338
0.90	0.7192	0.7205	0.7191	0.7262	0.7243	0.6922	0.7146	0.7224	0.7275	0.7304
1.00	0.7186	0.7200	0.7181	0.7216	0.7211	0.7020	0.7138	0.7214	0.7223	0.7243

(a) The results of the integration from Google to Yahoo!

	EHCI-ME					SFE-ME ($\alpha = 0.1$)				
λ_d / λ_s	0.10	0.30	0.50	0.70	0.90	0.10	0.30	0.50	0.70	0.90
0.10	0.9022	0.7937	0.8287	0.8312	0.8290	0.8833	0.8285	0.8165	0.8079	0.8059
0.20	0.8798	0.7817	0.8205	0.8258	0.8242	0.8514	0.8261	0.8138	0.8124	0.8069
0.30	0.8294	0.7787	0.8185	0.8254	0.8228	0.8394	0.8240	0.8138	0.8117	0.8059
0.40	0.8256	0.7777	0.8177	0.8269	0.8214	0.8357	0.8237	0.8144	0.8121	0.8079
0.50	0.8200	0.7769	0.8169	0.8226	0.8201	0.8350	0.8237	0.8141	0.8124	0.8090
0.60	0.8180	0.7761	0.8169	0.8223	0.8217	0.8357	0.8233	0.8141	0.8127	0.8097
0.70	0.8165	0.7771	0.8198	0.8212	0.8204	0.8350	0.8233	0.8144	0.8127	0.8100
0.80	0.8162	0.7768	0.8157	0.8205	0.8189	0.8350	0.8233	0.8151	0.8131	0.8107
0.90	0.8161	0.7735	0.8103	0.8184	0.8157	0.8340	0.8230	0.8151	0.8138	0.8117
1.00	0.8202	0.8585	0.8640	0.8638	0.8633	0.8449	0.8425	0.8367	0.8367	0.8360

(b) The results of the integration from Yahoo! to Google

From table 4, we can notice that SFE-ME is superior to EHCI-ME in more than 60% of all MiR scores for the G→Y integration process and in nearly 75% of all MiR scores for the Y→G integration process. Among these cases, SFE-ME can achieve the best G→Y MiR of 0.9301 and the best Y→G MiR of 0.9055 when $\lambda_s = 0.1$ and $\lambda_d = 0.1$. When $\lambda_d = 0.1$

and $\lambda_s \geq 0.3$, EHCI-ME outperforms SFE-ME for both MaR and MiR in the G→Y integration process. Considering the λ_d influences of Google's hierarchical thesaurus information shown in Table 3 (b), the experimental results suggest that over-emphasizing the weight of Google's hierarchical thesaurus information will impair the effectiveness of SFE.

Table 4. The micro-averaged recall (MiR) measures of EHCI-ME and SFE-ME.

	EHCI-ME					SFE-ME ($\alpha = 0.4$)				
λ_d / λ_s	0.10	0.30	0.50	0.70	0.90	0.10	0.30	0.50	0.70	0.90
0.10	0.8561	0.7999	0.7873	0.7945	0.7718	0.9301	0.9096	0.8972	0.8969	0.9109
0.20	0.8174	0.7807	0.7770	0.7934	0.7732	0.8369	0.8400	0.8325	0.8133	0.8284
0.30	0.7989	0.7797	0.7787	0.7907	0.7746	0.7838	0.8030	0.8058	0.7921	0.7962
0.40	0.7921	0.7797	0.7777	0.7873	0.7742	0.7698	0.7831	0.7917	0.7835	0.7849
0.50	0.7866	0.7787	0.7773	0.7862	0.7746	0.7640	0.7804	0.7821	0.7814	0.7825
0.60	0.7801	0.7773	0.7770	0.7859	0.7742	0.7650	0.7790	0.7818	0.7818	0.7818
0.70	0.7780	0.7766	0.7760	0.7831	0.7739	0.7585	0.7756	0.7814	0.7818	0.7818
0.80	0.7763	0.7739	0.7760	0.7828	0.7763	0.7575	0.7746	0.7780	0.7790	0.7818
0.90	0.7736	0.7739	0.7732	0.7831	0.7766	0.7575	0.7715	0.7760	0.7777	0.7790
1.00	0.7729	0.7736	0.7725	0.7801	0.7749	0.7619	0.7715	0.7753	0.7753	0.7766

(a) The results of the integration from Google to Yahoo!

	EHCI-ME					SFE-ME ($\alpha = 0.1$)				
λ_d / λ_s	0.10	0.30	0.50	0.70	0.90	0.10	0.30	0.50	0.70	0.90
0.10	0.8952	0.8620	0.8535	0.8559	0.8545	0.9055	0.8713	0.8699	0.8696	0.8679
0.20	0.8709	0.8504	0.8490	0.8535	0.8538	0.8768	0.8590	0.8572	0.8613	0.8610
0.30	0.8613	0.8480	0.8487	0.8538	0.8535	0.8651	0.8555	0.8538	0.8579	0.8548
0.40	0.8583	0.8473	0.8477	0.8545	0.8531	0.8610	0.8552	0.8542	0.8572	0.8528
0.50	0.8524	0.8470	0.8473	0.8528	0.8528	0.8596	0.8548	0.8528	0.8552	0.8446
0.60	0.8501	0.8466	0.8473	0.8531	0.8535	0.8593	0.8559	0.8524	0.8531	0.8442
0.70	0.8473	0.8473	0.8504	0.8524	0.8531	0.8579	0.8555	0.8514	0.8453	0.8439
0.80	0.8470	0.8470	0.8483	0.8521	0.8524	0.8572	0.8552	0.8483	0.8432	0.8425
0.90	0.8466	0.8459	0.8442	0.8514	0.8511	0.8562	0.8548	0.8473	0.8429	0.8425
1.00	0.8518	0.8562	0.8624	0.8627	0.8620	0.8552	0.8545	0.8463	0.8425	0.8418

(b) The results of the integration from Yahoo! to Google

From Table 5, we can notice that SFE-ME is superior to EHCI-ME in more than 80% of all MaP for the G→Y integration process, and in all cases for the Y→G integration process. In addition, SFE-ME achieves the best G→Y MaP of 0.6662 when λ_s = 1.0 and λ_d = 0.1, and the best Y→G MaP of 0.4663 when λ_s = 0.7 and λ_d = 0.9.

Table 5. The macro-averaged precision (MaP) measures of EHCI-ME and SFE-ME.

	EHCI-ME					SFE-ME (α = 0.4)				
λ_d / λ_s	0.10	0.30	0.50	0.70	0.90	0.10	0.30	0.50	0.70	0.90
0.10	0.1936	0.3273	0.3356	0.3426	0.3425	0.2122	0.2980	0.3139	0.3158	0.3557
0.20	0.3491	0.3482	0.3475	0.3459	0.3559	0.3664	0.3696	0.3572	0.3477	0.3510
0.30	0.3890	0.3537	0.3486	0.3460	0.3547	0.4707	0.3960	0.3793	0.3523	0.3486
0.40	0.4090	0.3613	0.3497	0.3482	0.3543	0.5794	0.4137	0.3797	0.3723	0.3531
0.50	0.4253	0.3657	0.3515	0.3521	0.3560	0.6279	0.4649	0.3971	0.3778	0.3552
0.60	0.4373	0.3734	0.3565	0.3588	0.3603	0.6613	0.4918	0.4192	0.3556	0.3624
0.70	0.4455	0.3811	0.3611	0.3681	0.3655	0.6600	0.5592	0.4397	0.3663	0.3916
0.80	0.4532	0.3876	0.3686	0.3735	0.3559	0.6607	0.6403	0.4872	0.3876	0.3333
0.90	0.4548	0.3904	0.3747	0.3853	0.3607	0.6636	0.6543	0.5738	0.4321	0.3548
1.00	0.4565	0.4125	0.3862	0.4070	0.3625	0.6662	0.6575	0.5955	0.5043	0.4304

(a) The results of the integration from Google to Yahoo!

	EHCI-ME					SFE-ME (α = 0.1)				
λ_d / λ_s	0.10	0.30	0.50	0.70	0.90	0.10	0.30	0.50	0.70	0.90
0.10	0.0565	0.1643	0.1952	0.1985	0.2004	0.0969	0.3236	0.3816	0.4159	0.4198
0.20	0.0963	0.1969	0.2090	0.2070	0.2090	0.1744	0.3439	0.3997	0.4362	0.4498
0.30	0.1171	0.2047	0.2080	0.2097	0.2120	0.2051	0.3566	0.4041	0.4470	0.4575
0.40	0.1279	0.2050	0.2085	0.2100	0.2126	0.2190	0.3749	0.4083	0.4510	0.4640
0.50	0.1345	0.2032	0.2096	0.2105	0.2145	0.2231	0.3785	0.4100	0.4525	0.4642
0.60	0.1375	0.2034	0.2104	0.2100	0.2164	0.2273	0.3827	0.4106	0.4544	0.4646
0.70	0.1375	0.2042	0.2128	0.2106	0.2153	0.2318	0.3854	0.4120	0.4551	0.4663
0.80	0.1378	0.2052	0.2138	0.2109	0.2149	0.2354	0.3854	0.4132	0.4555	0.4651
0.90	0.1382	0.2055	0.2132	0.2102	0.2121	0.2366	0.3840	0.4147	0.4534	0.4636
1.00	0.1018	0.1012	0.1024	0.1019	0.1017	0.1661	0.1797	0.1847	0.1876	0.1881

(b) The results of the integration from Yahoo! to Google

From Table 6, SFE-ME achieves the best G→Y MiP of 0.6078 when $\lambda_s = 0.9$ and $\lambda_d = 0.1$, and the best Y→G MiP of 0.2988 when $\lambda_s = 0.7$ and $\lambda_d = 0.9$. In addition, SFE-ME achieves MiP improvements in 90% of all cases for the G→Y integration process and in all cases for the Y→G integration process. These results show that the number of incorrectly integrated documents in SFE-ME is much lower. With high precision performance, SFE-ME may reduce a lot of time for users in manually verifying the integration correctness.

Table 6. The micro-averaged precision (MiP) measures of EHCI-ME and SFE-ME.

	EHCI-ME					SFE-ME ($\alpha = 0.4$)				
λ_s \ λ_d	0.10	0.30	0.50	0.70	0.90	0.10	0.30	0.50	0.70	0.90
0.10	0.1156	0.2504	0.2715	0.2740	0.2817	0.1205	0.2835	0.3099	0.2782	0.3687
0.20	0.2253	0.3018	0.2947	0.2822	0.3080	0.1569	0.3661	0.3570	0.3504	0.3629
0.30	0.2741	0.3170	0.2984	0.2858	0.3107	0.2737	0.3777	0.3946	0.3515	0.3642
0.40	0.3136	0.3329	0.3002	0.2897	0.3115	0.4721	0.3834	0.3776	0.3866	0.3688
0.50	0.3494	0.3390	0.3033	0.2695	0.3135	0.5581	0.4556	0.3862	0.3879	0.3666
0.60	0.3763	0.3475	0.3101	0.3101	0.3199	0.6061	0.4663	0.4032	0.3147	0.3700
0.70	0.3906	0.3583	0.3192	0.3336	0.3317	0.6041	0.4924	0.3952	0.3180	0.4151
0.80	0.3966	0.3759	0.3334	0.3485	0.3414	0.6016	0.5824	0.4335	0.3229	0.3452
0.90	0.3987	0.3826	0.3402	0.3734	0.3540	0.6078	0.5871	0.4726	0.3509	0.3800
1.00	0.3992	0.4332	0.3772	0.4198	0.3568	0.5999	0.5894	0.4937	0.3879	0.3974

(a) The results of the integration from Google to Yahoo!

	EHCI-ME					SFE-ME ($\alpha = 0.1$)				
λ_s \ λ_d	0.10	0.30	0.50	0.70	0.90	0.10	0.30	0.50	0.70	0.90
0.10	0.0677	0.1269	0.1172	0.1118	0.1111	0.0943	0.1818	0.2089	0.2344	0.2686
0.20	0.0999	0.1226	0.1145	0.1121	0.1120	0.1183	0.1903	0.2184	0.2485	0.2829
0.30	0.1087	0.1186	0.1137	0.1123	0.1122	0.1269	0.1929	0.2278	0.2534	0.2860
0.40	0.1125	0.1158	0.1131	0.1124	0.1119	0.1318	0.1948	0.2299	0.2585	0.2909
0.50	0.1142	0.1152	0.1127	0.1122	0.1121	0.1365	0.1984	0.2303	0.2606	0.2943
0.60	0.1147	0.1131	0.1123	0.1121	0.1118	0.1418	0.2042	0.2304	0.2612	0.2983
0.70	0.1147	0.1128	0.1122	0.1119	0.1117	0.1469	0.2131	0.2301	0.2597	0.2988
0.80	0.1150	0.1134	0.1121	0.1118	0.1116	0.1503	0.2152	0.2312	0.2591	0.2985
0.90	0.1151	0.1127	0.1116	0.1117	0.1110	0.1522	0.2154	0.2340	0.2581	0.2988
1.00	0.0979	0.0867	0.0865	0.0870	0.0865	0.1190	0.1388	0.1417	0.1468	0.1573

(b) The results of the integration from Yahoo! to Google

For many applications, a compromised performance may be required with a high F_1 score. From Table 7 and Table 8, we can notice that SFE-ME is superior to EHCI-ME in nearly 90% of all MaF and MiF scores for the G→Y integration process, and it has consistent improvements in all cases for the Y→G integration process. In our experiments with $\alpha = 0.4$, SFE-ME achieves the highest MaF (0.6839) and the highest MiF (0.6764) when $\lambda_s = 0.6$ and $\lambda_d = 0.1$ for the G→Y integration process. For the Y→G integration process, SFE-ME achieves the highest MaF (0.5919) and the highest MiF (0.4413) when $\alpha = 0.1$, $\lambda_s = 0.7$, and $\lambda_d = 0.9$. These two tables show that the SFE scheme can mostly get more balanced improvements in both recall and precision considerations.

Table 7. The macro-averaged F_1 (MaF) measures of EHCI-ME and SFE-ME.

	EHCI-ME					SFE-ME ($\alpha = 0.4$)				
λ_d / λ_s	0.10	0.30	0.50	0.70	0.90	0.10	0.30	0.50	0.70	0.90
0.10	0.3119	0.4556	0.4602	0.4670	0.4637	0.3430	0.4428	0.4585	0.4603	0.5046
0.20	0.4792	0.4724	0.4703	0.4700	0.4761	0.5000	0.5025	0.4894	0.4799	0.4870
0.30	0.5118	0.4773	0.4722	0.4702	0.4755	0.5738	0.5184	0.5041	0.4789	0.4767
0.40	0.5274	0.4840	0.4725	0.4718	0.4748	0.6415	0.5299	0.5020	0.4948	0.4780
0.50	0.5391	0.4872	0.4739	0.4751	0.4765	0.6658	0.5684	0.5154	0.4988	0.4789
0.60	0.5475	0.4932	0.4782	0.4810	0.4804	0.6839	0.5871	0.5336	0.4790	0.4852
0.70	0.5522	0.4994	0.4817	0.4886	0.4847	0.6766	0.6305	0.5497	0.4887	0.5106
0.80	0.5572	0.5040	0.4884	0.4931	0.4771	0.6761	0.6782	0.5836	0.5065	0.4584
0.90	0.5572	0.5064	0.4927	0.5035	0.4816	0.6776	0.6831	0.6396	0.5422	0.4776
1.00	0.5583	0.5245	0.5022	0.5205	0.4825	0.6836	0.6845	0.6525	0.5939	0.5399

(a) The results of the integration from Google to Yahoo!

	EHCI-ME					SFE-ME ($\alpha = 0.1$)				
λ_d / λ_s	0.10	0.30	0.50	0.70	0.90	0.10	0.30	0.50	0.70	0.90
0.10	0.1064	0.2723	0.3160	0.3205	0.3227	0.1746	0.4654	0.5201	0.5492	0.5521
0.20	0.1737	0.3145	0.3332	0.3310	0.3334	0.2895	0.4856	0.5361	0.5676	0.5776
0.30	0.2053	0.3242	0.3317	0.3344	0.3372	0.3296	0.4978	0.5400	0.5765	0.5837
0.40	0.2215	0.3245	0.3322	0.3349	0.3378	0.3471	0.5153	0.5439	0.5800	0.5895
0.50	0.2311	0.3221	0.3336	0.3352	0.3400	0.3521	0.5187	0.5454	0.5813	0.5899
0.60	0.2355	0.3223	0.3346	0.3345	0.3426	0.3574	0.5225	0.5459	0.5829	0.5904
0.70	0.2354	0.3234	0.3379	0.3352	0.3411	0.3629	0.5251	0.5472	0.5834	0.5919
0.80	0.2358	0.3247	0.3388	0.3356	0.3405	0.3672	0.5251	0.5484	0.5839	0.5911
0.90	0.2364	0.3248	0.3376	0.3345	0.3367	0.3686	0.5236	0.5497	0.5823	0.5902
1.00	0.1811	0.1810	0.1831	0.1823	0.1820	0.2776	0.2962	0.3026	0.3064	0.3071

(b) The results of the integration from Yahoo! to Google

We have also measured these six scores for the λ_s = 0.0, λ_d = 0.0, and α = 0.0 cases, which means that the integration is performed by only ME without EHCI and SFE enhancements. In this configuration, for the G→Y integration process, ME can achieve very prominent recall performance in MaR (0.9578) and MiR (0.9616) but with poor precision performance in MaP (0.0111) and MiP (0.0111). Its MaF and MiF are 0.022 and 0.0219, respectively. For the Y→G integration process, ME has similar performance. Although ME can attain the best recall performance, these results show that it allows many documents of other categories to be incorrectly integrated.

Table 8. The micro-averaged F_1 (MiF) measures of EHCI-ME and SFE-ME.

	EHCI-ME					SFE-ME (α = 0.4)				
λ_d / λ_s	0.10	0.30	0.50	0.70	0.90	0.10	0.30	0.50	0.70	0.90
0.10	0.2037	0.3814	0.4037	0.4075	0.4128	0.2133	0.4322	0.4607	0.4246	0.5249
0.20	0.3533	0.4353	0.4273	0.4163	0.4406	0.2642	0.5099	0.4997	0.4897	0.5047
0.30	0.4082	0.4508	0.4314	0.4199	0.4435	0.4058	0.5138	0.5298	0.4869	0.4998
0.40	0.4493	0.4666	0.4332	0.4236	0.4443	0.5852	0.5148	0.5113	0.5177	0.5018
0.50	0.4838	0.4723	0.4363	0.4306	0.4463	0.6450	0.5753	0.5170	0.5184	0.4993
0.60	0.5077	0.4803	0.4433	0.4447	0.4527	0.6764	0.5834	0.5320	0.4487	0.5023
0.70	0.5201	0.4904	0.4523	0.4679	0.4644	0.6725	0.6024	0.5249	0.4521	0.5422
0.80	0.5250	0.5060	0.4664	0.4823	0.4742	0.6706	0.6649	0.5568	0.4565	0.4789
0.90	0.5262	0.5121	0.4725	0.5057	0.4863	0.6744	0.6668	0.5874	0.4835	0.5108
1.00	0.5264	0.5554	0.5069	0.5458	0.4887	0.6713	0.6682	0.6032	0.5171	0.5257

(a) The results of the integration from Google to Yahoo!

	EHCI-ME					SFE-ME (α = 0.1)				
λ_d / λ_s	0.10	0.30	0.50	0.70	0.90	0.10	0.30	0.50	0.70	0.90
0.10	0.1258	0.2212	0.2061	0.1977	0.1967	0.1707	0.3009	0.3369	0.3692	0.4102
0.20	0.1792	0.2142	0.2017	0.1981	0.1980	0.2085	0.3115	0.3481	0.3857	0.4259
0.30	0.1930	0.2081	0.2005	0.1986	0.1983	0.2213	0.3148	0.3596	0.3913	0.4286
0.40	0.1989	0.2038	0.1995	0.1986	0.1978	0.2287	0.3174	0.3623	0.3972	0.4338
0.50	0.2014	0.2028	0.1989	0.1983	0.1982	0.2355	0.3220	0.3627	0.3995	0.4365
0.60	0.2021	0.1996	0.1983	0.1981	0.1977	0.2434	0.3298	0.3627	0.4000	0.4408
0.70	0.2021	0.1991	0.1982	0.1978	0.1975	0.2508	0.3413	0.3623	0.3973	0.4413
0.80	0.2025	0.2000	0.1980	0.1976	0.1974	0.2558	0.3439	0.3633	0.3964	0.4408
0.90	0.2027	0.1989	0.1972	0.1975	0.1964	0.2584	0.3441	0.3667	0.3952	0.4412
1.00	0.1757	0.1575	0.1572	0.1580	0.1572	0.2089	0.2387	0.2428	0.2501	0.2650

(b) The results of the integration from Yahoo! to Google

The experimental results show that SFE-ME can get more improved integration performance with the SFE scheme. Compared with EHCI-ME, SFE-ME shows that the semantic information of the hypernyms of the category-specific terms can be used to facilitate the integration process between two hierarchical taxonomies.

6. Conclusion

In recent years, the taxonomy integration problem has been progressively studied for integrating two homogeneous hierarchical taxonomies. Many types of implicit information embedded in the source taxonomy are explored to improve the integration performance. The semantic information embedded in the source taxonomy, however, has not been discussed in previous research.

In this paper, an enhanced integration approach (SFE) is proposed to exploit the semantic information of the hypernyms of the category-specific terms. Augmented with these additional semantic category features, the source documents can be more precisely integrated into the correct destination category in the experiments. The experimental results show that SFE-ME can achieve the best macro-averaged F_1 score and the best micro-averaged F_1 score. The results also show that the SFE scheme can get precision and recall enhancements in a significant portion of all cases.

There are still some issues left for future study. For example, the effectiveness of SFE on other classification schemes, such as SVM and NB, may need to be investigated to decide which one has the best integration performance. In addition, deciding the optimal parameter configuration is a classical classification problem which is also important to the taxonomy integration problem. Although mining more valuable implicit information can be a tough challenge, we believe that the integration performance can be further improved with appropriate assistance of more effective auxiliary information and advanced classifiers.

Acknowledgement

This work was supported in part by National Science Council of R.O.C. under grant NSC 96-2422-H-006-002 and NSC 96-2221-E-155-067. The authors would also like to express their gratitude to the anonymous reviewers for their precious comments.

References

Agrawal, R., & Srikan, R. (2001). On Integrating Catalogs. in *Proceedings of the 10th International Conference on World Wide Web*, 603-612.

Berger, A. L., Pietra, V. J. D., & Pietra, S. A. D. (1996). A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 39-71.

Chen, I.-X., Ho, J.-C., & Yang, C.-Z. (2005). An Iterative Approach for Web Catalog Integration with Support Vector Machines. in *Proceedings of the 2nd Asia Information Retrieval Symposium (AIRS 2005)*, 703-708.

Chen, I.-X., Ho, J.-C., & Yang, C.-Z. (2007). Hierarchical Web Catalog Integration with Conceptual Relationships in a Thesaurus. *International Journal of Computational Linguistics and Chinese Language Processing*, 12(2), 155-174.

Cheng, T.-H., & Wei, C.-P. (2008). A Clustering-based Approach for Integrating Document-Category Hierarchies. *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, 38(2), 410-424.

Darroch, J. N., & Ratcliff, D. (1972). Generalized Iterative Scaling for Log-linear Models. *Annals of Mathematical Statistics*, 43, 1470-1480.

Frakes, W., & Baeza-Yates, R. (1992). *Information Retrieval: Data Structures and Algorithms*, 1st edition, Prentice Hall, PTR.

Ho, J.-C., Chen, I.-X., & Yang, C.-Z. (2006). Learning to Integrate Web Catalogs with Conceptual Relationships in Hierarchical Thesaurus. in *Proceedings of the 3rd Asia Information Retrieval Symposium (AIRS 2006)*, 217-229.

Hsu, M.-H., Tsai, M.-F., & Chen, H.-H. (2006). Query Expansion with ConceptNet and WordNet: an Intrinsic Comparison. in *Proceedings of the 3rd Asia Information Retrieval Symposium (AIRS 2006)*, 1-13.

Krikos, V., Stamou, S., Kokosis, P., Ntoulas, A., & Christodoulakis, D. (2005). DirectoryRank: Ordering Pages in Web Directories. in *Proceedings of the 7th ACM International Workshop on Web Information and Data Management (WIDM 2005)*, 17-22.

Ng, H.-T., Goh, W.-B., & Low, K.-L. (1997). Feature selection, Perception Learning, and a Usability Case Study for Text Categorization. in *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 67-73.

Sarawagi, S., Chakrabarti, S., & Godbole, S. (2003). Cross-training: Learning Probabilistic Mappings between Topics. in *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 177-186.

Tseng, Y.-H., Lin, C.-J., Chen, H.-H., & Lin, Y.-I. (2006). Toward Generic Title Generation for Clustered Documents. in *Proceedings of the 3rd Asia Information Retrieval Symposium (AIRS 2006)*, 145-157.

Wu, C.-W., Tsai, T.-H., & Hsu, W.-L. (2005). Learning to Integrate Web Taxonomies with Fine-Grained Relations: A Case Study Using Maximum Entropy Model. in *Proceedings of the 2nd Asia Information Retrieval Symposium (AIRS 2005)*, 190-205.

Wu, C.-W., Tsai, T.-H., Lee, C.-W., & Hsu, W.-L. (2008). Web Taxonomy Integration with Hierarchical Shrinkage algorithm and Fine-Grained Relations. *Expert Systems with Applications*, 35(4), 2123-2131.

Yang, Y., & Pedersen, J. O. (1997). A Comparative Study on Feature Selection in Text Categorization. in *Proceedings of the 14th International Conference on Machine Learning (ICML'97)*, 412-420.

Zhang, D., & Lee, W.-S. (2004a). Web Taxonomy Integration using Support Vector Machines. in *Proceedings of the 13th International Conference on World Wide Web*, 472-481.

Zhang, D., & Lee, W.-S. (2004b). Web Taxonomy Integration Through Co-Bootstrapping. in *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 410–417.

Zhu, S., Yang, C. C., & Lam, W. (2004). CatRelate: A New Hierarchical Document Category Integration Algorithm by learning Category Relationships. in *Proceedings of the 7th International Conference on Asian Digital Libraries (ICADL 2004)*, Shanghai, China, 280-289.

Online Resources

Information Mapping Project, Computational Semantics Laboratory, Stanford University. http://infomap.stanford.edu/.

The Porter Stemming Algorithm., http://tartarus.org/˜martin/PorterStemmer.

WordNet, A lexical database for the English language: Cognitive Science Laboratory, Princeton University, http://wordnet.princeton.edu/.

Zhang, L., "Maximum Entropy Modeling Toolkit for Python and C++," http://homepages.inf.ed.ac.uk/s0450736/maxent.html.

Computational Linguistics and Chinese Language Processing
Vol. 13, No. 4, December 2008, pp. 443-460

Automatic Wikibook Prototyping via Mining Wikipedia

Jen-Liang Chou[*], and Shih-Hung Wu[*]

Abstract

Wikipedia is the world's largest collaboratively edited source of encyclopedic knowledge. Wikibook is a sub-project of Wikipedia that is intended to create a book that can be edited by various contributors, similar to how Wikipedia is composed and edited. Editing a book, however, requires more effort than editing separate articles. Therefore, methods of quickly prototyping a book is a new research issue. In this paper, we investigate how to automatically extract content from Wikipedia and generate a prototype of a Wikibook as a start point for further editing. Applying search technology, our system can retrieve relevant articles from Wikipedia. A table of contents is built automatically and is based on a two-stage searching method. Our experiments show that, given a keyword as the title of a book, our system can generate a table of contents, which can be treated as a prototype of a Wikibook. Such a system can help free textbook editing. We propose an evaluation method based on the comparison of system results to a traditional textbook and show the coverage of our system.

Keywords: Wikipedia, Wikibook, Table of Contents Generation

1. Introduction

The ability to quickly construct a free encyclopedia, such as Wikipedia, has shown that the Web 2.0 has been successful. Community and interactivity among users on the Internet has become a popular topic. A project named "*Science Online,*" which brought the wiki scheme to schools, lets students participate in collaborative writing (Forte & Bruckman, 2007). Wikipedia is useful in college education, both for general topics (Lally & Dunford, 2007, May/June) and for specific topics, such as physics (Muchnik, Itzhack, Solomon, & Louzoun, 2007). In this paper, we focus on another project of the Wikimedia Foundation, Wikibook, which is also useful in the classroom (Sajjapanroj, Bonk, Lee, & Lin, 2006). Wikibook provides free textbooks on the Internet via the Wiki system, letting global users edit the

[*] Department of Computer Science and Information Engineering, Chaoyang University of Technology, Taiwan R.O.C.
E-mail :shwu@cyut.edu.tw

[Received September 15, 2008; Revised Mar. 19, 2009; Accepted Mar. 25, 2009]

contents of textbooks. Creating a book without supporting data, however, is difficult. An expert or a system that can provide a general framework and useful references for a book is of much help. Thus, we propose an automatic Wikibook prototyping system that can pick relevant articles from Wikipedia and construct a hierarchy as the table of contents for a given topic. Our system consists of information retrieval and web mining technology.

In previous work, the TOC and anchor text of a Wikipedia entry has been used to form the TOC of a Wikibook (Yang, Han, Oh, & Kwak, 2007). A relevant research area is Topic Maps. Topic Maps are analogous to the Table of Contents (TOC) of a textbook. Users can realize and memorize the relevant concepts of a topic. "Topic Maps for learning" (TM4L) (Dicheva & Dichev, 2005) is an application of Topic Maps. These methods, though, mostly rely on humans without using information retrieval technology, which can provide a considerable amount of relevant information. Studies in knowledge acquisition provide some hints. Semi-automatic methods can aggregate a domain ontology via Internet search (Roberson & Dicheva, 2007).

We propose a method that can help prototype a Wikibook automatically using the contents from Wikipedia. In the following sections, we will describe our methodology in Section 2, system implementation in Section 3, experimental results in Section 4, discussions in Section 5, and give conclusion in the final section.

2. Methodology

We propose a framework for Wikibook prototyping which involves several modules. These modules can be replaced to fulfill the needs of different requirements. For example, we might customize the system for different languages, users of different ages, or topics with different contexts.

2.1 Corpus Preparation

The first module is the preparation of a corpus. We can use the whole of Wikipedia, certain language versions, or subsets as the searching target. The system then extracts and analyzes contents from the corpus. As a knowledge source, Wikipedia provides not only the content but also a lot of links to contexts, which are also valuable. We will extract keywords from the content of Wikipedia pages, and will find related terms from the anchor text in these pages.

2.2 Search Engine and Anchor Text Miner

The second module is a search engine and an anchor text miner. As mentioned above, relevant topics can be found not only from a full text keyword search, but also from links in Wikipedia. This module is important from the technical point of view. Our system searches relevant topics and their hyponyms using information retrieval technology. On the other hand, the anchor text miner can extract related terms from the anchor texts of the retrieved pages.

2.3 Hierarchical Construction

Given relevant topics, our system then generates a hierarchy. This hierarchy can be viewed as the table of contents of a Wikibook for further editing.

3. Implementation issues

Our methodology gives a general idea on how to generate a Wikibook automatically within a flexible framework. In the following sections, we discuss our system and experiments on computer science topics in both the English and Chinese versions of Wikipedia. Figure 1 shows the architecture of our system.

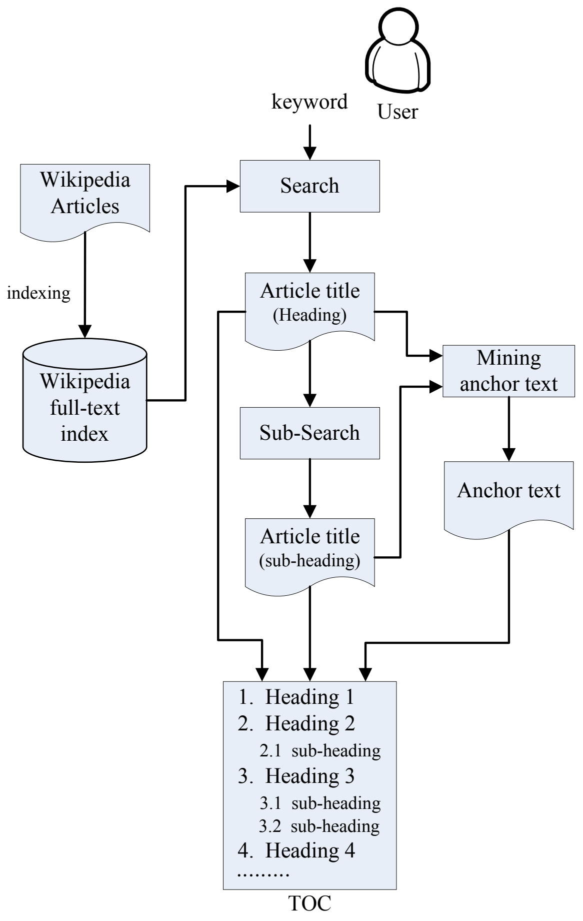


Figure 1. System Architecture

3.1 Corpus Choice

We chose Wikipedia as our corpus in the experiments for two reasons. First, the idea of the availability of the corpus; it is an ample resource of information which is available for every potential Wikibook editor and there are no copyright issues in regards to using the content for a Wikibook. Second, the quality of content: there is more professional knowledge in Wikipedia than in general Websites, and potential editors can compile the content from Wikipedia with less effort.

3.2 Search Strategy

Our search strategy is automatic two-stage iterative searching. The system takes a term as the query and performs context searching via a standard information retrieval process. We use pseudo-relevance feedback as our searching algorithm. The resulting set of the first search will be used as the corpus in the second stage.

The system ranks the search results according to a traditional TF-IDF scoring function that is restated in Formula (1) where $Score_i$ is the ranking score of an article, i denotes an article, j denotes a term occurring within the query, and T denotes the number of terms in the query. TF_j is the frequency of j occurring within the article title, TF_{ij} is the frequency that term j occurs in article i. We assume $token_{ij} = \left| \left\{ T_i - j \right\} \right|$ as the number of the rest words after deleting term j from the article i, where T_i is the number of terms in the article i:

$$Score_i = \sum_{j}^{T} \left(\frac{TF_j \times IDF_j}{\sqrt{\sum_j \left(TF_j \times IDF_j \right)^2}} \right) \times \left(\frac{TF_{ij} \times IDF_j}{\sqrt{token_j}} \right) \tag{1}$$

where $IDF_j = \log \left(\frac{|D|}{|D_j|} + 1 \right)$, measures term j involvement in other documents, D is all the articles in the index, D_j is articles that contain j.

Table 1 shows a collection of four documents, and the corresponding value of each variable in Formula (1) is shown in Table 2. Suppose the query terms are “A B C”, then T is 3, and i can be 1 to 4 as the Doc ID in Table 1. Then, the Score of a document according to the formula is shown in the last column of Table 2. Where $Score_1 = 0.551$; $Score_2 = 0.260$; $Score_3 = 0.675$; $Score_4 = 0$, we rank these scores from high to low. From this, we can attain results such that, if we input the query “A B C”, the system will output the order of the document as 3, 1, 2, 4.

Table 1. Example Documents

Doc ID	Title	Contents
1	A	A B A E C
2	B	B D F
3	C	C A C E
4	D	D E F B

Table 2. Calculate the value of each notion using Formula (1)

i	j	TF_j	TF_{ij}	$\|D\|$	$\|D_j\|$	IDF_j	$token_{ij}$	$Score_i$
1	A	1	2	4	2	0.477	3	0.551
1	B	0	1	4	3	0.368	4	0
1	C	0	1	4	2	0.477	4	0
2	A	0	0	4	2	0.477	3	0
2	B	1	1	4	3	0.368	2	0.260
2	C	0	0	4	2	0.477	3	0
3	A	0	1	4	2	0.477	3	0
3	B	0	0	4	3	0.368	4	0
3	C	1	2	4	2	0.477	2	0.675
4	A	0	0	4	2	0.477	4	0
4	B	0	1	4	3	0.368	3	0
4	C	0	0	4	2	0.477	4	0

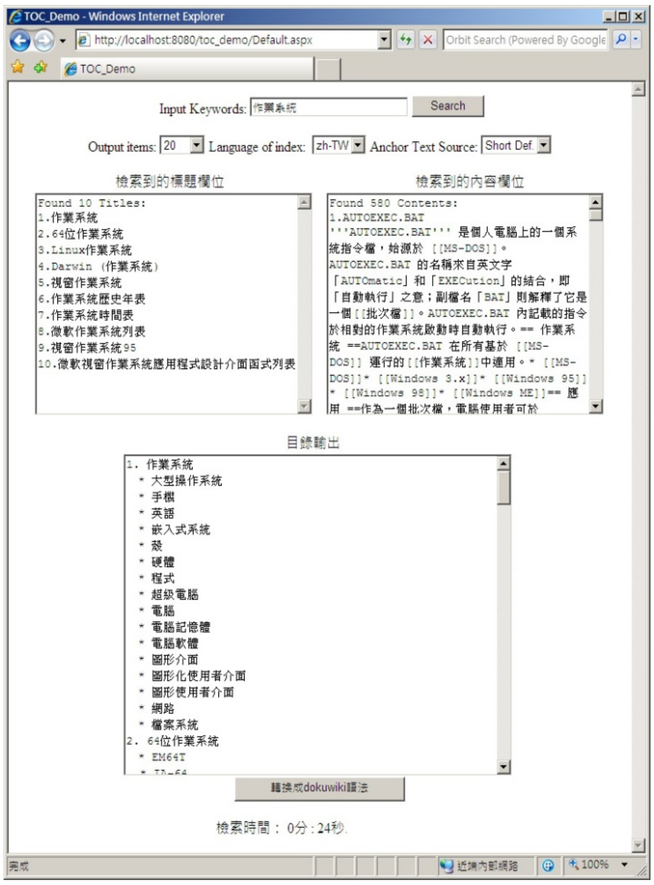
This means the frequency of a term occurring both in the title and in the article is important. The higher the term frequency is, the higher the score of the article will be.

After filtering out the noise in the resulting set, such as redirected pages, our system maintains the top N documents, which are the highest score articles, as the resulting set for further search and the user can customize the arbitrary parameter N. Our system thus finds the first level of relevant topics from the resulting set. These relevant topics can be treated as the backbone of a Wikibook.

3.3 Sub-Topic Finding

Since the extracted topics from the first search often contain the original query term, our system removes the original query term string and uses the reduced topics as the query terms in the second stage search based on the result of the first stage search. For example, if we input the keyword “Operating system (作業系統)” and retrieve the topic “Linux Operating System (Linux 作業系統)”, the shortened query will be “Linux”. As another example, if we input the keyword “Operating system (作業系統)” and retrieve the topic “Windows Operating System (視窗作業系統),” the shortened query will be “Windows (視窗)”. This query reformulation is the same both in the English version and the Chinese version.

Our system extracts keywords from the second stage search result as the sub-level topics of the output TOC. This is a recursive method; we can find the sub-sub-topics in the same manner. For example, we can further search for sub-topics of “Linux” or “Windows”. After finding the sub-level topics, our system will extract other related terms from the anchor text in these Wikipedia articles. As such, every related topic we find corresponds to a related Wikipedia article, which might be useful content of a Wikibook. There are two approaches for mining the related terms: one is to extract anchor texts only in the short definition of an article; the other is to extract from the whole article. Our system then combines these related terms as the sub-level topics.


TOC_Demo - Windows Internet Explorer
http://localhost:8080/toc_demo/Default.aspx
Orbit Search (Powered By Google
TOC_Demo
Search
檢索到的標題欄位
檢索到的內容欄位
目錄輸出
轉換成dokuwiki語法
近端內部網路
100%

Input Keywords: 作業系統

Output items: 20 Language of index: zh-TW Anchor Text Source: Short Def.

```
Found 10 Titles:
1.作業系統
2.64位作業系統
3.Linux作業系統
4.Darwin (作業系統)
5.視窗作業系統
6.作業系統歷史年表
7.作業系統時間表
8.微軟作業系統列表
9.視窗作業系統95
10.微軟視窗作業系統應用程式設計介面函式列表
```

```
Found 580 Contents:
1.AUTOEXEC.BAT
'''AUTOEXEC.BAT''' 是個人電腦上的一個系
統指令檔，始源於 [[MS-DOS]]。
AUTOEXEC.BAT 的名稱來自英文字
「AUTOmatic」和「EXECution」的結合，即
「自動執行」之意；副檔名「BAT」則解釋了它是
一個[[批次檔]]。AUTOEXEC.BAT 內記載的指令
於相對的作業系統啟動時自動執行。== 作業系
統 ==AUTOEXEC.BAT 在所有基於 [[MS-
DOS]] 運行的[[作業系統]]中適用。* [[MS-
DOS]]* [[Windows 3.x]]* [[Windows 95]]
* [[Windows 98]]* [[Windows ME]]== 應
用 ==作為一個批次檔，電腦使用者可於
```

```
1. 作業系統
  * 大型操作系統
  * 手機
  * 英語
  * 嵌入式系統
  * 殼
  * 硬體
  * 程式
  * 超級電腦
  * 電腦
  * 電腦記憶體
  * 電腦軟體
  * 圖形介面
  * 圖形化使用者介面
  * 圖形使用者介面
  * 網路
  * 檔案系統
2. 64位作業系統
  * EM64T
  * IA-64
```

檢索時間： 0分：24秒.

完成

Figure 2. User Interface

4. Experiment

4.1 Dataset Preparation

We downloaded the English and Chinese version of Wikipedia dump data from the Wikimedia downloads website (http://download.wikimedia.org/enwiki/). Then, we parsed the content and stored it in a MySQL database. In the following experiment, our system uses the data under the title and content XML tags. A searching tool is built based on the Lucene open source information retrieval API (http://lucene.apache.org/). The user interface is shown in Figure 2.

Our system can generate a TOC for any given title without other information. To evaluate the quality of our system, we conducted several experiments to find differences with previous work. As the target of comparison, we first observe a manually edited Wikibook entitled "Operating System Design"; part of the TOC is shown in Table 3. The first four level one topics in this TOC are general concepts, which are independent of the title of the Wikibook. The sub-topics of and after the fifth topic, however, are very likely to have been generated automatically. We also find that the sub-topic of the fifth topic, "Kernel Architecture," states the concept of "Kernel Architecture" or gives some illustrations. Our system can help to automatically generate this part of a Wikibook TOC.

Table 3. Partial view of the Wikibook TOC of "Operating System Design", which was edited manually

```
1. Preface
2. Introduction
3. Case studies
4. History
5. Kernel Architecture
   5.1 Monolithic Kernels
      5.1.1 Solaris
      5.1.2 Linux
      5.1.3 Windows 9x
   5.2Microkernels
      5.2.1 QNX
   5.3Exokernel
      5.3.1 XOK
   5.4 Hybrid Kernels
      5.4.1 Windows NT/XP
      5.4.2 Mac OSX
      5.4.3 BeOS
6. Initialisation
   6.1 Boot Loaders
   6.2 Hardware Initialisation
7. Processes
…
```

4.2 Experiment 1: Generate the TOC of "Operating System Design" via our System

As the first example, to derive a TOC about operating systems, we input the query term "Operating system," into our system. The system automatically generates a corresponding TOC, shown in Table 4. The arbitrary N can be 10, 20, 50, 100, or 1000. In our experiment, we choose the Top 20 (N=20) as our output number. The user can choose different N for greater or fewer topics in our system. The sub-topics are ranked according to Formula (1), discussed in Section 3.2. Next, the system uses the scheme described in Section 3.3 to perform a second stage search. In this case, the sub-topics at the second level are adequate. For example, the sub-topics of "Linux operating system" in Table 4 are instances of the Linux operating system. The scheme in Section 2.3 is effective.

Table 4. Partial view of automatically generated TOC of "Operating System"

1. Operating system
2. THE operating system
3. IBM AIX (operating system)
4. CPM operating system
5. Disk operating system
6. Kent Applicative Operating System
7. Linux operating system
7.1 Distro
7.2 Flask operating system
7.3 Sabayon
7.4 Red hatter
8. Operating system advocacy
9. Real-time operating system
9.1 Rubus (disambiguation)
9.2 RTMOS (Real-Time Multiprogramming Operating System)
10. VSE (operating system)
10.1 Exec
10.2 Protected procedure call
11. Computer network operating system
11.1 Faceless process
12. Network operating system
12.1 Faceless process
12.2 Brazil (operating system)
13. Solaris (operating system)
14. Pick Operating System
15. Darwin (operating system)
16. Operating system/kernel
16.1 Flask operating system

Table 5. Partial view of automatically generated TOC of the Chinese version "作業系統(Operating System)"

1.作業系統
2.64位作業系統
3.Linux作業系統
3-1.Ebuntu
4.Darwin (作業系統)
5.視窗作業系統
5-1.AUTOEXEC.BAT
5-2.JavaOS
5-3.Xfwm
5-4.WINGs Display Manager
6.作業系統歷史年表
7.作業系統時間表
8.微軟作業系統列表
9.視窗作業系統95
10.微軟視窗作業系統應用程式設計介面函式列表

Table 5 shows the result of the same experiment in the Chinese version of Wikipedia. The result is similar but with less recall. Since the size of the Chinese version of Wikipedia is much smaller than the English version of Wikipedia, this is a reasonable result. Table 6 shows the results of searching and mining the anchor text, which is described in Section 2.3. As we can see, the topic has * notion, meaning it is the anchor text, and the results show more topic information differences with Table 3.

Table 6. Partial view of automatically generated TOC of the Chinese version "Operating System", plus mining anchor texts from short definition

```
1.作業系統
    * 大型操作系統
    * 手機
    * 英語
    * 記憶體
    * 嵌入式系統
    * 殼
    * 硬體
…
2.64 位作業系統
    * EM64T
    * IA-64
    * Linux
    * Windows
…
3.Linux 作業系統
    * 386
    * Apache
    * DEV C++
    * GNOME
    * GNU  工程
…
  3-1.Ebuntu
    * 12 月 4 日
    * 2006 年
    * Edubuntu
…
4.Darwin (作業系統)
    * 2000 年
    * 2002 年 4 月
    * 2003 年 7 月
    * 2005 年 5 月
…
```

4.3 Experiment 2: Using the crawling method to generate the TOC of "Operating System Design"

We perform the same experiment with the crawling method described in the literature (Yang, Han, Oh, & Kwak, 2007), which takes a full TOC of a manually edited Wikipedia entry as the input, and outputs a more detailed TOC. The partial result is shown in Tables 7 and 8.

Table 7. Partial result of the TOC generated by the crawling method for the English version of "operating system"

```
1 Technology
    1.1 Program execution
        * Kernel
        * Process(computing)
    1.2 Interrupts
        * interrupt
        * kernel
        * register
        * stack
…
2 Security
        * classified information
        * emulates
        * file transfer protocols
        * firewalls
        * Government Department of Defense
        * p-code
        * Popek and Geldberg virtualization requirment
        * sandbox
        * Trusted Computer System Evaluation Criteria
    2.1 Example: Microsoft Windows
        * access privileges
        * administrator
    …
3 File system support in modern operating systems
    3.1 Linux and UNIX
        * ext2
        * ext3
        * FAT
        * GFS
        * GFS2
        * HFS
        * ISO 9660
   …
```

```
4 Graphical user interfaces
          * CDE
          * context switch
          * COSE
          * GNOME
          * graphical user interface
      …
5 History
          * 80286
          * AmigaOS
          * batch processing
          * Control Data Corporation
     …
6 Mainframes
          * ALGOL
          * B5000
          * Burroughs Corporation
      …
```

Table8. Partial result of the TOC generated by the crawling method for the Chinese version of "operating system"

```
1  歷史
        *  工作排序
        *  分時機制
        *  分散式系統
        *  批次樣式
     1.1 1980 年代前
        * 1947 年
        * 1963 年
        * 1964 年
        * 1970 年代
        * AT&T
        * C 語言
        * Direct access storage device
        * IBM System/360
        * Maurice V. Wilkes
        * Multics
        * Multics
        * Unix
        *  大型電腦
…
```

4.4 Evaluation by Comparing to Traditional Textbook

To measure the coverage of our system results over a traditional textbook, we compare the generated TOC to the TOC of a traditional textbook and show the hits and the precision. Table 9 is a partial TOC adopted from a textbook "Operating System Concepts" (Silberschatz, Galvin, & Gagne, 2001). A traditional textbook provides a suitable coverage, such that we can estimate how much content can be put into one book.

As in the TOC of the current Wikibook in Table 3, this TOC begins with general topics which can be used for each book. Then, it follows the main ideas, which are the most important topics. The order of the topics might be different according how the authors rank them. Nevertheless, there should be some causality, from problem to solution or from general idea to special case. In order to achieve such a goal (finding more relationships between topics (Völkel, Krötzsch, Vrandecic, Haller, & Studer, 2006), more AI technology might be involved, such as frame or logical inference. A predefined frame might guide a system to search general information for a Wikibook. Logic can be used to infer the causality between topics.

Table 9. TOC of an "Operating System" Textbook

1	Introduction
	1.1 What Is an Operating System?
	1.2 Mainframe Systems
…	
2	Computer-System Structures
	2.1 Computer-System Operation
…	
3	Operating-System Structures
	3.1 System Components
…	
4	Processes
	4.1 Process Concept
…	
Chapter 5	Threads …

We separate the number of matching hits and precision rate of the first level and second level in the following table. Table 10 shows the result using the English version. The number of first level and second level topics in the textbook is 23 and 177, respectively. Table 11 shows the results using the Chinese version. The number of first level and second level topics in the textbook is 22 and 191, respectively. We adopt two matching criteria: rigid for exact matching and relaxed for partial matching. Table 12 shows the evaluation results of English version using the crawling method, and Table 13 shows that of the Chinese version.

In this paper, we perform three experiments on our system. Searching only method cannot provide the user more precise concepts because of the low number of hits. The searching method plus mining short definitions, though, can give more hits than the above and attain better precision. On the other hand, the searching method plus mining full document definitions can give better hits, but lose some precision. The result is very similar to the result of crawling method, which requires a manually edited TOC as input.

Comparing our method plus mining anchor text with crawling method, we discuss two directions. In the first level, we can gain the same number of hits. In the second level, if the user wants fewer sub-topics for further editing, mining short definitions can archive precision close to crawling method. On the other hand, if the user wants to get more relative subtopics, mining full document definitions attains more hits than crawling method. In the Chinese version, we can get the same results.

Table 10. Hits and precision of the TOC of "Operating system"

	Searching method			Searching plus mining Short definition			Searching plus mining full document definition		
	# of output	Hits	Precision	# of output	Hits (Short)	Precision	# of output	Hits (Long)	Precision
First Level	20	**5**	0.250						
Second Level (Relax)	17	3	0.176	340	68	**0.200**	2154	**284**	0.132
Second Level (Rigid)		2	0.118		36	**0.106**		**93**	0.043

Table 11. Hits and precision of the TOC of Chinese version "Operating system"

	Searching method			Searching plus mining Short definition			Searching plus mining full document definition		
	# of output	Hits	Precision	# of output	Hits (Short)	Precision	# of output	Hits (Long)	Precision
First Level	10	2	0.200						
Second Level (Relax)	5	1	0.200	179	43	**0.240**	1273	**150**	0.118
Second Level (Rigid)		0	0.000		21	**0.117**		**61**	0.048

Table 12. Hits and precision of the TOC of "Operating system" using crawling method

	Searching method [18]		
	# of output	Hits	Precision
First Level	13	**5**	0.385
Second Level (Relax)	406	105	0.259
Second Level (Rigid)		46	0.113

Table 13. Hits and precision of the TOC of Chinese version "Operating system" using crawling method

	Searching method [18]		
	# of output	Hits	Precision
First Level	8	**3**	0.375
Second Level (Relax)	280	54	0.193
Second Level (Rigid)		30	0.107

5. Discussion and Future Work

According to the observations in the previous section, we would like to discuss issues that might improve the results of automatic Wikibook prototyping.

5.1 Fast Prototyping of Wikibooks

Speeding up editing collaboratively is our goal. Our system can generate a prototype of a Wikibook on any topic. We present our system to depict the TOC in relevant concepts of a book, and it looks like the TOC in general books. It will let editors get relevant concepts of the given topic more quickly and with a good starting point. The TOC generated by our system can be a prototype for editors to revise. Thus, it saves time and stimulates interest in editors to revise it. Since the topics are related articles in Wikipedia, they also provide contents of that topic. Thus, our system not only provides TOC but also some related content.

5.2 Identification of Hypernym/Hyponym Relation

To know the relations between relevant topics is very important in this application, especially the relation between the upper and lower concepts, known as the hypernym/hyponym relation.

With this relation, a tree structure can be built and a hierarchy formed (Nguyen, Matsuo, & Ishizuka, 2007). Supervised (Aggarwal, Gates, & Yu, 1999) or semi-supervised methods (Huang, Zhang, & Lam, 2006) of hierarchical clustering algorithms are promising methods. After finding a set of relevant documents, a system may be clustered into a hierarchy according to the content. The titles of these articles can be treated as the TOC. A knowledge-based approach is also possible. We can try to identify the hypernym/hyponym relation between relevant titles by using WordNet (Farreres, Rodríguez, & Gibert, 2002) or SUMO.

5.3 Importance of an Article

Currently, our system ranks relevant documents according to the scoring function of Lucene. We might combine the result with Google's PageRank (Page, Brin, Motwani, & Winograd, 1999). Each page of the Wikipedia entry contains a link to a page that reports how many entries link to this entry, *cf.* Figure 3. The more inward the link is, the higher the importance of this entry is. With the analysis of the link relation, importance can be ranked (Wissner-Gross, 2006). This information helps to decide whether the TOC should contain this entry or not. The relatedness of words in Wikipedia might also help (Ponzetto & Strube, 2007). In the future, we will acquire more TOC of good Wikibooks as a training set to develop a better process via machine learning algorithm. Clustering algorithms are promising tools for this task.

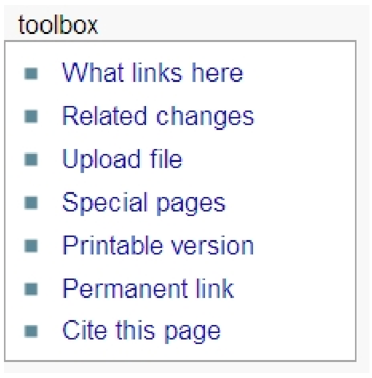


Figure 3. "What links here" in each page of Wikipedia entry links to a page that reports all the entries that link to this entry

6. Conclusion

We proposed an automatic process that can generate a Wikibook by mining the content of Wikipedia. Our method involves document searching, keyword extraction, related term mining, and hierarchy construction technology. We built an experimental system and conducted primary experiments using both English and Chinese. The results showed the automatically generated TOC might help the community to edit a Wikibook more rapidly.

Previous works have not evaluated their system. In this paper, we proposed a method to evaluate the result by comparing to the traditional textbook and report the hits and precision. This gives a standard to compare systems. We find that using the anchor text from the short definition or the full article of Wikipedia will give better precision and more useful terms. The major improvement over previous work is that we do not need a manually edited TOC. Our system uses a searching mechanism that requires only the title as a search keyword, rather than using the full TOC in a manually edited article.

Acknowledgement

This research was partly supported by the National Science Council under NSC 96-2221-E-324-046.

References

Aggarwal, C. C., Gates, S. C., & Yu, P. S. (1999). On the merits of building categorization systems by supervised clustering. In *Proceedings of the Fifth ACM SIGKDD international Conference on Knowledge Discovery and Data Mining*. KDD '99., 352-356.

The Apache Software Foundation. (2008). Lucene Project. Retrieved December 26, 2007, http://lucene.apache.org/

Dicheva, D., & Dichev, C. (2005). Authoring educational topic maps: can we make it easier? In *Proceedings of the Fifth IEEE International Conference on Advanced Learning Technologies*. ICALT'06., 216-218.

Forte, A., & Bruckman, A. (2007). Constructing text:: Wiki as a toolkit for (collaborative?) learning. In *Proceedings of the 2007 international Symposium on Wikis*. WikiSym '07., 31-42.

Farreres, J., Rodríguez, H., & Gibert, K. (2002). Semiautomatic creation of taxonomies. In *Coling-02 on Semanet: Building and Using Semantic Networks - Volume 11* International Conference On Computational Linguistics., 1-7.

Huang, R., Zhang, Z., & Lam, W. (2006). Refining hierarchical taxonomy structure via semi-supervised learning. In *Proceedings of the 29th Annual international ACM SIGIR Conference on Research and Development in information Retrieval.* SIGIR '06., 653-654.

Lally, A. M., & Dunford, C. E. (2007, May/June). Using Wikipedia to Extend Digital Collections. *D-Lib Magazine*, 13.

Muchnik, L., Itzhack, R., Solomon, S., & Louzoun, Y. (2007). Self-emergence of knowledge trees: Extraction of the Wikipedia hierarchies. *The American Physical Society*, Phys. Rev. E 76, 016106.

Nguyen, Dat P.T., Matsuo, Y., & Ishizuka, M. (2007). Subtree Mining for Relation Extraction from Wikipedia. In *Proceeding of NAACL/HLT 2007*, Companion Volume, 125-128.

Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). The pagerank citation ranking: Bringing order to the web. *Technical Report*, Stanford University.

Ponzetto, S. P., & Strube, M. (2007). An API for Measuring the Relatedness of Words in Wikipedia. *Proceedings of the ACL 2007 Demo and Poster Sessions*, 49-52.

Roberson, S., & Dicheva, D. (2007). Semi-automatic ontology extraction to create *draft* topic maps. In *Proceedings of the 45th Annual Southeast Regional Conference*. ACM-SE 45., 100-105.

Sajjapanroj, S., Bonk, C., Lee, M., & Lin, G. (2006). The Challenges and Successes of Wikibookian Experts and Want-To-Bees. In T. Reeves & S. Yamashita (Eds.), *Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education* 2006, 2329-2333.

Silberschatz, A., Galvin, P. B., & Gagne, G. (2001). *Operating System Concepts, Sixth Edition*. John Wiley & Sons.

Völkel, M., Krötzsch, M., Vrandecic, D., Haller, H., & Studer, R. (2006). Semantic Wikipedia. In *Proceedings of the 15th international Conference on World Wide Web*. WWW '06., 585-594.

Wissner-Gross, A. D. (2006). Preparation of Topical Reading Lists from the Link Structure of Wikipedia. In *Proceedings of the Sixth IEEE international Conference on Advanced Learning Technologies*. ICALT'05., 825-829.

Wikimedia Foundation. (2008). Wikimedia Downloads / enwiki. Retrieved December 26, 2007, http://download.wikimedia.org/enwiki/

Yang, J., Han, J., Oh, I., & Kwak, M. (2007). Using Wikipedia technology for topic maps design. In *Proceedings of the 45th Annual Southeast Regional Conference*. ACM-SE 45, 106-110.