Learning Ontology Resolution for Document Representation and its Applications in Text Mining*

Lidong Bing[♮,§], Bai Sun[♮], Shan Jiang[♮], Yan Zhang[♮†], Wai Lam[§]

[♮]Department of Machine Intelligence
Peking University
Beijing 100871, China
{sunbai, jsh}@pku.edu.cn
zhy@cis.pku.edu.cn

[§]Department of Systems Engineering and
Engineering Management
The Chinese University of Hong Kong
Shatin, Hong Kong
{ldbing, wlam}@se.cuhk.edu.hk

ABSTRACT

It is well known that synonymous and polysemous terms often bring in some noises when calculating the similarity between documents. Existing ontology-based document representation methods are static, hence, the chosen semantic concept set for representing a document has a fixed resolution and it is not adaptable to the characteristics of a document collection and the text mining problem in hand. We propose an Adaptive Concept Resolution (ACR) model to overcome this issue. ACR can learn a concept border from an ontology taking into consideration of the characteristics of a particular document collection. Then this border can provide a tailor-made semantic concept representation for a document coming from the same domain. Another advantage of ACR is that it is applicable in both classification task where the groups are given in the training document set, and clustering task where no group information is available. Furthermore, the result of this model is not sensitive to the model parameter. The experimental results show that ACR outperforms an existing static method significantly.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: Miscellaneous

General Terms

Algorithms, Theory

*Supported by NSFC under Grant No. 60673129 and 60773162, 863 Program under Grant No. 2007AA01Z154, the 2008/2009 HP Labs Innovation Research Program, and National Key Technology R&D Pillar Program in the 11th Five-year Plan of China (Research No: 2009BAH47B00).
Also supported by grants from the Research Grant Council of the Hong Kong SAR, China (Project No: CUHK4128/07). This work is also affiliated with the Microsoft-CUHK Joint Laboratory for Human-centric Computing and Interface Technologies.

†Corresponding author

CIKM'10, October 26–30, 2010, Toronto, Ontario, Canada.

Keywords

ontology, adaptive concept resolution

1. INTRODUCTION

Some carefully edited ontologies include WordNet [7], Cyc [6], Mesh [9] and etc. Previous empirical results have shown improvement in some cases utilizing these ontologies [4, 8, 9]. However, the existing works have an obvious shortcoming: the strategies they adopted are static. For example, one strategy is to use each synonym set in the WordNet as one dimension in the representation vector of the documents. Therefore, the resolution for representing the documents belonging to different collections is the same. Suppose we have two document collections, the first one has coarse granularity categories, such as sports, military and etc, while the second one has finer granularity categories, such as football, basketball and etc. In the first collection, we should consider football players and basketball players are related, while in the second one they should be unrelated. So an adaptive strategy should outperform the static one.

In this paper, the proposed Adaptive Concept Resolution (ACR) model can learn a concept border from an ontology taking into consideration of the characteristics of a particular document collection. Then this border can provide a tailor-made semantic concept representation for a document coming from the same domain. The structure of an ontology is a hierarchical directed acyclic graph[1] (refer to the example in Figure 1), and the border is a cross section in the graph. All the concepts located below the border will be merged into one of the concepts on the border. We use a gain value to measure whether a concept is a good candidate for the border. The gain value is calculated based on the characteristics of the given document collection. Therefore, our model can generate different tailor-made borders for different collections adaptively. Another advantage of ACR is that it is applicable in both classification task where the groups are given in the training document set, and clustering task where no group information is available. To do so, we only need to change the granularity, that is either cluster or individual document, for calculating the gain value. So ACR can be applied to both classification and clustering. The experimental results show that our model can outperform an existing static method in almost all cases.

[1]A hierarchical directed acyclic graph is a directed acyclic graph with the layer information on each node. The head node of an edge must have a higher layer than the tail.

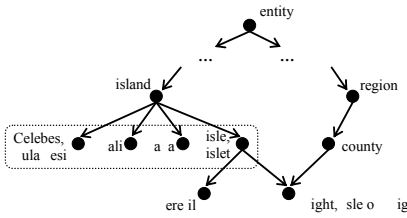

Figure 1: A fragment of WordNet structure. Each node is a concept, and the label is its synonym set.

2. ADAPTIVE CONCEPT RESOLUTION

2.1 Preliminary and Overview

Concept is the basic component of an ontology, and each concept refers to an abstract entity or a real entity. We give a formal definition of a concept:

DEFINITION 2.1. ***Concept**: A concept π is a quadruple $\langle id, \Omega, \sigma, \Upsilon \rangle$, and the items indicate its ID, synonym set, gloss, and hyponym concept set respectively. We refer to the items with $\pi.id$, $\pi.\Omega$, $\pi.\sigma$ and $\pi.\Upsilon$.*

In this paper, we select WordNet2.1 as an instance of the ontology. In Figure 1, a fragment of WordNet is given. Take the concept "island" as an example, Ω is {island}, σ is "a land mass that is surrounded by water", and its hyponym concepts are shown in the dashed box. The first term in a synonym set can be used to refer to the concept.

Some concepts may have more than one hypernyms, as exemplified by "Wight" at the bottom right in Figure 1. We call this kind of concept *ambiguous concept*. Therefore, ontology's structure is a Hierarchical Directed Acyclic Graph (HDAG). The depth $d(\pi)$ for the π is defined as follows:

$$d(\pi) = \begin{cases} 0 & if\ \pi = root, \\ \max_{\pi' \in \{\pi' | \pi \in \pi'.\Upsilon\}} d(\pi') + 1 & otherwise. \end{cases} \quad (1)$$

Figure 2 depicts an overview of the Adaptive Concept Resolution (ACR) model, which has two main parts, namely, the learning part on the left and the utilizing part on the right. In the learning part, given a document collection and an ontology graph, the algorithm learns which concepts have better information gain and generates the elements for the border $\mathcal{B}$. Each concept in $\mathcal{B}$ encapsulates all its descendants shown in the original ontology graph. For example, the terms in π_1 and π_2 are added into $\pi.\Omega$ to get a derived concept π^b. Thus, the border is tailor-made for the given document collection. In the border utilizing part, $\mathcal{B}$ is used to represent a document coming from the same domain collection, and each concept in $\mathcal{B}$ is one dimension in the vector. These two parts will be discussed in detail in the following subsections. Other technical details, such as *virtual concept* for solving the unbalance problem, recursive calculation of the gain value and time complexity can be found in our technical report [1].

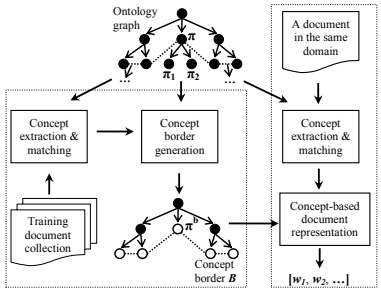

Figure 2: Framework of ACR model.

2.2 Concept Border Generation

2.2.1 Gain Function

To generate the concept border, the leaf concept is merged into its hypernym π recursively, and we define $gain(\pi)$ to measure whether the merging is profitable:

$$gain(\pi) = \frac{\frac{1}{|\pi.\Upsilon|}\sum_{\pi' \in \pi.\Upsilon} Gentropy(\pi')}{Gentropy(\pi)}, \quad (2)$$

in which $Gentropy(\pi)$ is defined as:

$$\begin{aligned} Gentropy(\pi) &= entropy(\Omega_\pi^g) \\ &= -\sum_{u_i \in D} p(u_i|\Omega_\pi^g) \log p(u_i|\Omega_\pi^g), \end{aligned} \quad (3)$$

where $\Omega_\pi^g = \pi.\Omega \cup_{\pi' \in \Pi_\pi^d} \pi'.\Omega$, and $\Pi_\pi^d = \{\pi' | \pi \rightsquigarrow \pi'\}$ is the descendant set of π. $D = \{u_1, u_2, \cdots\}$ is a document set. If there exist clusters in D, each u_i represents a cluster. Otherwise, each u_i represents a single document. And $p(u_i|\Omega_\pi^g)$ is the probability of u_i given Ω_π^g, which is calculated as in Equation 4:

$$p(u_i|\Omega_\pi^g) = \frac{\sum_{t_j \in \Omega_\pi^g} w_{i,j}}{\sum_{u_k \in D, t_j \in \Omega_\pi^g} w_{k,j}}, \quad (4)$$

where $w_{i,j}$ is the weight of t_j in u_i.

It can be observed that $0 < gain(\pi) \leq 1$. The larger the gain value is, the less the noise is brought in because of merging π' into π. If $gain(\pi) \geq \theta$ (profitable threshold), the merging will be performed. Until now, we discuss the problem in a bottom-up fashion (generalization). We can also consider in a top-down fashion (specialization) using merging operation instead of splitting operation, and $gain(\pi)$ can still be used in the same way. We name these two methods as GBG-g and GBG-s respectively.

2.2.2 Gain-based Border Generation (GBG)

GBG-g is summarized in Algorithm 1. In each loop, we attempt to merge the deepest leaves into their hypernyms. First we get the deepest leaf π^l (line 5), and locate π^l's hypernym π. If π^l is an ambiguous concept, we select its hypernym which has the largest depth (line 6). If π contains non-leaf and leaf hyponyms at the same time, these hyponyms will not be merged into π, and set the border flag under π (line 8). Otherwise, if it is profitable to merge π's leaves into it, all leaves' synonym sets will be added into $\pi.\Omega$ (line 11), then delete these leaves from G to make π become a leaf (line 12). If the merging is not profitable, we set the border flag under π (line 14). Finally, the border is composed of all leaves with $flg = true$. In the subprocedure *set_border_flag*, the unambiguous leaves of π become the members in $\mathcal{B}$ (line 8), while the ambiguous leaves will be removed from $\pi.\Upsilon$ and its depth is reset based on the depth definition (Equation 1). Suppose an ambiguous leaf π' has two hypernyms. After removed from $\pi.\Upsilon$, π' becomes an unambiguous leaf, and can be treated as an ordinary leaf hereafter in the remaining processing of GBG-g. Note that the depth of π' should be reset based on its remaining hypernym. The larger the depth of π''s hypernym is, the earlier the hypernym is considered. Thus, we always try to merge π' into its more specialized hypernym.

GBG-s is summarized in Algorithm 2. A recursive splitting operation is performed from the root. If using π to represent all of its descendants is profitable enough, we will encapsulate its descendants into π first (line 3 of *top_down*), then add π into $\mathcal{B}$ (line 4 of *top_down*). Otherwise each of

Algorithm 1 GBG-g

```
1: input: the HDAG G of an ontology, threshold θ
2: output: concept border B
3: each concept has a flag flg, and is false initially
4: while G has leaves with flg = false do
5:    get the deepest leaf π^l with flg = false
6:    get π among π^l's hypernyms, which is the deepest
7:    if π has non-leaf hyponym concepts then
8:       set_border_flag(π)
9:    else
10:      if gain(π) ≥ θ then
11:         set π.Ω ← π.Ω ∪_{π'∈π.Υ} π'.Ω
12:         set π.Υ ← {}
13:      else
14:         set_border_flag(π)
15:      end if
16:   end if
17: end while
18: set B = {π|π's flg is true}
1: proc set_border_flag(π)
2: for all π' in π.Υ do
3:    if |π'.Υ| = 0 then
4:       if π' is ambiguous then
5:          delete π' from π.Υ
6:          reset the depth of π'
7:       else
8:          set flg ← true for π'
9:       end if
10:   end if
11: end for
```

π's hyponyms will be used as the parameter to invoke the *top_down* procedure (line 8 of *top_down*). Once an ambiguous concept is merged into any one of its hypernyms (direct or undirect), it will be removed from G (line 5 of *top_down*).

Before π is added into $\mathcal{B}$, all terms contained by π's descendants are encapsulated into π, see line 11 in Algorithm 1 and line 3 of *top_down* in Algorithm 2. As a result, the descendants' semantic meanings are merged into π. This merging is performed under the guidance of $gain(\pi)$, which guarantees the trade-off is profitable.

2.3 Concept-based Document Representation

Before feeding documents into border learning part or representing a document into a concept vector, it is necessary to extract concepts from the documents. We propose a forward maximum cutting method to extract the concepts. After that, a context matching method is used to find the correct matching for an ambiguous concept. The details of concept extraction and matching are given in the technical report [1].

In the vector space model, each concept π_i in $\mathcal{B}$ is one dimension. Similar to TF-IDF, we introduce CF-IDF to indicate the importance of π_i in a certain document d_j, calculated as $cfidf_{i,j} = cf_{i,j} \times idf_i$, in which $cf_{i,j} = \frac{f_{i,j}}{\sum_{\pi_k \in d_j} f_{k,j}}$,

Algorithm 2 GBG-s

```
1: input: the HDAG G of an ontology, threshold θ
2: output: concept border B
3: top_down(root)
1: proc top_down(π)
2: if gain(π) ≥ θ then
3:    set π.Ω ← π.Ω ∪_{π'∈{π'|π⇝π'}} π'.Ω
4:    put π into B
5:    remove all concepts in {π'|π ⇝ π'} from G
6: else
7:    for all π' in π.Υ do
8:       top_down(π')
9:    end for
10: end if
```

Table 1: Details of the data sets.

	No. of Doc.	No. of Cate.	Categories
NG20	19,997	20	ALL
TREC	12,637	20	354, 362, 365, 376, 393, 394, 397, 398, 401, 417, 422, 423, 432, 433, 434, 442, 446, 617, 625, 627
ODP	5,000	5	Arts, Business, Computers, Health, Sports
MED	1,870	101	ALL

and $idf_i = \log \frac{|D|}{1+|\{d|\pi_i \in d\}|}$, where $f_{i,j} = \sum_{t_l \in \pi_i.\Omega} n_{l,j}$ is the frequency of π_i in d_j ($n_{l,j}$ is the frequency of the term t_l in d_j), $\pi_i \in d$ means that at least one term in $\pi_i.\Omega$ is contained by the document d.

3. EXPERIMENTS

A previous method, known as the "only" strategy, in [4] is implemented for conducting the comparison. In this strategy, each concept is used as one dimension in the document vector, and its weight is decided by the terms in the synonym set. The strategy is called "Hotho" in this paper.

Four data sets are used in the experiments: 20 Newsgroups (NG20), TREC data extracted from the document collection Disc 5, ODP page set and OHSUMED (MED). The details are given in Table 1.

3.1 Mining Tasks and Evaluation Criteria

In document classification, LibSVM [2] with linear kernel is employed to conduct the classification, and 5-fold cross-validation is adopted. Note that the *gain* calculation only needs the training set. Because there exist many small clusters in MED, we do not use this data set in classification. In document clustering, K-Means algorithm is used to perform the clustering. The entire document collection is used as the input information of the *gain* value calculation. Each u_i refers to an individual document.

The commonly used F-measure (including macro- and micro-average) and purity are used to evaluate the results of classification and clustering respectively.

3.2 Results and Parameter Analysis

The results are given Table 2. We can see that except for the F^{ma} of GBG-s on NG20, ACR dominates all other cases in classification. Especially on the TREC data, both of our methods can improve the existing method Hotho about 6%. GBG-g performs better than GBG-s on NG20 and TREC, while GBG-s achieves a better result on the ODP data. In clustering experiment, considering NG20, TREC and ODP, the improvements are more significant, about 4% to 8%. Of the first three data sets, the performances of GBG-g and GBG-s are similar. For the fourth data MED, GBG-g outperforms GBG-s by more than 6%. So the tailer-made border for document representation is much better than the static one in both kinds of text mining task.

Table 2: Performance comparison

		NG20		TREC		ODP		MED
		F^{mi}	F^{ma}	F^{mi}	F^{ma}	F^{mi}	F^{ma}	N/A
Classification	Hotho	.904	.793	.631	.580	.783	.653	N/A
	GBG-g	.934	.818	.696	.643	.809	.677	N/A
	GBG-s	.907	.780	.692	.635	.825	.689	N/A
Clustering	Hotho	.752		.449		.783		.711
	GBG-g	.808		.516		.825		.795
	GBG-s	.807		.536		.830		.731

The effect of θ in the clustering is shown in Table 3. The performance of GBG-g algorithm is not sensitive to θ. It

is because the GBG-g algorithm merges the concepts in a bottom-up fashion, and each merging is performed among the concepts with high semantic relation to each other. Therefore, the consistency of the semantic meaning of a derived new concept can be guaranteed, even when the θ value is small. When the θ value is too large, say 1.0, the constraint becomes too strict, and the related concepts cannot be merged sufficiently. Consequently, the result is not as good as the result under a smaller θ, say 0.7. GBG-s is relatively more sensitive to θ than GBG-g. When θ is small, the top-down splitting will stop early at some general concepts, and these concepts are added into $\mathcal{B}$. As a result, the general meaning of the concepts in $\mathcal{B}$ brings in more noises to the similarity calculation. So in GBG-s, a larger θ value can achieve better results than a smaller value in general. We use 0.7 and 0.9 as θ values for GBG-g and GBG-s respectively. Generally speaking, GBG-g is better and more stable than GBG-s. It is because GBG-g considers the specialized concepts first in generating the concept border, which are more important than the general ones from the semantic point of view. Furthermore, GBG-g can deal with the unbalanced structure more effectively, because it does not merge the leaf concepts with the non-leaf concepts.

The effect of θ in the classification is shown in Table 4. Again we find that GBG-s is more sensitive to θ than GBG-g because of the same reasons discussed above. Without exception, the best results for both GBG-g and GBG-s are achieved when θ is 1. Under the predefined cluster granularity of calculating the gain value, a larger θ can prevent the concepts which may bring in much noise to be added into $\mathcal{B}$. At the same time, because the *gain* value is calculated considering the cluster information, the related semantic meaning in the same cluster can still be merged. Thus, the value of θ used in both GBG-g and GBG-s is 1 in the classification. Interestingly, we find that on NG20, GBG-g can perform slightly better with both small and large θ values than with the medium values. One possible reason is that after the concepts are sufficiently merged under a small θ, the benefit obtained for calculating the similarity within a cluster overwhelms the noises brought in at the same time. While a larger θ achieves a better result by suppressing the amount of noise. For other data sets, this exceptional situation does not happen. Therefore, we adopt a larger θ value for all data sets.

4. RELATED WORK

As an important expert-edited ontology, WordNet has been used to improve the performance of clustering and classification. Hotho et al. [3, 4] showed that incorporating the synonym set and the hypernym as background knowledge into the document representation can improve the clustering results. Jing et al. [5] constructed a term similarity matrix using WordNet to improve text clustering. However, their approach only uses synonyms and hyponyms, and fails to handle polysemy, and breaks the multi-word concepts into a group of single words. In Recupero's work [8], two strategies, namely, WordNet lexical categories (WLC) technique and WordNet ontology (WO) technique, are used to create a new vector space with low dimensionality for the documents. In WLC, 41 lexical categories for nouns and verbs are used to construct the feature vector. As a result the vector has 41 dimensions. In WO, the hierarchical structure is used as the ontology information. Then words are grouped based on the ontology they are related to.

Table 3: Parameter θ's effect in the clustering.

	GBG-g				GBG-s			
θ	NG20	TREC	ODP	MED	NG20	TREC	ODP	MED
0.1	.795	.494	.807	.766	.710	.447	.729	.599
0.2	.795	.503	.804	.740	.726	.450	.732	.754
0.3	.765	.502	.824	.753	.779	.503	.718	.744
0.4	.792	.499	.815	.772	.781	.507	.751	.756
0.5	.800	.501	.815	.780	.755	.515	.781	.734
0.6	.799	.500	.819	.773	.795	.497	.785	.761
0.7	.808	.516	.825	.795	.792	.500	.780	.762
0.8	.803	.531	.828	.770	.789	.521	.766	.739
0.9	.806	.502	.837	.762	.807	.536	.830	.731
1.0	.802	.497	.809	.785	.805	.535	.826	.741

Table 4: Parameter θ's effect in the classification.

	GBG-g						GBG-s					
	NG20		TREC		ODP		NG20		TREC		ODP	
θ	F^{mi}	F^{ma}	F^{mi}	F^{ma}	F^{mi}	F^{ma}	F^{mi}	F^{ma}	F^{mi}	F^{ma}	F^{mi}	F^{ma}
0.1	.930	.814	.659	.607	.796	.666	.824	.708	.537	.493	.688	.575
0.2	.932	.818	.655	.604	.802	.671	.820	.699	.530	.486	.692	.579
0.3	.930	.816	.657	.608	.799	.669	.830	.715	.544	.499	.657	.549
0.4	.921	.808	.665	.613	.805	.674	.875	.749	.524	.479	.666	.555
0.5	.917	.802	.655	.604	.791	.661	.887	.764	.565	.515	.764	.639
0.6	.919	.804	.669	.617	.796	.666	.883	.757	.595	.538	.802	.670
0.7	.917	.803	.664	.613	.792	.662	.902	.772	.642	.587	.806	.673
0.8	.930	.816	.680	.626	.807	.675	.898	.772	.666	.615	.799	.667
0.9	.930	.812	.686	.633	.807	.675	.896	.768	.685	.631	.800	.667
1.0	.934	.818	.696	.643	.809	.677	.907	.780	.692	.635	.825	.689

5. CONCLUSIONS

In this paper, we propose an adaptive concept resolution model to adaptively learn a concept border from an ontology taking into consideration of the characteristics of a particular document collection. Then this border can provide a tailor-made semantic concept representation for a document coming from the same domain. Another advantage of ACR is that it is applicable in both classification task where the groups are given in the training document set, and clustering task where no group information is available. Two algorithms are proposed, namely, GBG-g and GBG-s, to generate the concept border. In the experiments, GBG-g performs better and is more stable than GBG-s.

6. REFERENCES

[1] L. Bing, B. Sun, S. Jiang, Y. Zhang, and W. Lam. Learning ontology resolution for document representation and its applications in text mining. Website, 2010. `http://www.cis.pku.edu.cn/faculty/system/zhangyan/papers/acr_2010.pdf`.

[2] C. C. Chang and C. J. Lin. *LIBSVM: a library for support vector machines*, 2001.

[3] A. Hotho, A. Maedche, and S. Staab. Ontology-based text document clustering. *Data Knowledge Engineering*, 16(4), 2002.

[4] A. Hotho, S. Staab, and G. Stumme. Wordnet improves text document clustering. In *Proceeding of the SIGIR 2003 Semantic Web Workshop*, pages 541–544, 2003.

[5] L. Jing, L. Zhou, M. K. Ng, and J. Z. Huang. Ontology-based distance measure for text clustering. In *Proceedings of the Text Mining Workshop, SIAM International Conference on Data Mining*. SIAM, 2006.

[6] C. Matuszek, J. Cabral, M. Witbrock, and J. Deoliveira. An introduction to the syntax and content of cyc. In *Proceedings of the AAAI*, pages 44–49, 2006.

[7] G. A. Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 38:39–41, 1995.

[8] D. Reforgiato Recupero. A new unsupervised method for document clustering by using WordNet lexical and conceptual relations. *Information Retrieval*, pages 563–579, 2007.

[9] I. Yoo, X. Hu, and I.-Y. Song. Integration of semantic-based bipartite graph representation and mutual refinement strategy for biomedical literature clustering. In *SIGKDD'06*, pages 791–796, New York, NY, USA, 2006. ACM.