Adaptive Concept 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
binglidong@gmail.com
{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
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 concepts for representing a document have a a fixed resolution and they are not adaptable to the characteristics of 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 the consideration of the characteristics of the 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 a static existing method in almost all cases.

1. INTRODUCTION

Traditionally, text document representation is usually based on the Bag of Words (BOW) approach, which represents the documents with features as weighted occurrence frequencies of individual words. This technique has several drawbacks: first, it breaks a phrase, say "text mining", into independent features. Second, it maps synonymous words into different features. Third, it merges a polysemous word's different meanings into a single feature. These drawbacks make the document similarity unable to be computed by BOW accurately. The methods that overcome these drawbacks can be categorized into two classes: latent semantic analysis (including LSA [3], PLSA [6] and LDA [1]) and ontology-based methods [8, 12]. In this paper, we focus on the later methodology.

Some carefully edited ontologies include WordNet [11], Cyc [10], Mesh [16] and etc. Previous empirical results have shown improvement in some cases utilizing these ontologies [8, 12, 16]. 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 resolutions for representing the documents belonging to different collections are 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 they should be unrelated. So an adaptive strategy can surely outperform the static one.

In this paper, the proposed Adaptive Concept Resolution (ACR) model can learn a concept border from an ontology taking into the consideration of the characteristics of the 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. As a result 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,

[*]Supported by NSFC under Grant No. 60673129 and 60773162, 863 Program under Grant No. 2007AA01Z154, and the 2008/2009 HP Labs Innovation Research Program.

[†]Currently study at The Chinese University of Hong Kong

[‡]Corresponding author

[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 of the edge.

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 a static existing method in almost all cases.

As far as we know, this work is the first one to investigate an adaptive strategy to utilize ontology on different document collections. The efficacy of our model for solving text mining problems is demonstrated in experiments. In the remainder of this paper, we first introduce the preliminary of ontology and the framework of ACR model in Section 2. Then two main components of ACR, namely, concept border generation and concept-based document representation, are discussed in Section 3 and Section 4 respectively. Some technique details and the time complexity of ACR are presented in Section 5. Then the experiment design and results are given in Section 6. The related works are reviewed in Section 7. Finally we conclude the paper.

2. PRELIMINARY AND OVERVIEW

2.1 Ontology Preliminary

Concepts are the basic components of an ontology. Each concept may refer to an abstract entity or a real entity. In each concept, several components are involved, such as a *synonym set*, *hyponymy* (is-a) and *holonymy* (part-of) relations with other concepts, a gloss, and etc. From the semantic point of view, hyponymy relation is an important one, which is also the relation considered in this paper. We give a formal definition of a concept:

DEFINITION 2.1. ***Concept:*** *A semantic concept π is a quadruple $\langle id, \Omega, \sigma, \Upsilon\rangle$, where id is its ID, Ω is the synonym set, σ denotes the gloss, and Υ is the set of its hyponym concepts. We refer to the items with $\pi.id$, $\pi.\Omega$, $\pi.\sigma$ and $\pi.\Upsilon$ respectively. Ω's element is denoted as ω, and a set of concepts is denoted as Π. If Υ is empty, π is a leaf concept.*

WordNet is a popular ontology, and it has been extensively adopted in text mining in the past decade or so. In this paper, we select WordNet2.1 as an instance of the ontology to illustrate the framework. In Figure 1, a fragment of WordNet is given. Take the concept "island" (*id*: 09316454) as an example. Then Ω is {island}, σ is "a land mass (smaller than a continent) that is surrounded by water", and some of its hyponym concepts are shown in the dashed box. Sometimes the first term in a synonym set is used to refer to the concept.

In WordNet, the concept "00001740" with synonymy set {entity} is the root concept. There are two kinds of "is a" relation, "Java" is a real instance of "island", while "islet" is a semantic instance. Some concepts may have more than one hypernym concepts, as exemplified by the concept "Wight" at the bottom right in Figure 1. We call this kind of concept an *ambiguous concept.* As a result, ontology's structure is a hierarchical directed acyclic graph. The depth $d(\pi)$ for the concept π in the graph is defined as follows:

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

Ambiguous terms are terms that may be included by more than one concepts' synonym sets. The term "Java" refers to an island in Figure 1, it can also refer to a kind of coffee or a programming language as shown in Figure 2.

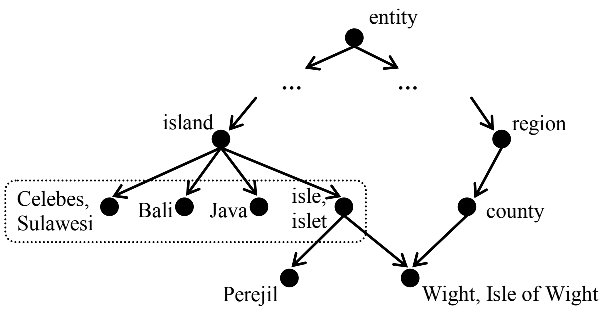

Figure 1: A fragment of WordNet structure. Each node is a concept, whose synonym set contains the terms attached to the node.

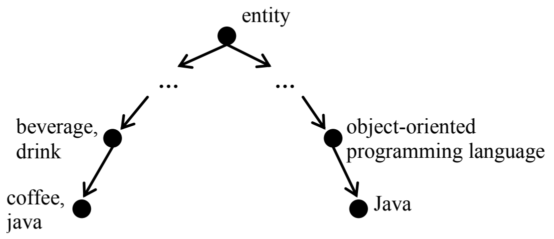

Figure 2: Ambiguous term "java" in WordNet.

2.2 Overview of ACR

Figure 3 depicts an overview of the Adaptive Concept Resolution model. There are two main parts in the framework indicated by a dashed box, namely, the learning part on the left hand side and the utilizing part on the right hand side. The learning part aims at generating a concept border based on an ontology and a training document collection. After that, the utilizing part utilizes this border to construct a concept-based representation for a new document.

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 $\mathcal{B}$. The border is composed of all the concepts (represented by empty circles) located on the dashed line. 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 $\pi.\Omega$. Then we get a derived concept, π^b, of π in the border. 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 as the training document collection, and each concept in $\mathcal{B}$ is treated as one dimension in the vector. If t is a term in the document, and contained by the synonym set of π^b, its weight will be accumulated onto the dimension of π^b. If t is contained by a concept above the border, it will be omitted.

In both parts, the concept extraction and matching component is involved to preprocess the documents. We present the details of this component as well as the concept-based document representation in Section 4.

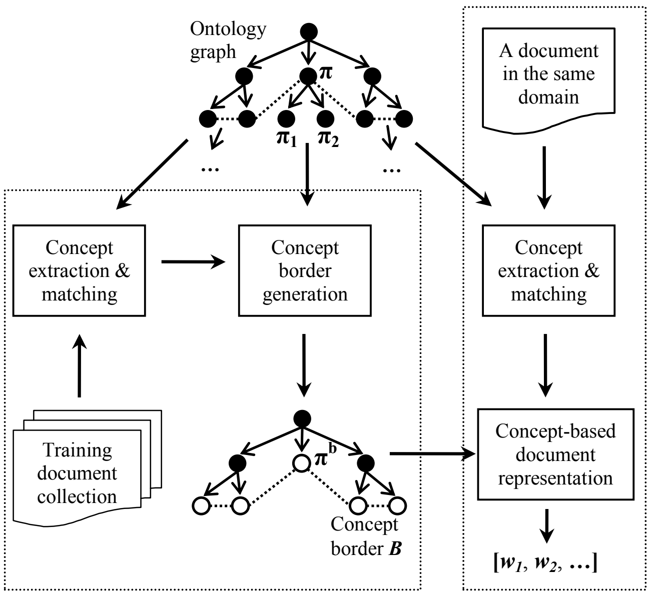
Ontology graph
π
π1
π2
A document in the same domain
Concept extraction & matching
Concept border generation
Concept extraction & matching
Training document collection
π^b
Concept border B
Concept-based document representation
[w1, w2, …]

Figure 3: Framework of our model. The learning part is in the left hand side dashed box, and the utilizing part in the right hand side dashed box.

3. CONCEPT BORDER GENERATION

In this section, we first introduce a definition of generalized entropy of a concept in Subsection 3.1. Then this entropy is used to define a *gain* function to guide the concept border generation process in Subsection 3.2.

3.1 Concept Entropy

Suppose D is a document set, and it can be partitioned into groups $D = \{u_1, u_2, \cdots\}$. If there exist clusters in the document set, each u_i represents a cluster. Otherwise, each u_i represents a single document.

Traditionally, document frequency (*df*) is used to indicate whether a term is commonly used in a document set. But *df* is a simple measure and it ignores the weight of the term in different documents. We propose an entropy based method to measure the popularity of terms in a document collection. The entropy of a term set T is calculated as in Equation 2:

$$entropy(T) = -\sum_{u_i \in D} p(u_i|T) \log p(u_i|T), \tag{2}$$

where $p(u_i|T)$ is the probability of u_i given T, and calculated as in Equation 3:

$$p(u_i|T) = \frac{\sum_{t_j \in T} w_{i,j}}{\sum_{u_k \in D, t_j \in T} w_{k,j}}, \tag{3}$$

where $w_{i,j}$ is the weight of t_j in u_i. When T just contains a single term t, Equation 2 becomes the term entropy. If t is frequently used in most of u_i, its entropy will be large, i.e., its uncertainty is large. Consequently t is not a good feature. If t is very frequently used in only one or several u_i, and only mentioned once or very few times in others, t is a good feature.

Each concept's synonym set contains one or several terms, and these terms have identical or very similar meanings. Thus, we measure the concept's entropy $Centropy(\pi)$ with the entropy of its synonym set:

$$Centropy(\pi) = entropy(\pi.\Omega). \tag{4}$$

If the concept entropy is large, it denotes the fact that the semantic meaning of this concept is popular in D.

π's descendants represent the specialized meanings of π. Considering all these concepts, including π, as a whole, we use the generalized entropy $Gentropy(\pi)$ to measure its popularity in the document collection:

$$Gentropy(\pi) = entropy(\Omega^g_\pi), \tag{5}$$

where $\Omega^g_\pi = \pi.\Omega \cup_{\pi' \in \Pi^d_\pi} \pi'.\Omega$, and $\Pi^d_\pi = \{\pi' | \pi \rightsquigarrow \pi'\}$ is the descendant set of π. Consequently generalized entropy indicates the popularity of an entire concept including its descendants. For a leaf concept π^l, we have $Centropy(\pi^l) = Gentropy(\pi^l)$.

We observe that a general concept is likely to be used more commonly than a special one, and it is formally stated as:

OBSERVATION 3.1. *If $\pi' \in \pi.\Upsilon$, we have $Gentropy(\pi) \geq Gentropy(\pi')$.*

The intuitive understanding is as follows: If $|\pi.\Upsilon| = 1$, comparing with $Gentropy(\pi')$, the calculation of $Gentropy(\pi)$ considers the general terms in $\pi.\Omega$, so its value should not be smaller than the former. If $|\pi.\Upsilon| > 1$, $Gentropy(\pi)$ also considers the siblings of π', which makes its value becomes larger.

3.2 Gain-based Border Generation (GBG)

To generate the concept border, the leaf concepts are merged into their hypernyms recursively. In this process, a gain value is used to measure whether the merging is profitable.

Refer to Figure 1, let's consider whether we should merge "Perejil" and "Wight" into "isle". Based on Observation 3.1, $Gentropy(isle)$ is equal or greater than the average of $Gentropy(Perejil)$ and $Gentropy(Wight)$. If we use one feature to represent these 3 concepts, some noises will be brought in. But at the same time the merging also provides more accurate similarity. For example, the similarity among the documents belonging to the same cluster and talking about "isle", "Perejil" and "Wight" respectively will be increased, and this is exactly the desired result. For measuring whether the trade-off is worthwhile, we define the gain function $gain(\pi)$ for a concept π:

$$gain(\pi) = \frac{\frac{1}{|\pi.\Upsilon|}\sum_{\pi' \in \pi.\Upsilon} Gentropy(\pi')}{Gentropy(\pi)}. \tag{6}$$

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. $gain(\pi) = 1$ means no noise brought in. So we always prefer the merging with larger $gain(\pi)$. A parameter θ is used as a profitable threshold. If $gain(\pi) \geq \theta$, the merging will be performed.

Until now, we discuss the problem in a bottom-up fashion, which is a generalization process. We can also consider it in a top-down fashion, which is a specialization process. And the merging operation becomes the splitting operation, where $gain(\pi)$ can still be used in the same way. We name these two methods as GBG-g and GBG-s respectively.

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 $flg = true$.

Algorithm 1 GBG-g

```
1: input: the HDAG $G$ of an ontology, threshold $\theta$
2: output: concept border $\mathcal{B}$
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 $\pi^l$ with $flg = false$
6:    get $\pi$ among $\pi^l$'s hypernyms, which is the deepest
7:    if $\pi$ has non-leaf hyponym concepts then
8:       set_border_flag($\pi$)
9:    else
10:      if $gain(\pi) \geq \theta$ then
11:         set $\pi.\Omega \leftarrow \pi.\Omega \cup_{\pi' \in \pi.\Upsilon} \pi'.\Omega$
12:         set $\pi.\Upsilon \leftarrow null$
13:      else
14:         set_border_flag($\pi$)
15:      end if
16:   end if
17: end while
18: set $\mathcal{B} = \{\pi | \pi' s\ flg\ is\ true\}$
1: proc set_border_flag($\pi$)
2: for all $\pi'$ in $\pi.\Upsilon$ do
3:    if $\pi'.\Upsilon = null$ then
4:       if $\pi'$ is ambiguous then
5:          delete $\pi'$ from $\pi.\Upsilon$
6:          reset the depth of $\pi'$
7:       else
8:          set $flg \leftarrow true$ for $\pi'$
9:       end if
10:   end if
11: end for
```

In the sub-procedure of *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 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*).

Algorithm 2 GBG-s

```
1: input: the HDAG $G$ of an ontology, threshold $\theta$
2: output: concept border $\mathcal{B}$
3: top_down(root)
1: proc top_down($\pi$)
2: if $gain(\pi) \geq \theta$ then
3:    set $\pi.\Omega \leftarrow \pi.\Omega \cup_{\pi' \in \{\pi' | \pi \rightsquigarrow \pi'\}} \pi'.\Omega$
4:    put $\pi$ into $\mathcal{B}$
5:    remove all concepts in $\{\pi' | \pi \rightsquigarrow \pi'\}$ from $G$
6: else
7:    for all $\pi'$ in $\pi.\Upsilon$ do
8:       top_down($\pi'$)
9:    end for
10: end if
```

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.

4. CONCEPT-BASED DOCUMENT REPRESENTATION

After the concept border is generated, it can be used to represent a new document as a concept vector, and each concept in $\mathcal{B}$ is one dimension in the vector. In this section, the details of the concept extraction and matching component is described first. Then the method of representing a document in a weighted concept vector is presented.

4.1 Concept Extraction

As seen in the previous examples, some concepts are represented by phrases, such as "Isle of Wight" and "object-oriented programming language". Hence it is necessary to extract such concepts from the documents. Noun phrase chunking operation may help, but it is too time consuming.

We use the term set contained by the ontology as a dictionary, and perform a forward maximum cutting to extract the concepts from the documents. First, we detect sentence boundaries in a document, and get a set of sentences denoted as $\{S_1, S_2, \ldots\}$. Let $(\varpi_1^i, \varpi_2^i, \ldots)$ denote a sequence of tokens in the sentence S_i. Then the segment of the first l tokens $(\varpi_1^i, \ldots, \varpi_l^i)$ is treated as a candidate concept, and retrieved in the ontology dictionary. If it fails, the token at the end of the segment is omitted, thus the remaining is $(\varpi_1^i, \ldots, \varpi_{l-1}^i)$, and then the above retrieval is performed again. When a certain segment $(\varpi_1^i, \ldots, \varpi_k^i)$ is found in the dictionary, it will be treated as a concept in the document. Then we go on to consider the next segment $(\varpi_{k+1}^i, \ldots, \varpi_{k+l}^i)$. If the retrieval of (ϖ_1^i) still fails, we go on to consider $(\varpi_2^i, \ldots, \varpi_{1+l}^i)$. By this procedure, we can get all concepts contained in the document. l is a predefined maximum cutting length. We use "word" refer to a single word, and "term" refer to both a single word and a multi-word phase in this paper.

4.2 Concept Matching

Now suppose the ambiguous term "Java" appears in two documents, "Object-oriented programming language" and "The top 10 populous islands in the world". The above concept extraction method can only tell us that “Java” is a concept in both documents, but cannot tell us which one is its matching concept in the ontology, a programming language or an island. To match “Java” to a correct concept, we consider its context in the document, i.e. the surrounding sentences of “Java”, denoted as $\mathcal{C}$. First locate the concepts that contain “Java”, denoted as Π. Then for each π in Π, we calculate the matching score between $\mathcal{C}$ and π with $match(\mathcal{C}, \pi)$:

$$match(\mathcal{C}, \pi) = \alpha sim(\mathcal{C}, \pi) + (1 - \alpha) sim(\mathcal{C}, \Pi^c_\pi), \quad (7)$$

where $\Pi^c_\pi = \{\pi' | \pi \in \pi'.\Upsilon\} \cup \{\pi' | \pi \rightsquigarrow_n \pi'\}$ is the context of π ($\pi \rightsquigarrow_n \pi'$ means π' is a descendant of π in n hops), and α decides the weight distribution between the two parts on the right hand side. In the experiment, we use 0.5 for α, and 2 for n. A variant of Dice's coefficient [4] is used to calculate the similarity between two text fragments str_1 and str_2:

$$dice(str_1, str_2) = \frac{2|N_1 \cap N_2|}{|N_1| + |N_2|}, \quad (8)$$

where N_i is the multiset (or bag) [14] of the nouns in str_i. For example, $dice$({a,b}, {a,a}) is (2 * 1) / 4 = 0.5, and $dice$({a,a}, {a,a}) is (2 * 2) / 4 = 1. The similarity between a text fragment str and π is defined as:

$$sim(str, \pi) = dice(str, Str(\pi)), \quad (9)$$

where $Str(\pi)$ denotes concatenating $\pi.\sigma$ with each term in $\pi.\Omega$ to get a new string. We use Equation 10 to compute $sim(str, \Pi)$:

$$sim(str, \Pi) = \frac{\sum_{\pi \in \Pi} sim(str, \pi)}{|\Pi|}. \quad (10)$$

Finally, according to Equation 11:

$$\pi^m = \arg\max_{\pi \in \Pi} \; match(\mathcal{C}, \pi), \quad (11)$$

the concept π^m with the maximum matching score with $\mathcal{C}$ is selected as the correct matching concept.

4.3 Weighted Concept-based Document Representation

The border $\mathcal{B}$ generated by the GBG algorithms is used to represent a document. 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 follows:

$$cf_{i,j} = \frac{f_{i,j}}{\sum_{\pi_k \in d_j} f_{k,j}}, \quad (12)$$

$$idf_i = \log \frac{|D|}{1 + |\{d | \pi_i \in d\}|}, \quad (13)$$

$$cfidf_{i,j} = cf_{i,j} \times idf_i, \quad (14)$$

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.

5. TECHNIQUE DETAILS AND TIME COMPLEXITY

5.1 Virtual Concept

We find that ontology's structure is quite unbalanced. In the graph on left hand side of Figure 4, although c, d and e have a common hypernym r, c has more descendants than d and e. Reviewing the structure shown in Figure 1, the semantic meanings between “Bali” and “Java” are closer than that between “Bali” and “isle”. If the degree of the unbalance becomes severe, as shown in Figure 4, this kind of difference will increase dramatically. Hence combining c, d and e together will surely cause a lot of noise. However, d and e are very likely to have similar meanings because both of them are leaves. We introduce a *virtual concept* v to provide d and e a chance to be combined, and meanwhile make c excluded, as shown in the graph on the right hand side of Figure 4. More formally, if the concept π contains more than one leaf concepts and at least one non-leaf concept, we introduce a virtual concept v for π with the following operations:

1. $v.\Upsilon \leftarrow \{\pi' | \pi' \in \pi.\Upsilon \; and \; \pi'.\Upsilon = null\}$,
2. $\pi.\Upsilon \leftarrow \pi.\Upsilon - v.\Upsilon$,
3. $\pi.\Upsilon \leftarrow \pi.\Upsilon \cup \{v\}$.

We perform this processing before the algorithms GBG-g and GBG-s.

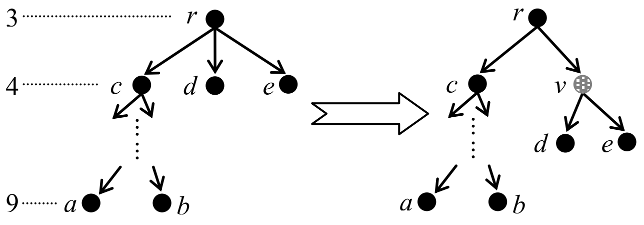

Figure 4: Unbalanced structure and virtual concept. The depth of the concept is given at the beginning dashed line.

5.2 Gain Function Calculation Technique

We first analyze the intrinsic structure of $gain(\pi)$, then propose an efficient algorithm to calculate the gain value for each π in an ontology.

Suppose idx is an inverted index of the ontology concepts on a document set $D = \{u_1, u_2, \cdots\}$. The index record for π is $\pi \to \{\langle w^\pi_1, w^\pi_2, \cdots \rangle, w^\pi\}$, where $w^\pi_i = \sum_{t_j \in \pi.\Omega} w_{i,j}$ indicates the weight of π in u_i, and $w^\pi = \sum_{u_k \in D} w^\pi_k$ is the accumulated weight of π in D. Now suppose π has only two leaf hyponyms, π_1 and π_2. The weight of Ω^g_π in u_i, denoted as $w_i^{\Omega^g_\pi}$, is additively separable, $w_i^{\Omega^g_\pi} = \sum_{t_j \in \Omega^g_\pi} w_{i,j} = w^{\pi_1}_i + w^{\pi_2}_i + w^\pi_i$. This property can be generalized to the concepts with any depth, and the recursive calculation formula is:

$$w_i^{\Omega^g_\pi} = \begin{cases} w^\pi_i & if \; \pi.\Upsilon = null, \\ w^\pi_i + \sum_{\pi' \in \pi.\Upsilon} (w_i^{\Omega^g_{\pi'}}) & otherwise. \end{cases} \quad (15)$$

Then the probability $p(u_i | \Omega^g_\pi)$ can be calculated by Equation 16.

$$p(u_i | \Omega^g_\pi) = \frac{w_i^{\Omega^g_\pi}}{\sum_{u_k \in D} w_k^{\Omega^g_\pi}}. \quad (16)$$

Based on Equation 2 and 5, $Gentropy(\pi)$ can be calculated.

The calculation of $gain$ for each π is summarized in Algorithm 3. In a bottom-up fashion (line 3), each $w_i^{\Omega^g_\pi}$ for π is calculated recursively, and saved for reuse in the future (line 6). Then $Gentropy(\pi)$ is calculated and saved in lines 7 and 8. Finally $gain(\pi)$ is calculated in line 10.

Algorithm 3 Gain calculation

```
1: **input**: An ontology graph $G$, $idx$ on a document set $D$
2: **output**: $gain$ value for each $\pi$
3: **for** $depth$ from $maxDepth$ to 0 **do**
4:     **for all** $\pi$ with $d(\pi) = depth$ **do**
5:         retrieve $\pi \to \{\langle w_1^\pi, w_2^\pi, \cdots \rangle, w^\pi\}$ from $idx$
6:         calculate and save $w_i^{\Omega^g_\pi}$ for each $u_i$ (Equation 15)
7:         calculate $p(u_i|\Omega^g_\pi)$ for each $u_i$ (Equation 16)
8:         calculate and save $Gentropy(\pi)$
9:         **if** $\pi.\Upsilon \neq null$ **then**
10:            calculate $gain(\pi)$ according to Equation 6
11:        **end if**
12:    **end for**
13: **end for**
```

5.3 Time Complexity

The overall time complexity of ACR model should include the time consuming of text preprocessing, concept extraction and matching, inverted indexing and the GBG algorithms. The first three parts are well investigated in text mining and information retrieval. So we only analyze the complexity of the GBG algorithms.

The most time consuming operation in the GBG algorithms is the gain value calculation for each concept. Now suppose each document in D is treated as an individual u_i, the time complexity of calculating $w_i^{\Omega^g_\pi}$ for all u_i is $O(|D| * |\pi.\Upsilon|)$. Then the calculation of $p(u_i|\Omega^g_\pi)$ for all u_i takes $O(|D|)$. $Gentropy(\pi)$ calculation also takes $O(|D|)$. $gain(\pi)$ calculation takes $O(|\pi.\Upsilon|)$. Thus, the time complexity for calculating one concept's gain value is $O(|D| * |\pi.\Upsilon|)$. Let Π denote the concept set of an ontology, and $avgDeg$ denote the average size of Υ in Π, the overall time complexity of the GBG algorithm is $O(|D| * avgDeg * |\Pi|)$.

6. EXPERIMENTS

We conducted a comparative experiment evaluating the performance of our framework. A previous method, known as the "only" strategy, in [8] is also implemented for conducting the comparison. In this strategy, each concept's synonym set is used as one dimension in the document representation vector, and its weight is decided by the terms in the synonym set. The strategy is called "Hotho" in this paper. Both our framework and "Hotho" are methods for providing a better representation for a document. We perform experiments on two text mining tasks, namely, document clustering and document classification.

6.1 Data Set

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). 20 Newsgroups and TREC data are two ordinary document sets; OHSUMED [5] is a professional medical science data; ODP page set contains the Web pages crawled from five ODP categories: Arts, Business, Computers, Health, Sports. In OHSUMED, there are 106 queries which have manual labeled results. We use each query as a pre-defined cluster including the documents which are labeled as "definitely relevant". The documents definitely relevant to more than one queries are eliminated. Finally, we get 101 clusters with 1,870 documents. The details of each data set are given in Table 1.

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

6.2 Document Clustering

In document clustering experiments, we use K-Means algorithm to perform the clustering, and it is executed three times for each data set to get an average result. The entire document collection is used as the input information of the $gain$ value calculation in GBG algorithms. No cluster information is used in the calculation, hence, each u_i mentioned in Section 3 refers to an individual document.

6.2.1 *Evaluation Criteria*

The purity measure is employed to evaluate the clustering performance. Let $G = \{g_1, g_2, \cdots\}$ denote the cluster set generated by K-Means, and $C = \{c_1, c_2, \cdots\}$ denote the pre-defined clusters. To compute the purity, each g_k is assigned to the pre-defined cluster c_i which is the most frequent in g_k, and then the purity is measured by counting the number of correctly assigned documents and divided by $|D|$. Formally:

$$Pur(C, G) = \frac{1}{|D|} \sum_k \max_i |g_k \cap c_i|, \tag{17}$$

6.2.2 *Results*

The clustering result is given in Table 2. Both of our methods can outperform the existing method Hotho. Considering NG20 and TREC data set, the improvements are more significant, about 5% 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%.

Table 2: Clustering performance comparison.

	NG20	TREC	ODP	MED
Hotho	.752	.449	.783	.711
GBG-g	.808	.516	.825	.795
GBG-s	.807	.536	.830	.731

6.2.3 *Parameter Sensitivity Analysis*

The effect of θ in the clustering is shown in Table 3. The performance of GBG-g algorithm is not sensitive to θ, and it can outperform the existing method Hotho under any θ value. 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 a better result than a smaller value in general. 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.

Based on the discussions above, in the comparative result depicted in Table 2, the value of θ used in GBG-g is 0.7, and in GBG-s is 0.9.

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

6.3 Document Classification

In document classification experiments, LibSVM [2] with linear kernel is employed to conduct the classification experiment and 5-fold cross-validation is adopted. Note that the *gain* calculation only needs the training set. Because there exist many small clusters, with less than 5 documents, in MED, we do not use this data set in the classification experiment.

6.3.1 Evaluation Criteria

The overall F-measure score of the classification result can be computed by two metrics, namely, micro-average and macro-average [15].

In macro-average, the precision and recall for each category c_i are calculated first, denoted as P_i and R_i. Then F-measure for each category c_i is calculated as:

$$F_i = \frac{2P_iR_i}{P_i + R_i},$$

The macro-averaged F-measure is the average of F-measure for each category:

$$F^{ma} = \frac{\sum_i F_i}{|C|}.$$

In micro-averaging, F-measure is computed globally over all category decisions. the global precision and recall are calculated as:

$$P = \frac{\sum_i TP_i}{\sum_i (TP_i + FP_i)},$$

and

$$R = \frac{1}{|C|} \sum_i \frac{TP_i}{(TP_i + FN_i)},$$

where TP_i, FP_i and FN_i are true positive, false positive and false negative numbers for category c_i. Micro-averaged F-measure is defined as:

$$F^{mi} = \frac{2PR}{P + R}.$$

6.3.2 Results

The classification result is given in Table 4. Except for the F^{ma} of GBG-s on NG20, our methods dominate all other cases. 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.

Table 4: Classification performance comparison

	NG20		TREC		ODP	
	F^{mi}	F^{ma}	F^{mi}	F^{ma}	F^{mi}	F^{ma}
Hotho	.904	.793	.631	.580	.783	.653
GBG-g	.934	.818	.696	.643	.809	.677
GBG-s	.907	.780	.692	.635	.825	.689

6.3.3 Parameter Sensitivity Analysis

The effect of θ in the classification is shown in Table 5. 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 cluster granularity of calculating the gain value, a larger θ can prevent the concepts which can 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 satisfy and be merged. Thus, the value of θ used in both GBG-g and GBG-s is 1 in the classification experiment.

Interestingly, we find that on NG20, GBG-g can perform slightly better with both small and large θ values than with the medium values. It may be because after the concepts are significantly 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.

7. RELATED WORK

As an important expert-edited ontology, WordNet has been used to improve the performance of clustering and classification. Hotho et al. [7, 8] show that incorporating the

Table 5: 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

synonym set and the hypernym as background knowledge into the document representation can improve the clustering results. They also apply the WordNet knowledge in a heterarchy form to generate different clustering views on a set of documents. Jing et al. [9] construct 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 [12], 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 group words based on the ontology they are related to. The authors in [13] successfully integrate the WordNet resource for document classification. They show improved classification results with respect to the Rocchio and Widrow-Hoff algorithms.

A significant difference between our ACR model and the methods mentioned above is that we adopt a learning process to determine the dimensions in the new representation for the documents, which gives our method more adaptability in dealing with different document collections.

8. CONCLUSIONS

In this paper, we propose an Adaptive Concept Resolution model to adaptively learn a concept border from an ontology taking into the consideration of the characteristics of the 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.

9. REFERENCES

[1] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

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

[3] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407, 1990.

[4] L. R. Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, July 1945.

[5] W. Hersh, C. Buckley, T. J. Leone, and D. Hickam. Ohsumed: an interactive retrieval evaluation and new large test collection for research. In *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 192–201, New York, NY, USA, 1994.

[6] T. Hofmann. Probabilistic latent semantic indexing. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57, New York, NY, USA, 1999. ACM.

[7] A. Hotho, A. Maedche, and S. Staab. Ontology-based text document clustering. *Data Knowledge Engineering*, 16(4):48–54, 2002.

[8] 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.

[9] 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.

[10] C. Matuszek, J. Cabral, M. Witbrock, and J. Deoliveira. An introduction to the syntax and content of cyc. In *Proceedings of the 2006 AAAI Spring Symposium on Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering*, pages 44–49, 2006.

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

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

[13] S. Scott and S. Matwin. Text classification using wordnet hypernyms. In *Workshop on usage of WordNet in NLP Systems (COLING-ACL '98)*, pages 45–51, August 1998.

[14] A. Syropoulos. Mathematics of multisets. In *WMP '00: Proceedings of the Workshop on Multiset Processing*, pages 347–358, London, UK, 2001. Springer-Verlag.

[15] Y. Yang and X. Liu. A re-examination of text categorization methods. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 42–49, New York, NY, USA, 1999. ACM.

[16] I. Yoo, X. Hu, and I.-Y. Song. Integration of semantic-based bipartite graph representation and mutual refinement strategy for biomedical literature clustering. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 791–796, New York, NY, USA, 2006. ACM.