Employing Topic Models for Pattern-based Semantic Class Discovery

Huibin Zhang[1*] **Mingjie Zhu**[2*] **Shuming Shi**[3] **Ji-Rong Wen**[3]
[1]Nankai University
[2]University of Science and Technology of China
[3]Microsoft Research Asia
{v-huibzh, v-mingjz, shumings, jrwen}@microsoft.com

Abstract

A semantic class is a collection of items (words or phrases) which have semantically peer or sibling relationship. This paper studies the employment of topic models to automatically construct semantic classes, taking as the source data a collection of *raw semantic classes* (RASCs), which were extracted by applying predefined patterns to web pages. The primary requirement (and challenge) here is dealing with multi-membership: An item may belong to multiple semantic classes; and we need to discover as many as possible the different semantic classes the item belongs to. To adopt topic models, we treat RASCs as "documents", items as "words", and the final semantic classes as "topics". Appropriate preprocessing and postprocessing are performed to improve results quality, to reduce computation cost, and to tackle the fixed-*k* constraint of a typical topic model. Experiments conducted on 40 million web pages show that our approach could yield better results than alternative approaches.

1 Introduction

Semantic class construction (Lin and Pantel, 2001; Pantel and Lin, 2002; Pasca, 2004; Shinzato and Torisawa, 2005; Ohshima et al., 2006) tries to discover the peer or sibling relationship among terms or phrases by organizing them into semantic classes. For example, {red, white, black…} is a semantic class consisting of color instances. A popular way for semantic class discovery is pattern-based approach, where predefined patterns (Table 1) are applied to a collection of web pages or an online web search engine to produce some *raw semantic classes* (abbreviated as RASCs, Table 2). RASCs cannot be treated as the ultimate semantic classes, because they are typically noisy and incomplete, as shown in Table 2. In addition, the information of one real semantic class may be distributed in lots of RASCs (R_2 and R_3 in Table 2).

Type	Pattern
SENT	NP {, NP}*{,} (and|or) {other} NP
TAG	<UL> <LI>item</LI> … <LI>item</LI> </UL>
TAG	<SELECT> <OPTION>item…<OPTION>item </SELECT>

* SENT: Sentence structure patterns; TAG: HTML Tag patterns

Table 1. Sample patterns

R_1: {gold, silver, copper, coal, iron, uranium}
R_2: {red, yellow, *color*, gold, silver, copper}
R_3: {red, green, blue, yellow}
R_4: {HTML, Text, PDF, MS Word, *Any file type*}
R_5: {*Today*, *Tomorrow*, Wednesday, Thursday, Friday, Saturday, Sunday}
R_6: {*Bush*, Iraq, *Photos*, USA, *War*}

Table 2. Sample raw semantic classes (RASCs)

This paper aims to discover high-quality semantic classes from a large collection of noisy RASCs. The primary requirement (and challenge) here is to deal with *multi-membership*, i.e., one item may belong to multiple different semantic classes. For example, the term "Lincoln" can simultaneously represent a person, a place, or a car brand name. Multi-membership is more popular than at a first glance, because quite a lot of English common words have also been borrowed as company names, places, or product names. For a given item (as a query) which belongs to multiple semantic classes, we intend to return the semantic classes separately, rather than mixing all their items together.

Existing pattern-based approaches only provide very limited support to multi-membership. For example, RASCs with the same labels (or hypernyms) are merged in (Pasca, 2004) to gen-

[*] This work was performed when the authors were interns at Microsoft Research Asia

erate the ultimate semantic classes. This is problematic, because RASCs may not have (accurate) hypernyms with them.

In this paper, we propose to use topic models to address the problem. In some topic models, a document is modeled as a mixture of hidden topics. The words of a document are generated according to the word distribution over the topics corresponding to the document (see Section 2 for details). Given a corpus, the latent topics can be obtained by a parameter estimation procedure. Topic modeling provides a formal and convenient way of dealing with multi-membership, which is our primary motivation of adopting topic models here. To employ topic models, we treat RASCs as "documents", items as "words", and the final semantic classes as "topics".

There are, however, several challenges in applying topic models to our problem. To begin with, the computation is intractable for processing a large collection of RASCs (our dataset for experiments contains 2.7 million unique RASCs extracted from 40 million web pages). Second, typical topic models require the number of topics (k) to be given. But it lacks an easy way of acquiring the ideal number of semantic classes from the source RASC collection. For the first challenge, we choose to apply topic models to the RASCs containing an item q, rather than the whole RASC collection. In addition, we also perform some preprocessing operations in which some items are discarded to further improve efficiency. For the second challenge, considering that most items only belong to a small number of semantic classes, we fix (for all items q) a topic number which is slightly larger than the number of classes an item could belong to. And then a postprocessing operation is performed to merge the results of topic models to generate the ultimate semantic classes.

Experimental results show that, our topic model approach is able to generate higher-quality semantic classes than popular clustering algorithms (e.g., K-Medoids and DBSCAN).

We make two contributions in the paper: On one hand, we find an effective way of constructing high-quality semantic classes in the pattern-based category which deals with multi-membership. On the other hand, we demonstrate, for the first time, that topic modeling can be utilized to help mining the *peer* relationship among words. In contrast, the general *related* relationship between words is extracted in existing topic modeling applications. Thus we expand the application scope of topic modeling.

2 Topic Models

In this section we briefly introduce the two widely used topic models which are adopted in our paper. Both of them model a document as a mixture of hidden topics. The words of every document are assumed to be generated via a generative probability process. The parameters of the model are estimated from a training process over a given corpus, by maximizing the likelihood of generating the corpus. Then the model can be utilized to inference a new document.

pLSI: The probabilistic Latent Semantic Indexing Model (pLSI) was introduced in Hofmann (1999), arose from Latent Semantic Indexing (Deerwester et al., 1990). The following process illustrates how to generate a document d in pLSI:

1. Pick a topic mixture distribution $p(\cdot|d)$.
2. For each word w_i in d
   a. Pick a latent topic z with the probability $p(z|d)$ for w_i
   b. Generate w_i with probability $p(w_i|z)$

So with k latent topics, the likelihood of generating a document d is

$$p(d) = \prod_i \sum_z p(w_i|z)p(z|d) \qquad (2.1)$$

LDA (Blei et al., 2003): In LDA, the topic mixture is drawn from a conjugate Dirichlet prior that remains the same for all documents (Figure 1). The generative process for each document in the corpus is,

1. Choose document length N from a Poisson distribution *Poisson(ξ)*.
2. Choose θ from a Dirichlet distribution with parameter α.
3. For each of the N words w_i.
   a. Choose a topic z from a Multinomial distribution with parameter θ.
   b. Pick a word w_i from $p(w_i|z,\beta)$.

So the likelihood of generating a document is

$$p(d) = \int_\theta p(\theta|\alpha) \prod_i \sum_z p(z|\theta)p(w_i|z,\beta)\, d\theta \qquad (2.2)$$

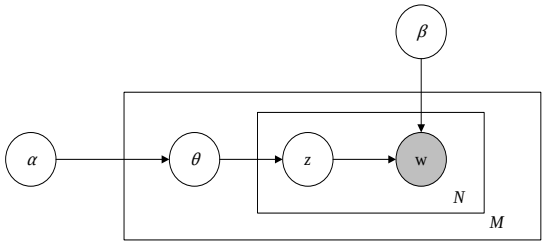

Figure 1. Graphical model representation of LDA, from Blei et al. (2003)

3 Our Approach

The source data of our approach is a collection (denoted as C_R) of RASCs extracted via applying patterns to a large collection of web pages. Given an item as an input query, the output of our approach is one or multiple semantic classes for the item. To be applicable in real-world dataset, our approach needs to be able to process at least millions of RASCs.

3.1 Main Idea

As reviewed in Section 2, topic modeling provides a formal and convenient way of grouping *documents* and *words* to *topics*. In order to apply topic models to our problem, we map RASCs to *documents*, items to *words*, and treat the output *topics* yielded from topic modeling as our semantic classes (Table 3). The motivation of utilizing topic modeling to solve our problem and building the above mapping comes from the following observations.

1) In our problem, one item may belong to multiple semantic classes; similarly in topic modeling, a word can appear in multiple topics.

2) We observe from our source data that some RASCs are comprised of items in multiple semantic classes. And at the same time, one document could be related to multiple topics in some topic models (e.g., pLSI and LDA).

Topic modeling	Semantic class construction
word	item (word or phrase)
document	RASC
topic	semantic class

Table 3. The mapping from the concepts in topic modeling to those in semantic class construction

Due to the above observations, we hope topic modeling can be employed to construct semantic classes from RASCs, just as it has been used in assigning documents and words to topics.

There are some critical challenges and issues which should be properly addressed when topic models are adopted here.

Efficiency: Our RASC collection C_R contains about 2.7 million unique RASCs and 26 million (1 million unique) items. Building topic models directly for such a large dataset may be computationally intractable. To overcome this challenge, we choose to *apply topic models to the RASCs containing a specific item* rather than the whole RASC collection. Please keep in mind that our goal in this paper is to construct the semantic classes for an item when the item is given as a query. For one item q, we denote $C_R(q)$ to be all the RASCs in C_R containing the item. We believe building a topic model over $C_R(q)$ is much more effective because it contains significantly fewer "documents", "words", and "topics". To further improve efficiency, we also perform preprocessing (refer to Section 3.4 for details) before building topic models for $C_R(q)$, where some low-frequency items are removed.

Determine the number of topics: Most topic models require the number of topics to be known beforehand[1]. However, it is not an easy task to automatically determine the exact number of semantic classes an item q should belong to. Actually the number may vary for different q. Our solution is to set (for all items q) the topic number to be a fixed value (k=5 in our experiments) which is slightly larger than the number of semantic classes most items could belong to. Then we perform postprocessing for the k topics to produce the final properly semantic classes.

In summary, our approach contains three phases (Figure 2). We build topic models for every $C_R(q)$, rather than the whole collection C_R. A preprocessing phase and a postprocessing phase are added before and after the topic modeling phase to improve efficiency and to overcome the fixed-k problem. The details of each phase are presented in the following subsections.

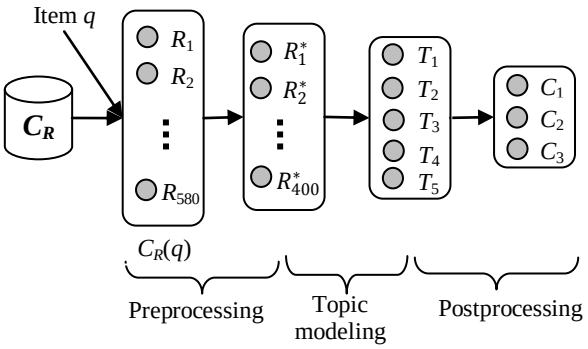

Figure 2. Main phases of our approach

3.2 Adopting Topic Models

For an item q, topic modeling is adopted to process the RASCs in $C_R(q)$ to generate k semantic classes. Here we use LDA as an example to

[1] Although there is study of non-parametric Bayesian models (Li et al., 2007) which need no prior knowledge of topic number, the computational complexity seems to exceed our efficiency requirement and we shall leave this to future work.

illustrate the process. The case of other generative topic models (e.g., pLSI) is very similar.

According to the assumption of LDA and our concept mapping in Table 3, a RASC ("document") is viewed as a mixture of hidden semantic classes ("topics"). The generative process for a RASC R in the "corpus" $C_R(q)$ is as follows,

1) Choose a RASC size (i.e., the number of items in R): $N_R \sim \text{Poisson}(\xi)$.
2) Choose a k-dimensional vector θ_R from a Dirichlet distribution with parameter α.
3) For each of the N_R items a_n:
   a) Pick a semantic class z_n from a multinomial distribution with parameter θ_R.
   b) Pick an item a_n from $p(a_n|z_n,\beta)$, where the item probabilities are parameterized by the matrix β.

There are three parameters in the model: ξ (a scalar), α (a k-dimensional vector), and β (a $k \times V$ matrix where V is the number of distinct items in $C_R(q)$). The parameter values can be obtained from a training (or called parameter estimation) process over $C_R(q)$, by maximizing the likelihood of generating the corpus. Once β is determined, we are able to compute $p(a|z,\beta)$, the probability of item a belonging to semantic class z. Therefore we can determine the members of a semantic class z by selecting those items with high $p(a|z,\beta)$ values.

The number of topics k is assumed known and fixed in LDA. As has been discussed in Section 3.1, we set a constant k value for all different $C_R(q)$. And we rely on the postprocessing phase to merge the semantic classes produced by the topic model to generate the ultimate semantic classes.

When topic modeling is used in document classification, an inference procedure is required to determine the topics for a new document. Please note that inference is not needed in our problem.

One natural question here is: Considering that in most topic modeling applications, the words within a resultant topic are typically semantically *related* but may not be in peer relationship, then what is the intuition that the resultant topics here are semantic classes rather than lists of *generally related* words? The magic lies in the "documents" we used in employing topic models. Words co-occurred in real documents tend to be semantically *related*; while items co-occurred in RASCs tend to be peers. Experimental results show that most items in the same output semantic class have peer relationship.

It might be noteworthy to mention the exchangeability or "bag-of-words" assumption in most topic models. Although the order of words in a document may be important, standard topic models neglect the order for simplicity and other reasons[2]. The order of items in a RASC is clearly much weaker than the order of words in an ordinary document. In some sense, topic models are more suitable to be used here than in processing an ordinary document corpus.

3.3 Preprocessing and Postprocessing

Preprocessing is applied to $C_R(q)$ before we build topic models for it. In this phase, we discard from all RASCs the items with frequency (i.e., the number of RASCs containing the item) less than a threshold h. A RASC itself is discarded from $C_R(q)$ if it contains less than two items after the item-removal operations. We choose to remove low-frequency items, because we found that low-frequency items are seldom important members of any semantic class for q. So the goal is to reduce the topic model training time (by reducing the training data) without sacrificing results quality too much. In the experiments section, we compare the approaches with and without preprocessing in terms of results quality and efficiency. Interestingly, experimental results show that, for some small threshold values, the results quality becomes *higher* after preprocessing is performed. We will give more discussions in Section 4.

In the postprocessing phase, the output semantic classes ("topics") of topic modeling are merged to generate the ultimate semantic classes. As indicated in Sections 3.1 and 3.2, we fix the number of topics (k=5) for different corpus $C_R(q)$ in employing topic models. For most items q, this is a larger value than the real number of semantic classes the item belongs to. As a result, one real semantic class may be divided into multiple topics. Therefore one core operation in this phase is to merge those topics into one semantic class. In addition, the items in each semantic class need to be properly ordered. Thus main operations include,

1) Merge semantic classes
2) Sort the items in each semantic class

Now we illustrate how to perform the operations.

Merge semantic classes: The merge process is performed by repeatedly calculating the simi-

[2] There are topic model extensions considering word order in documents, such as Griffiths et al. (2005).

larity between two semantic classes and merging the two ones with the highest similarity until the similarity is under a threshold. One simple and straightforward similarity measure is the Jaccard coefficient,

$$sim(C_1, C_2) = \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|} \quad (3.1)$$

where $C_1 \cap C_2$ and $C_1 \cup C_2$ are respectively the intersection and union of semantic classes C_1 and C_2. This formula might be over-simple, because the similarity between two different items is not exploited. So we propose the following measure,

$$sim(C_1, C_2) = \frac{\sum_{a \in C_1} \sum_{b \in C_2} sim(a, b)}{|C_1| \cdot |C_2|} \quad (3.2)$$

where $|C|$ is the number of items in semantic class C, and $sim(a,b)$ is the similarity between items a and b, which will be discussed shortly. In Section 4, we compare the performance of the above two formulas by experiments.

Sort items: We assign an importance score to every item in a semantic class and sort them according to the importance scores. Intuitively, an item should get a high rank if the average similarity between the item and the other items in the semantic class is high, and if it has high similarity to the query item q. Thus we calculate the importance of item a in a semantic class C as follows,

$$g(a|C) = \lambda \cdot sim(a,C) + (1-\lambda) \cdot sim(a,q) \quad (3.3)$$

where λ is a parameter in [0,1], $sim(a,q)$ is the similarity between a and the query item q, and $sim(a,C)$ is the similarity between a and C, calculated as,

$$sim(a, C) = \frac{\sum_{b \in C} sim(a, b)}{|C|} \quad (3.4)$$

Item similarity calculation: Formulas 3.2, 3.3, and 3.4 rely on the calculation of the similarity between two items.

One simple way of estimating item similarity is to count the number of RASCs containing both of them. We extend such an idea by distinguishing the reliability of different patterns and punishing term similarity contributions from the same site. The resultant similarity formula is,

$$sim(a, b) = \sum_{i=1}^{m} \log(1 + \sum_{j=1}^{k_i} w(P(C_{i,j}))) \quad (3.5)$$

where $C_{i,j}$ is a RASC containing both a and b, $P(C_{i,j})$ is the pattern via which the RASC is extracted, and $w(P)$ is the weight of pattern P. Assume all these RASCs belong to m sites with $C_{i,j}$ extracted from a page in site i, and k_i being the number of RASCs corresponding to site i. To determine the weight of every type of pattern, we randomly selected 50 RASCs for each pattern and labeled their quality. The weight of each kind of pattern is then determined by the average quality of all labeled RASCs corresponding to it.

The efficiency of postprocessing is not a problem, because the time cost of postprocessing is much less than that of the topic modeling phase.

3.4 Discussion

3.4.1 *Efficiency of processing popular items*

Our approach receives a query item q from users and returns the semantic classes containing the query. The maximal query processing time should not be larger than several seconds, because users would not like to wait more time. Although the average query processing time of our approach is much shorter than 1 second (see Table 4 in Section 4), it takes several minutes to process a popular item such as "Washington", because it is contained in a lot of RASCs. In order to reduce the maximal online processing time, our solution is offline processing popular items and storing the resultant semantic classes on disk. The time cost of offline processing is feasible, because we spent about 15 hours on a 4-core machine to complete the offline processing for *all* the items in our RASC collection.

3.4.2 *Alternative approaches*

One may be able to easily think of other approaches to address our problem. Here we discuss some alternative approaches which are treated as our baseline in experiments.

RASC clustering: Given a query item q, run a clustering algorithm over $C_R(q)$ and merge all RASCs in the same cluster as one semantic class. Formula 3.1 or 3.2 can be used to compute the similarity between RASCs in performing clustering. We try two clustering algorithms in experiments: K-Medoids and DBSCAN. Please note *k*-means cannot be utilized here because coordinates are not available for RASCs. One drawback of RASC clustering is that it cannot deal with the case of one RASC containing the items from multiple semantic classes.

Item clustering: By Formula 3.5, we are able to construct an item graph G_I to record the neighbors (in terms of similarity) of each item. Given a query item q, we first retrieve its neighbors from G_I, and then run a clustering algorithm over the neighbors. As in the case of RASC clustering, we try two clustering algorithms in experiments: K-Medoids and DBSCAN. The primary disadvantage of item clustering is that it cannot assign an item (except for the query item q) to

multiple semantic classes. As a result, when we input "gold" as the query, the item "silver" can only be assigned to one semantic class, although the term can simultaneously represents a color and a chemical element.

4 Experiments

4.1 Experimental Setup

Datasets: By using the Open Directory Project (ODP[3]) URLs as seeds, we crawled about 40 million English web pages in a breadth-first way. RASCs are extracted via applying a list of sentence structure patterns and HTML tag patterns (see Table 1 for some examples). Our RASC collection C_R contains about 2.7 million unique RASCs and 1 million distinct items.

Query set and labeling: We have volunteers to try Google Sets[4], record their queries being used, and select overall 55 queries to form our query set. For each query, the results of all approaches are mixed together and labeled by following two steps. In the first step, the *standard* (or *ideal*) *semantic classes* (SSCs) for the query are manually determined. For example, the ideal semantic classes for item "Georgia" may include Countries, and U.S. states. In the second step, each item is assigned a label of "Good", "Fair", or "Bad" with respect to each SSC. For example, "silver" is labeled "Good" with respect to "colors" and "chemical elements". We adopt metric MnDCG (Section 4.2) as our evaluation metric.

Approaches for comparison: We compare our approach with the alternative approaches discussed in Section 3.4.2.

LDA: Our approach with LDA as the topic model. The implementation of LDA is based on Blei's code of variational EM for LDA[5].

pLSI: Our approach with pLSI as the topic model. The implementation of pLSI is based on Schein, et al. (2002).

KMedoids-RASC: The *RASC clustering* approach illustrated in Section 3.4.2, with the K-Medoids clustering algorithm utilized.

DBSCAN-RASC: The *RASC clustering* approach with DBSCAN utilized.

KMedoids-Item: The *item clustering* approach with the K-Medoids utilized.

DBSCAN-Item: The *item clustering* approach with the DBSCAN clustering algorithm utilized.

K-Medoids clustering needs to predefine the cluster number *k*. We fix the *k* value for all different query item *q*, as has been done for the topic model approach. For fair comparison, the same postprocessing is made for all the approaches. And the same preprocessing is made for all the approaches except for the item clustering ones (to which the preprocessing is not applicable).

4.2 Evaluation Methodology

Each produced semantic class is an ordered list of items. A couple of metrics in the information retrieval (IR) community like Precision@10, MAP (mean average precision), and nDCG (normalized discounted cumulative gain) are available for evaluating a *single* ranked list of items per query (Croft et al., 2009). Among the metrics, nDCG (Jarvelin and Kekalainen, 2000) can handle our three-level judgments ("Good", "Fair", and "Bad", refer to Section 4.1),

$$nDCG@k = \frac{\sum_{i=1}^{k} G(i)/\log(i+1)}{\sum_{i=1}^{k} G^*(i)/\log(i+1)} \qquad (4.1)$$

where $G(i)$ is the gain value assigned to the i'th item, and $G^*(i)$ is the gain value assigned to the i'th item of an ideal (or perfect) ranking list.

Here we extend the IR metrics to the evaluation of *multiple* ordered lists per query. We use nDCG as the basic metric and extend it to MnDCG.

Assume labelers have determined m SSCs (SSC_1~SSC_m, refer to Section 4.1) for query q and the weight (or importance) of SSC_i is w_i. Assume n semantic classes are generated by an approach and n_1 of them have corresponding SSCs (i.e., no appropriate SSC can be found for the remaining n-n_1 semantic classes). We define the *MnDCG* score of an approach (with respect to query q) as,

$$MnDCG(q) = \frac{n_1}{n} \cdot \frac{\sum_{i=1}^{m} w_i \cdot Score(\mathrm{SSC}_i)}{\sum_{i=1}^{m} w_i} \qquad (4.2)$$

where

$$Score(SSC_i) = \begin{cases} 0 & if\ k_i = 0 \\ \frac{1}{k_i} \max_{j \in [1, k_i]} (nDCG(G_{i,j})) & if\ k_i \neq 0 \end{cases} \qquad (4.3)$$

In the above formula, nDCG($G_{i,j}$) is the nDCG score of semantic class $G_{i,j}$; and k_i denotes the number of semantic classes assigned to SSC_i. For a list of queries, the MnDCG score of an algorithm is the average of all scores for the queries.

The metric is designed to properly deal with the following cases,

[3] http://www.dmoz.org
[4] http://labs.google.com/sets
[5] http://www.cs.princeton.edu/~blei/lda-c/

i). One semantic class is wrongly split into multiple ones: Punished by dividing k_i in Formula 4.3;

ii). A semantic class is too noisy to be assigned to any SSC: Processed by the "n_1/n" in Formula 4.2;

iii). Fewer semantic classes (than the number of SSCs) are produced: Punished in Formula 4.3 by assigning a zero value.

iv). Wrongly merge multiple semantic classes into one: The nDCG score of the merged one will be small because it is computed with respect to only one single SSC.

The gain values of nDCG for the three relevance levels ("Bad", "Fair", and "Good") are respectively -1, 1, and 2 in experiments.

4.3 Experimental Results

4.3.1 *Overall performance comparison*

Figure 3 shows the performance comparison between the approaches listed in Section 4.1, using metrics MnDCG@*n* (*n*=1…10). Postprocessing is performed for all the approaches, where Formula 3.2 is adopted to compute the similarity between semantic classes. The results show that that the topic modeling approaches produce higher-quality semantic classes than the other approaches. It indicates that the topic mixture assumption of topic modeling can handle the multi-membership problem very well here. Among the alternative approaches, RASC clustering behaves better than item clustering. The reason might be that an item cannot belong to multiple clusters in the two item clustering approaches, while RASC clustering allows this. For the RASC clustering approaches, although one item has the chance to belong to different semantic classes, one RASC can only belong to one semantic class.

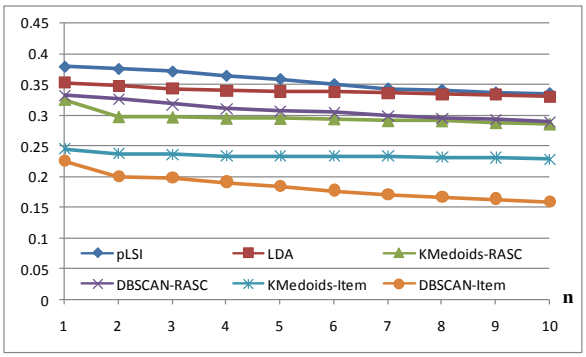

Figure 3. Quality comparison (MnDCG@n) among approaches (frequency threshold h = 4 in preprocessing; k = 5 in topic models)

4.3.2 *Preprocessing experiments*

Table 4 shows the average query processing time and results quality of the LDA approach, by varying frequency threshold *h*. Similar results are observed for the pLSI approach. In the table, *h*=1 means no preprocessing is performed. The average query processing time is calculated over all items in our dataset. As the threshold *h* increases, the processing time decreases as expected, because the input of topic modeling gets smaller. The second column lists the results quality (measured by MnDCG@10). Interestingly, we get the best results quality when *h*=4 (i.e., the items with frequency less than 4 are discarded). The reason may be that most low-frequency items are noisy ones. As a result, preprocessing can improve both results quality and processing efficiency; and *h*=4 seems a good choice in preprocessing for our dataset.

h	Avg. Query Proc. Time (seconds)	Quality (MnDCG@10)
1	0.414	0.281
2	0.375	0.294
3	0.320	0.322
4	0.268	**0.331**
5	0.232	0.328
6	0.210	0.315
7	0.197	0.315
8	0.184	0.313
9	0.173	0.288

Table 4. Time complexity and quality comparison among LDA approaches of different thresholds

4.3.3 *Postprocessing experiments*

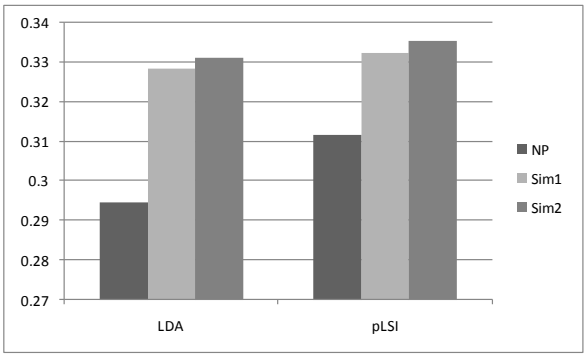

Figure 4. Results quality comparison among topic modeling approaches with and without postprocessing (metric: MnDCG@10)

The effect of postprocessing is shown in Figure 4. In the figure, NP means no postprocessing is performed. Sim1 and Sim2 respectively mean Formula 3.1 and Formula 3.2 are used in postprocessing as the similarity measure between

semantic classes. The same preprocessing (h=4) is performed in generating the data. It can be seen that postprocessing improves results quality. Sim2 achieves more performance improvement than Sim1, which demonstrates the effectiveness of the similarity measure in Formula 3.2.

4.3.4 *Sample results*

Table 5 shows the semantic classes generated by our LDA approach for some sample queries in which the bad classes or bad members are highlighted (to save space, 10 items are listed here, and the query itself is omitted in the resultant semantic classes).

Query	Semantic Classes
apple	C1: ibm, microsoft, sony, dell, toshiba, samsung, panasonic, canon, nec, sharp … C2: peach, strawberry, cherry, orange, banana, lemon, pineapple, raspberry, pear, grape …
gold	C1: silver, copper, platinum, zinc, lead, iron, nickel, tin, aluminum, manganese … C2: silver, red, black, white, blue, purple, orange, pink, brown, navy … C3: silver, platinum, earrings, diamonds, rings, bracelets, necklaces, pendants, jewelry, watches … *C4*: silver, home, money, business, metal, furniture, shoes, gypsum, hematite, fluorite …
lincoln	C1: ford, mazda, toyota, dodge, nissan, honda, bmw, chrysler, mitsubishi, audi … C2: bristol, manchester, birmingham, leeds, london, cardiff, nottingham, newcastle, sheffield, southampton … C3: jefferson, jackson, washington, madison, franklin, *sacramento*, *new york city*, monroe, *Louisville*, marion …
computer science	C1: chemistry, mathematics, physics, biology, psychology, education, history, music, business, economics …

Table 5. Semantic classes generated by our approach for some sample queries (topic model = LDA)

5 Related Work

Several categories of work are related to ours. The first category is about set expansion (i.e., retrieving one semantic class given one term or a couple of terms). Syntactic context information is used (Hindle, 1990; Ruge, 1992; Lin, 1998) to compute term similarities, based on which similar words to a particular word can directly be returned. Google sets is an online service which, given one to five items, predicts other items in the set. Ghahramani and Heller (2005) introduce a Bayesian Sets algorithm for set expansion. Set expansion is performed by feeding queries to web search engines in Wang and Cohen (2007) and Kozareva (2008). All of the above work only yields one semantic class for a given query. Second, there are pattern-based approaches in the literature which only do limited integration of RASCs (Shinzato and Torisawa, 2004; Shinzato and Torisawa, 2005; Pasca, 2004), as discussed in the introduction section. In Shi et al. (2008), an ad-hoc approach was proposed to discover the multiple semantic classes for one item. The third category is distributional similarity approaches which provide multi-membership support (Harris, 1985; Lin and Pantel, 2001; Pantel and Lin, 2002). Among them, the CBC algorithm (Pantel and Lin, 2002) addresses the multi-membership problem. But it relies on term vectors and centroids which are not available in pattern-based approaches. It is therefore not clear whether it can be borrowed to deal with multi-membership here.

Among the various applications of topic modeling, maybe the efforts of using topic model for Word Sense Disambiguation (WSD) are most relevant to our work. In Cai et al (2007), LDA is utilized to capture the global context information as the topic features for better performing the WSD task. In Boyd-Graber et al. (2007), Latent Dirichlet with WordNet (LDAWN) is developed for simultaneously disambiguating a corpus and learning the domains in which to consider each word. They do not generate semantic classes.

6 Conclusions

We presented an approach that employs topic modeling for semantic class construction. Given an item q, we first retrieve all RASCs containing the item to form a collection $C_R(q)$. Then we perform some preprocessing to $C_R(q)$ and build a topic model for it. Finally, the output semantic classes of topic modeling are post-processed to generate the final semantic classes. For the $C_R(q)$ which contains a lot of RASCs, we perform offline processing according to the above process and store the results on disk, in order to reduce the online query processing time.

We also proposed an evaluation methodology for measuring the quality of semantic classes. We show by experiments that our topic modeling approach outperforms the item clustering and RASC clustering approaches.

Acknowledgments

We wish to acknowledge help from Xiaokang Liu for mining RASCs from web pages, Changliang Wang and Zhongkai Fu for data process.

References

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022.

Bruce Croft, Donald Metzler, and Trevor Strohman. 2009. *Search Engines: Information Retrieval in Practice.* Addison Wesley.

Jordan Boyd-Graber, David Blei, and Xiaojin Zhu.2007. A topic model for word sense disambiguation. In *Proceedings EMNLP-CoNLL 2007*, pages 1024–1033, Prague, Czech Republic, June. Association for Computational Linguistics.

Jun Fu Cai, Wee Sun Lee, and Yee Whye Teh. 2007. NUS-ML: Improving word sense disambiguation using topic features. In *Proceedings of the International Workshop on Semantic Evaluations*, volume 4.

Scott Deerwester, Susan T. Dumais, GeorgeW. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407.

Zoubin Ghahramani and Katherine A. Heller. 2005. Bayesian Sets. In *Advances in Neural Information Processing Systems (NIPS05)*.

Thomas L. Griffiths, Mark Steyvers, David M. Blei,and Joshua B. Tenenbaum. 2005. Integrating topics and syntax. In *Advances in Neural Information Processing Systems* 17, pages 537–544. MIT Press

Zellig Harris. *Distributional Structure. The Philosophy of Linguistics*. New York: Oxford University Press. 1985.

Donald Hindle. 1990. Noun Classification from Predicate-Argument Structures. In *Proceedings of ACL90*, pages 268–275.

Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR99*, pages 50–57, New York, NY, USA. ACM.

Kalervo Jarvelin, and Jaana Kekalainen. 2000. IR Evaluation Methods for Retrieving Highly Relevant Documents. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR2000)*.

Zornitsa Kozareva, Ellen Riloff and Eduard Hovy. 2008. Semantic Class Learning from the Web with Hyponym Pattern Linkage Graphs, In *Proceedings of ACL-08.*

Wei Li, David M. Blei, and Andrew McCallum. Nonparametric Bayes Pachinko Allocation. In *Proceedings of Conference on Uncertainty in Artificial Intelligence (UAI)*, 2007.

Dekang Lin. 1998. Automatic Retrieval and Clustering of Similar Words. In *Proceedings of COLING-ACL98*, pages 768-774.

Dekang Lin and Patrick Pantel. 2001. Induction of Semantic Classes from Natural Language Text. In *Proceedings of SIGKDD01*, pages 317-322.

Hiroaki Ohshima, Satoshi Oyama, and Katsumi Tanaka. 2006. Searching coordinate terms with their context from the web. In *WISE*06, pages 40–47.

Patrick Pantel and Dekang Lin. 2002. Discovering Word Senses from Text. In *Proceedings of SIGKDD02*.

Marius Pasca. 2004. Acquisition of Categorized Named Entities for Web Search. In *Proc. of 2004 CIKM*.

Gerda Ruge. 1992. Experiments on Linguistically-Based Term Associations. In *Information Processing & Management, 28(3)*, pages 317-32.

Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar and David M. Pennock. 2002. Methods and metrics for cold-start recommendations. In *Proceedings of SIGIR02*, pages 253-260.

Shuming Shi, Xiaokang Liu and Ji-Rong Wen. 2008. Pattern-based Semantic Class Discovery with Multi-Membership Support. In *CIKM2008*, pages 1453-1454.

Keiji Shinzato and Kentaro Torisawa. 2004. Acquiring Hyponymy Relations from Web Documents. In *HLT/NAACL04*, pages 73–80.

Keiji Shinzato and Kentaro Torisawa. 2005. A Simple WWW-based Method for Semantic Word Class Acquisition. In *RANLP05*.

Richard C. Wang and William W. Cohen. 2007. Langusage-Independent Set Expansion of Named Entities Using the Web. In *ICDM2007*.