Scalable Community Discovery on Textual Data with Relations[*]

Huajing Li[†], Zaiqing Nie[§], Wang-Chien Lee[†], C. Lee Giles[†,‡], Ji-Rong Wen[§]

[†]Department of Computer Science and Engineering
[‡]College of Information Sciences and Technology
The Pennsylvania State University
University Park, PA 16802, USA

[§]Web Search and Mining Group
Microsoft Research Asia
Beijing, 100080 China
{huali, wlee}@cse.psu.edu
{znie, jrwen}@microsoft.com
giles@ist.psu.edu

ABSTRACT

Every piece of textual data is generated as a method to convey its authors' opinion regarding specific topics. Authors deliberately organize their writings and create links, i.e., references, acknowledgments, for better expression. Thereafter, it is of interest to study texts as well as their relations to understand the underlying topics and communities. Although many efforts exist in the literature in data clustering and topic mining, they are not applicable to community discovery on large document corpus for several reasons. First, few of them consider both textual attributes as well as relations. Second, scalability remains a significant issue for large-scale datasets. Additionally, most algorithms rely on a set of initial parameters that are hard to be captured and tuned. Motivated by the aforementioned observations, a hierarchical community model is proposed in the paper which distinguishes community cores from affiliated members. We present our efforts to develop a scalable community discovery solution for large-scale document corpus. Our proposal tries to quickly identify potential cores as seeds of communities through relation analysis. To eliminate the influence of initial parameters, an innovative attribute-based core merge process is introduced so that the algorithm promises to return consistent communities regardless initial parameters. Experimental results suggest that the proposed method has high scalability to corpus size and feature dimensionality, with more than 15% topical precision improvement compared with popular clustering techniques.

Categories and Subject Descriptors

H.4.m [**Information Systems**]: Miscellaneous; D.2 [**Software**]: Software Engineering; H.3.3 [**Information Systems**]: Information Search and Retrieval—*Clustering*

[*]This work was done when the first author was visiting Microsoft Research Asia.

General Terms

Algorithms, Design.

Keywords

Community Mining, Clustering, Relational Data.

1. INTRODUCTION

Texts are generated by their authors as a media to express and interchange information, opinions, ideas and comments regarding specific topics. In authors' writings, relations[1] to external resources, such as references, acknowledgements, are deliberately created for better understanding of the mentioned topics. In social sciences, communities constructed on topics and interests have been extensively studied to understand the causality of events and roles of actors. Correspondingly, it is of interest to find communities organized based on latent topics from documents, which can be taken as an accurate reflection of the social communities behind the scene. Imagine that a researcher wants to make a comprehensive survey of a research domain (e.g., data mining), she might issue the following queries:

- *Give me a list of classical papers in the research domain of data mining.*
- *Give me some idea of the names of active researchers in the domain of data mining and their recent publications.*

A traditional information retrieval system may find it difficult to support such queries. However, experts in a specific domain are aware of such information. They will leave answers to these topic-based questions in their writings. Therefore, communities constructed on top of document collections can be used as an effective utility to assist the query processing in intelligent information retrieval systems.

Community discovery can be taken as a special clustering problem. However, text datasets have their unique characteristics which prevent many state-of-the-art clustering techniques from being feasible. First, the size as well as the feature dimensionality are usually very large for text collections. The corpus can grow into a size of millions or billions. Meanwhile, either we follow the "bag of words" principle by

[1]In this paper, *relation* denotes an association between objects and we use it interchangeably with relationship.

taking unigrams as features or use *k-gram* approach, a huge vocabulary size is always expected. Second, inter-document relations are generated and maintained intentionally by document creators, making relations a useful clue in locating and relating topics. These characteristics make community discovery on large data collections an open and challenging problem.

Current clustering methods can be classified into three categories. (1) Textual attribute clustering, e.g., LDA [3] and pLSI [14]. This category of clustering methods utilizes texts to find implicit topics. Although proved effective in discovering latent topics, text-based clustering methods suffer from severe scalability issues. For online archives such as Libra[2] and CiteSeer[3], the document corpus contains more than $1,000,000$ records, which requires highly scalable methods for topic mining. Additionally, such methods usually require solid prior knowledge regarding the parameter settings, which is difficult to be captured. Even worse, these algorithms are sensitive to these parameters. Any update on the document collection may put previously tuned parameters out-of-date and a severe degradation of performance. (2) Relation-based clustering. This category of methods relies purely on relations between documents to discover communities. However, without considering attributes, it is difficult to interpret the meaning of relations. Probing only based on relations can include too many topics into a community and thus cannot yield good results. (3) Hybrid clustering. Some works in the literature propose to consider both attributes and relations. However, these approaches either inherit the scalability issue from the aforementioned attribute-based clustering techniques or fail to recognize the important role of relations in conveying topics. These limitations create a critical obstacle for these methods to be effective.

In this paper, we address the aforementioned issues of community discovery on relational data and propose a solution that utilizes both textual attributes and relations. Our approach is unsupervised, scalable to dataset size and feature dimensionality, and consistent with input parameters. To improve scalability, a filtering process based on relation analysis is first used to find representative community cores. Afterwards, an innovative core merge process is adopted so that consistent communities are returned regardless the initial parameters. The merge step maps cores into virtual bounding boxes in a low-dimension feature space to analyze similarity between cores. Textually-relevant community cores are then merged to form topically coherent ones. Finally, community cores are propagated via relations to form communities and a text-based classification procedure is used to improve the topical precision within communities.

In summary, our contributions are four-fold:

- By analyzing the performance of textual clustering methods, we exploit their scalability and parameter sensitivity issues in community discovery for large-scale textual datasets.

- Based on collected requirements, we define a hierarchical community model. A scalable community discovery solution utilizing both textual attributes and relations is developed.

- To remove the influences of initial parameters, an innovative community merge step is introduced into our solution to generate consistent communities.

- We perform an extensive evaluation and compare the proposed solution using different parameter settings. It is shown our proposal outperforms popular topic clustering methods by at least 15% in topic precision.

The rest of the paper is organized as follows. We give a review on related works in Section 2. An extensive analysis to the performance of LDA, a popular textual topic mining algorithm, is given in Section 3. Steps of the proposed solution for community discovery is given in Section 4. The experimental results are presented in Section 5. Finally, the concluding remarks are given in Section 6.

2. RELATED WORK

In this section some state-of-the-art clustering methods on data attributes and relations are reviewed. We focus the discussion on textual data clustering techniques.

2.1 Textual Attribute Clustering

The first category of data clustering techniques is *Textual Attribute Clustering*, which utilizes texts to discover topics and then uses topics to cluster data objects. This category of approaches (e.g., LSI [7], pLSI [14]) does not consider relations and only relies on attributes. As one of the popular methods applied to textual data collections, *Latent Dirichlet Allocation* (LDA) [3] is a generative probabilistic model for generating large text corpus. LDA is a three-level model, in which for each document a distribution of topics θ is sampled and for each word in the document a topic z is sampled from θ and the word is sampled according to $p(w_n|z,\beta)$, a multinomial probability conditioned on z. In order to use LDA, a group number K needs to be provided and each document will be transformed into a K-dimension vector to represent its location in the topic space. However, shown by literature studies [12], the level of topic coherence of clusters returned by LDA is sensitive to K. In addition, its scalability remains an issue when the document corpus is very large. The performance analysis to LDA is detailed in Section 3.

2.2 Relational Clustering

This category of data clustering methods does not rely on data attributes. They explore object adjacency relations and graph topology to cluster objects. For example, the HITS algorithm finds web object communities [11, 16], which are characterized by a set of *authoritative* pages linked by *hub* pages. Academic community discovery has been performed [20] by expanding from some key papers as centroids of clusters and including affiliated papers. Traditionally, relational clustering methods often use both link analysis and graph partition to locate communities [5, 8, 9, 10]. Random walks [13], for instance, are utilized on a weighted directed graph to update the weights of inter-cluster edges and intra-cluster edges in order to distinguish clusters. Relation-based clustering methods can effectively generate topologically-coherent clusters. However, simply studying links cannot guarantee to find topically-coherent communities because multiple topics can exist within a group of tightly-linked documents.

[2]http://libra.msra.cn

[3]http://citeseer.ist.psu.edu

2.3 Hybrid Clustering

In the literature, there have been some discussions regarding hybrid approaches in text clustering, many of which are extended from generative language models such as LDA. In [2], the authors use a probabilistic generative model to train soft clusters, in which each object is provided with a Dirichlet random variable. When it comes to decide the relationship between two objects, each object will sample a topic distribution from its given Dirichlet random variable, which are supplied to a Bernoulli distribution to determine the relationship. LDA is also extended to include author, group variables for cluster mining [18, 21, 22]. MMRC [17] generates a latent indicator vector to represent the class of a single data object, based on which attributes, homogeneous and heterogeneous relations are modeled. In another piece of work, PLSA is combined with PHITS [6] to provide a unified view of topics in generating texts and links between documents under the assumption that an underlying document generation process exists, which creates both words and hypertext links. However, because these models are extended from generative language models, they also inherit the issues we have discussed in Section 2.1.

In [19], the authors identify the problem of discovering an unknown number of clusters on attribute and relational data, trying to find clusters that are compact inside and distinctive from their *neighborhoods* in a connecting graph. A two-phase solution is proposed. Although both aim at removing the limitation of input parameters, unlike the problem definition of *Connected X Clusters Problem* [19], which utilizes relations to define cluster boundaries, we use the relations to provide topic clues regarding linked documents. Also, scalability remains a critical issue for large datasets.

In summary, although many alternative methods are available for clustering relational data, as far as we know, none of them can be used to effectively and efficiently solve the problem of community discovery for large text collections.

3. ANALYSIS TO LDA ON LARGE-SCALE TEXTUAL DATA

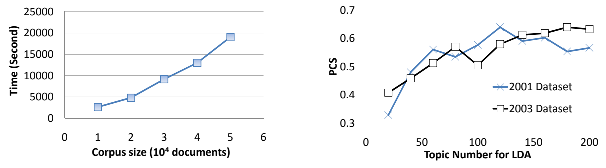
As introduced in Section 2, LDA is a popular language model for topic mining. LDA relaxes the assumption by allowing multiple topics for one document and thus has demonstrated its effectiveness in machine learning tasks on document sets. In terms of text-based topic mining, LDA yields very good performance. Meanwhile, many state-of-the-art hybrid topic mining approaches are extended from LDA. Thereafter, in this section, we exploit the applicability and performance of LDA in clustering large-scale text collections. Two aspects of LDA performance are studied: 1) its scalability to dataset size, and 2) its sensitivity to initial parameter settings.

In this experiment, the computer science publication dataset extracted from CiteSeer is used, which includes abstracts of $575,598$ computer science papers published before 2006. This dataset contains $238,028$ unique terms in the vocabulary, as well as the $1,438,596$ citation links between publications. We use the LDA implementation made available by its authors[4] to mine K topics and conduct all experiments on a Linux server with dual AMD Opteron Processor 252 CPUs ($1G$ Hz), $4GB$ main memory. The parameter settings used in the experiments are summarized in Table 1:

Parameter	Value
Iterations of variational inference	20
Convergence criteria for variational inference	1×10^{-6}
Iterations of variational EM	30
Convergence criteria for variational EM	1×10^{-4}

Table 1: Default LDA parameter settings.

In the scalability experiment, each time a subset of documents, ranging from $20,000$ to $100,000$, is extracted and the time it takes to generate 20 clusters is measured. During our text parsing step, 2-grams are collected and put into the vocabulary instead of unigrams to improve clustering precision. The results are given in Figure 1(a), from which it is found that the overhead basically grows up with the dataset size. Furthermore, we note that with the increase of collection size, the overhead growth rate also increases accordingly. This can be explained by the fact that the vocabulary size grows with the dataset size to make the feature dimension higher. When LDA is applied to the complete dataset, due to the huge size of the document collection, LDA failed to return any results and ran out of memory in initialization. This scalability limitation makes LDA unable to be applied in real systems for topic mining.



(a) LDA scalability to corpus size. (b) LDA sensitivity to topic parameter K.

Figure 1: LDA performance study.

In the next step, the sensitivity of LDA to K, the topic number parameter, is studied. In this study, two non-overlapping datasets, computer science publications in 2001 and 2003, are used. These datasets include $29,869$ and $17,085$ records respectively. For each dataset, LDA is performed using different K settings.

By studying the ACM metadata record of each document, such as publication venue, author affiliation, etc, each documents is classified into one or several of 17 major computer science research topic categories[5]. Each document d_i is assigned with a 17-dimension topic vector z_i. A soft-classification approach is adopted by allowing a paper d_i to belong to multiple topic categories. After clustering, we use the scripts provided by LDA authors to generate top N (N is set to be 100 in our analysis) representative keywords for each cluster, which are manually compared with ACM classification keywords and labeled in the same 17-dimension topic space, i.e, each cluster $\mathcal{C}_i$ is assigned a topic vector Z_i. Based on the LDA cluster topic vectors and document topic vectors, we define a metric $P\bar{C}S$ for clustering precision evaluation, which represents the topical precision for the community discovery task. $P\bar{C}S$ is also used in the

[4]http://www.cs.princeton.edu/ blei/lda-c/

[5]http://www.acm.org/class/1998/overview.html

evaluation section to compare community qualities.

$$HIT_i = \sum_{k:d_{ik} \in \mathcal{C}_i} \eta(d_{ik}), \text{ where } \eta(d_{ik}) = \begin{cases} 1 & \text{if } z_{ik} = Z_i, \\ 0 & \text{if } z_{ik} \neq Z_i. \end{cases} \tag{1}$$

$$PCS_i = \frac{HIT_i}{n_i}, \text{ where } n_i \text{ is the size of } \mathcal{C}_i. \tag{2}$$

In Equation 1 and 2, d_{ik} denotes the k-th document in $\mathcal{C}_i$ and z_{ik} is the topic vector of d_{ik}. The topic of each document is then compared with its affiliated cluster topic label. The result is summarized in PCS_i to represent the topical precision of $\mathcal{C}_i$. We average PCS_i over all communities. The result, denoted as $P\bar{C}S$, is used as our evaluation metrics.

Figure 1(b) presents the change of $P\bar{C}S$ over K for the two experiment datasets. It is observed that the two curves display different patterns. LDA gets the best clustering precision when K is around 120 for the 2001 dataset and 180 for the 2003 dataset. This finding suggests that there is not a best setting of K for all datasets. On the contrary, it needs to be tested repeatedly and tuned to get the best clustering precision, which is around 65% in the tests. Additionally, it is found that both curves in Figure 1(b) do not change monotonically with K. It is observed the $P\bar{C}S$ value decreases when K grows from 80 to 100 and starts to increase when K continues to grow to 120 for the 2003 dataset. In summary, to better utilize LDA for topic clustering, prior knowledge regarding the approximate number of topics in the underlying dataset is required because the clustering precision is very sensitive to the topic number. Otherwise many trials and manual comparisons are needed to infer the best K value. This is not applicable for real datasets, considering the scalability issue of LDA.

4. COMMUNITY DISCOVERY PROCESS

In this section, we first propose a hierarchical community model for textual datasets, based on which a solution for community discovery is introduced. The proposed solution builds internal hierarchy of communities simultaneously when it defines community boundary. This section is organized as follows. First, Section 4.1 illustrates the community model conceptually. Section 4.2 gives an overview of the proposed solution using an example. Afterwards, Section 4.3 introduces essential steps involved in community discovery.

4.1 Community Model

In the literature, there are definitions already given for *communities* [4, 9, 15], most of which claim that a community consists of a set of relevant objects sharing similar interests or topics. A study to Web communities [8] gives a definition by comparing Web linkage with social groups and classifies webpages into *cores* and *fans*. From both empirical observation to real world communities and definitions given by social sciences, we believe that a community consists of a set of topically relevant members with similar attributes and internal relations. Inside a community, there exist *core objects* that are representative and influential in scoping the topic of the community. Remaining members of a community have their roles and ranks inside a community, which forms a hierarchy. It is noted that an object can cross the boundary of communities to make it an interdisciplinary member.

Figure 2 gives an illustration of the proposed community model, in which two communities ($\mathcal{C}_1$ and $\mathcal{C}_2$) are shown.

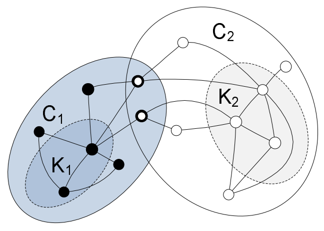


Figure 2: Inner structure of communities for relational data, in which $\mathcal{C}_i$ represents a community and $\mathcal{K}_i$ represents the core members of $\mathcal{C}_i$.

Definition 1 gives the formal definition of our community model:

DEFINITION 1. *A community $\mathcal{C}_i$ is defined as a set of objects $\{d_i | i = 1, 2, ..., n, d_i =< A_i, R_i >\}$, in which A_i denotes the attributes of d_i and R_i is the set of relations connecting with d_i. The topics associated with $\mathcal{C}_i$ are denoted as Z_i. Within $\mathcal{C}_i$, there exists a set of core members $\mathcal{K}_i$ that determines and thus best describes Z_i. Affiliated members of $\mathcal{C}_i$ are ranked by their relevance to Z_i, which creates a hierarchy within $\mathcal{C}_i$.*

4.2 Solution Overview

Based on the proposed community model in Definition 1, we develop a solution to support scalable community discovery on large document collection, utilizing text contents as well as relations. The proposed solution first quickly decomposes the data collection into smaller units by exploring relations. Compared with textual attributes, relations between documents convey consistent topics because they are constructed deliberately by their creators. Handling relations first may reduce the overhead in interpreting and summarizing texts significantly so that the solution has a high scalability regardless the dataset size. Through a relation topology analysis, a set of preliminary community cores is returned and later expanded into communities. To reduce the effects of parameter setting used in the first step and produce consistent communities, a core merge step is added on the preliminary cores. Afterwards, attributes are used within a community, whose size is much smaller compared with the original data collection, to determine the relevance of each community member that is included into the community through relation propagation. Non-relevant documents, although connected with the community, are eliminated.

Figure 3 illustrates the steps taken for community discovery in the solution. In this figure, a given collection of documents (d_a to d_l) are represented by solid circles and labeled from a to l. Directed relations are also given by solid arrows. Basically, the algorithm consists of four steps, namely:

1. **Core Probing**: In this step, the solution starts from relation analysis to find cores, which are characterized by documents that are frequently referenced. Each core, denoted as $\mathcal{K}_i$, will be taken as a community seed in the following steps. This step is effective in limiting the analysis scope and thus improves solution scalability.

2. **Core Merge**: In this step, cores are merged based on their topical similarity to reduce the sensitivity to

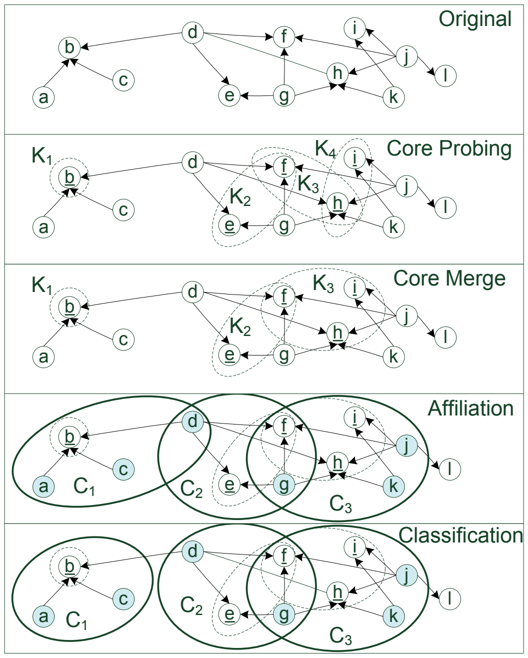


Figure 3: Illustration of community discovery steps.

parameters used in core probing. In the figure, It is found $\mathcal{K}_3$ and $\mathcal{K}_4$ are linked and relevant. They are merged in this step to contain d_f, d_h, and d_i.

3. **Affiliation**: After cores are constructed, relations are used to expand cores into initial communities. Bold solid circles in the subfigure of this step denote community boundaries.

4. **Classification**: Since linked documents may not be topically relevant, a classification process is conducted on each member of a community based on texts to finalize the affiliation relationship. False hits are removed, e.g., d_d in C_1 in the figure.

In the following section, we give algorithmic details to the community discovery steps listed above.

4.3 Community Discovery Steps

4.3.1 Core Probing

Based on Definition 1, cores dictate the formation and topics of communities. Previous approaches usually use simple heuristics such as relation frequency to locate cores. We argue, however, finding core documents of a community via simple relation count or content analysis does not necessarily yield good results. As suggested in [16], *hub* objects tend to have many outgoing relations, which usually cover many communities and are thus too generic in topics. On the other hand, attribute analysis can possibly find documents that are typical in a specific topic but without actual influence, which is both expensive in computation and inaccurate.

To obtain more topically coherent cores without incurring high computational overhead, in our solution core candidates are discovered by co-occurrence analysis. We believe that when multiple objects are linked simultaneously by others, they are more likely to be able to define a coherent topic scope. Definition 2 specifies the community cores definition.

Definition 2. *The core $\mathcal{K}$ of a community $\mathcal{C}$ is defined as a set of members that are simultaneously linked by many affiliated members in $\mathcal{C}$. It is not required that objects in the core of a community should be tightly linked to each other.*

Based on Definition 2, the co-occurrence relation analysis can be used to find community cores for textual datasets. We do not require that core members have many direct links in between because it is often observed that several independent topics eventually merge into one coherent new topic over time. For each document d_i, the list of all outgoing relations $R_i^{out} = \{r_{ij}\}$ is generated. Thus, the problem of finding core documents is reduced to a problem of calculating frequent itemsets in the associate rule algorithm [1]. Since finding all possible combinations of community members is extremely expensive, we start from short itemsets, which will be reused to find longer itemsets. By using relation analysis, the probing step can efficiently find a set of cores that is very small compared with the whole document collection, which achieves very high scalability.

The core-discovery algorithm is based on the *Apriori* algorithm but different in the following two aspects:

- Instead of using a fix filtering threshold to find popular patterns, variant filtering thresholds are used according to the length of itemsets because long frequent itemsets tend to have smaller supports than short ones.
- It is found that long itemsets generated by *Apriori* algorithm are often supersets of other itemsets. In such circumstances, if two frequent itemsets S_1 and S_2 are found and $S_1 \subset S_2$, we do not keep S_1.

The formal description of the core-probing algorithm is outlined in Algorithm 1.

Algorithm 1 Extracting core members of a community.

```
Input: $L_1$: a large 1-item set of objects $\{d_j | d_j$ is linked by a document $d_i$ in the corpus$\}$
$t$: filtering threshold
Output: a set of cores, $\mathbf{K}$
  for $m = 2$; $L_{m-1} \neq \emptyset$; $m++$ do
    $C_m = apriori - gen(L_{m-1})$;
    for each candidate $c \in C_m$ do
      c.count = co-occurrence number of c;
    end for
    $L_m = \{c \in C_m | c.count * m \geq t\}$;
    for each candidate $c \in L_m$ do
      for each candidate $c' \in L_{m-1}$ do
        if $c' \subset c$ then
          delete $c'$ from $L_{m-1}$;
        end if
      end for
    end for
  end for
  return $\mathbf{K} = \bigcup_{i=1}^{m} L_i$;
```

4.3.2 Improving Core Consistency

In Algorithm 1, a filtering threshold t needs to be provided, based on which cores are probed. Intuitively, a lower t value can create more cores than a higher t value, which places a similar problem as with LDA, which needs users to supply the number of topics. To overcome this problem, a core merge process is introduced, which tries to analyze topical similarity using textual analysis and reduce the effects of improper t settings.

Remember that t stands for the number of co-occurrences for frequent itemsets. It is observed that after applying two different thresholds t_1 and t_2 ($t_1 < t_2$) every core returned by t_2 can find a corresponding core in the result returned by t_1, which can be formally summarized as:

PROPERTY 1. *Given two core probing thresholds t_1 and t_2 ($t_1 < t_2$), the core probing algorithm returns two sets of core candidates respectively, namely $\mathbf{K}_1 = \{\mathcal{K}_i^{(1)}\}$ and $\mathbf{K}_2 = \{\mathcal{K}_j^{(2)}\}$. $\forall \mathcal{K}_i^{(2)} \in \mathbf{K}_2$, $\exists \mathcal{K}_j^{(1)} \in \mathbf{K}_1$, s.t. $\mathcal{K}_j^{(1)} \supseteq \mathcal{K}_i^{(2)}$.*

Property 1 suggests that low t values can always find a more complete core candidate set and the overhead remains much lower than LDA. For example, if $\{d_1, d_2\}$ is returned in $\mathbf{K}_2$, there must be a core in $\mathbf{K}_1$ that is the superset of $\{d_1, d_2\}$. However, the negative effect is that too many candidates are returned by lower t values and many of them contain overlapping members such as $\{d_1, d_2, d_3\}$ and $\{d_1, d_2, d_4\}$. For such cases, it is often found actually d_3 and d_4 are really relevant documents. It would be good if these two sets can be merged as $\{d_1, d_2, d_3, d_4\}$. Another problem arises here: How can we decide when to merge these similar cores? Simply relying on the overlapped entries in the two sets cannot promise good results because d_3 and d_4 in the above sets may discuss two totally different topics whereas d_1 and d_2 are only two documents providing tools or backgrounds. A core merge step is introduced to identify core candidate sets that can be merged to generate more complete and consistent cores. A hybrid policy is used, which not only studies the overlapping level of cores but also uses their topical distributions to determine when to merge cores.

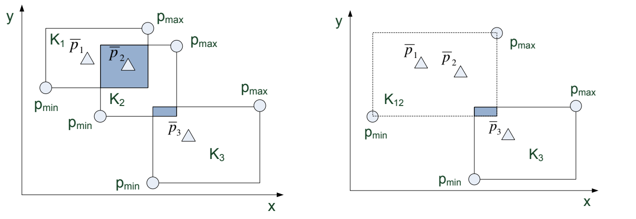
Given two cores $\mathcal{K}_i$ and $\mathcal{K}_j$, it is required that they have at least one member in common to make them qualified for merging. Namely,

$$\|\mathcal{K}_i \cap \mathcal{K}_j\| > 0$$

Afterwards, the candidate cores are sent to study their topical similarities by text analysis. LDA is applied on all core candidate documents, whose size is much smaller than that of the whole collection. LDA is used not to find communities. On the other hand, it is used as a utility to reduce the dimension of the feature space for documents. Although the original term space can be used, this approach builds a feature space that is as large as the size of the textual vocabulary of the collection, which is expensive in computation. Moreover, the term-document matrix is very sparse. Among all terms in the vocabulary, only a very small subset has strong discrimination effect in topic clustering, which is very difficult to capture without global-scale distributional study. On the other hand, as proven in [3], LDA can effectively reduce the dimensionality of text representation and keep good discriminative information.

LDA is told to generate n' topics, in which n' should be an integer that is large enough. The low-dimensional document representation is used to discover the similarity between documents. n' will not influence the merge results very much, which will be shown by experiments in Section 5. After dimension reduction, each document in the dataset is represented as a n'-tuple vector.

For each core $\mathcal{K}_i$, in the low-dimension feature space, the



(a) Original cores. (b) After merging $\mathcal{K}_1$ and $\mathcal{K}_2$.

Figure 4: Example core merge in a two-dimension view.

following coordinates are calculated:

$$p_i^{min} =< v_{ij}^{min} | j = 1, 2, ..., n', v_{ij}^{min} = \min_{d \in \mathcal{K}_i} v_j > \quad (3)$$

$$p_i^{max} =< v_{ij}^{max} | j = 1, 2, ..., n', v_{ij}^{max} = \max_{d \in \mathcal{K}_i} v_j > \quad (4)$$

$$\bar{p_i} =< \bar{v_{ij}} | j = 1, 2, ..., n', \bar{v_{ij}} = \frac{\sum_{d \in \mathcal{K}_i} v_j}{\|\mathcal{K}_i\|} > \quad (5)$$

In the above formulae, v_{ij}^{min} and v_{ij}^{max} stand for the minimal and maximal value of feature dimension j for all documents that belong to the core $\mathcal{K}_i$, respectively. Thus, p_i^{min} and p_i^{max} generate a bounding box for each core in the feature space. For each pair of cores $\mathcal{K}_i$ and $\mathcal{K}_j$, we calculate their intersection, denoted as R_{ij}, based on their p^{min} and p^{max}. Notice that there is no need to calculate R_{ij} if the value of any dimension of p_i^{min} is larger than the corresponding dimension value of p_j^{max} since they cannot overlap at all. This initial condition is very helpful in eliminating unnecessary comparisons. Additionally, in our implementation, two indices for p^{min} and p^{max} are maintained respectively to facilitate the calculation.

If two cores overlap in the feature space ($R_{ij} \neq \emptyset$). The significance of their overlapping is analyzed by checking if either $\bar{p_i}$ or $\bar{p_j}$ is bounded by R_{ij}. We select to merge $\mathcal{K}_i$ and $\mathcal{K}_j$ if $\bar{p_i} \in R_{ij}$ or $\bar{p_j} \in R_{ij}$. After two cores, $\mathcal{K}_i$ and $\mathcal{K}_j$, are merged, the new core is formed as $\mathcal{K}_{ij} = \mathcal{K}_i \cup \mathcal{K}_j$.

p^{min} and p^{max} of $\mathcal{K}_{ij}$ are updated using Equation 3 and Equation 4 respectively. However, instead of updating $\bar{p_{ij}}$ using the topic centroid of all members of $\mathcal{K}_{ij}$, the separate topic centroids of $\mathcal{K}_i$ and $\mathcal{K}_j$ are kept. When it is found any topic centroid of $\mathcal{K}_{ij}$ falls in its intersection region with other cores, the two cores can merge. Formally speaking, we select to merge two core $\mathcal{K}_i$ and $\mathcal{K}_j$ if and only if:

$$\exists \bar{p} \in \bar{p_i} \cup \bar{p_j}, \bar{p} \in \mathcal{K}_i \cap \mathcal{K}_j \quad (6)$$

A stronger merge condition can be placed by requiring all topic centroids of involved cores to fall into $\mathcal{K}_i \cap \mathcal{K}_j$, which is described as:

$$\forall \bar{p} \in \bar{p_i} \cup \bar{p_j}, \bar{p} \in \mathcal{K}_i \cap \mathcal{K}_j \quad (7)$$

From Equation 6 and Equation 7, the following property of the core merge process can be found, in which the operator "$\ltimes$" represents the core merge operation.

PROPERTY 2.

$$(\mathcal{K}_i \ltimes \mathcal{K}_j) \ltimes \mathcal{K}_l = \mathcal{K}_i \ltimes (\mathcal{K}_j \ltimes \mathcal{K}_l)$$

Based on Property 2, it is found the sequence of selecting the core candidates for merge does not have any effect on the final results, which is denoted as *Associativity* of the core merge operation. So the algorithm will start from arbitrary pairs of core candidates and apply the merge procedure on them. This process is performed iteratively until no merge is available.

The process can be illustrated with Figure 4, in which there are originally three core candidate sets ($\mathcal{K}_1$, $\mathcal{K}_2$, and $\mathcal{K}_3$) in Figure 4(a). The p^{min} and p^{max} values are shown as small solid circles on the corners of bounding boxes. Also, the topic centroid of each core is plotted in the figure with a small triangle. It is evident that $\bar{p_2}$ lies within the intersection area of $\mathcal{K}_1$ and $\mathcal{K}_2$, which makes them qualified for merge according to Equation 6. On the other hand, although $\mathcal{K}_2$ and $\mathcal{K}_3$ have an intersection area in the figure, they cannot be merged because neither have its topic centroid in the shadowed intersection region. After merge, the newly constructed core $\mathcal{K}_{12}$ is then compared again with $\mathcal{K}_3$ (Figure 4(b)). It is found there is none of $\bar{p_1}$, $\bar{p_2}$ and $\bar{p_3}$ is in the shadow region of Figure 4(b). So the merge process terminates. Otherwise, if we choose the policy described in Equation 7, $\mathcal{K}_1$ and $\mathcal{K}_2$ cannot be merged either.

The whole process is specified in Algorithm 2.

Algorithm 2 Merging core candidates.

```
Input: $S$: initial core candidates returned by core probing
Output: a refined set of cores $S'$
  $S_0 = \phi$;
  while $S_0 \neq S$ do
    $S_0 = S$;
    for each pair of cores $\mathcal{K}_i$ and $\mathcal{K}_j$ in $S$ do
      if $\|\mathcal{K}_i \cap \mathcal{K}_j\| > 0$ then
        calculate $p^{min}$, $p^{max}$, and $\bar{p}$ using Equation 3,
        Equation 4, Equation 5 respectively;
        if the intersection of $\mathcal{K}_i$ and $\mathcal{K}_j$, $R_{ij}$, is not empty
        then
          if either $\bar{p_i} \in R_{ij}$ or $\bar{p_j} \in R_{ij}$ then
            add $\mathcal{K}_i \cup \mathcal{K}_j$ into $S$;
            remove $\mathcal{K}_i$ and $\mathcal{K}_j$ from $S$;
          end if
        end if
      end if
    end for
  end while
  return $S' = S$;
```

By applying the core merge step to the initial cores, more succinct yet complete cores can be generated for later expansion. More importantly, this step guarantees that lower t values can generate a collection of cores that include all cores that would be returned by using a higher t values. In the core probing process, for each core $\mathcal{K}_i$ we can record its frequency f_i of co-occurrences in the collection and rank communities based on f_i. The following property stands for the community ranking:

PROPERTY 3. *Given two core probing thresholds t_1 and t_2 ($t_1 < t_2$), the core merge algorithm returns two sets of core candidates respectively, namely $\mathbf{K}_1 = \{\mathcal{K}_i^{(1)}\}$ and $\mathbf{K}_2 = \{\mathcal{K}_j^{(2)}\}$. Cores in $\mathbf{K}_1$ and $\mathbf{K}_2$ are ranked according to their co-occurrence frequencies respectively and recorded in the ranking sequence. $\forall \mathcal{K}_i^{(2)} \in \mathbf{K}_2$, $\exists \mathcal{K}_j^{(1)} \in \mathbf{K}_1$, s.t. $\mathcal{K}_j^{(1)} \supseteq \mathcal{K}_i^{(2)}$ and $\mathcal{K}_j^{(1)}.rank = \mathcal{K}_i^{(2)}.rank$.*

According to Property 3, the frequency-based community ranking is consistent regardless t values. For $\mathbf{K}_1$ and $\mathbf{K}_2$ ($t_1 < t_2$), the top-k ($k = \|\mathbf{K}_2\|$) rankings of communities based on their co-occurrence frequencies are consistent. Higher t values can promise to return a small set of representative cores and lower t values can generate a larger size of cores with representative cores promoted. Therefore, Property 3 guarantees a high consistency of returned cores regardless the initial clustering setting.

4.3.3 Affiliation Propagation

After finding cores $\mathcal{K}$ of a community $\mathcal{C}$, all remaining documents $\{d_i | d_i \in \mathcal{C} - \mathcal{K}\}$ are taken as affiliated members, whose affiliation relationship can be determined through relation propagation. $\mathcal{C}$ is initialized to be $\mathcal{K}$. Iteratively, for each document d_i in $\mathcal{C}$, we find all documents in the corpus that link to d_i and add them to $\mathcal{C}$. To avoid infinity loops caused by relation circles, we can limit the number of iterations. The propagation process terminates if no new documents are added to $\mathcal{C}$ in an iteration.

Among affiliated members, interdisciplinary members $\mathcal{D}$ can be identified by checking the common member sets between any two communities. If it is found that a member belongs to multiple communities, it is identified as *interdisciplinary member* of $\mathcal{C}$.

During the affiliation propagation process, internal hierarchy of a community is built simultaneously. We evaluate the relevance of a member to the affiliated community with its closeness to a community core. The closer a member is to $\mathcal{K}$, the higher its rank is. In the community propagation process, every time a new document is inserted into a community, the iteration value is recorded to represent its distance to the cores.

4.3.4 Intra-Community Classification

Finding communities solely based on relation propagation can generate false hits because of weak relations with topical ambiguity. After the relation study, in this stage, communities are refined by filtering out these false hits from community candidates using attribute analysis.

Theoretically, this pruning process can be viewed as a special classification task. Given a community $\mathcal{C}_i = \{d_{i1}, d_{i2}, ..., d_{in}\}$ with its core $\mathcal{K}_i$, we want to classify $\mathcal{C}_i$ into two collections $\mathcal{C}_i^{'}$ and $\bar{\mathcal{C}}_i$ so that $\mathcal{C}_i = \mathcal{C}_i^{'} \cup \bar{\mathcal{C}}_i$ and $\mathcal{K}_i \subseteq \mathcal{C}_i^{'}$. In such a setting, each document d_j in the core $\mathcal{K}_i$ is taken as a positive example, which suggests *d_j belongs to $\mathcal{C}_i$*. We also sample documents from other community cores, which have dramatic different latent topic distributions, as negative examples. LDA is applied to the dataset to reduce dimensionality (n'). After clustering, each document in $\mathcal{C}_i$ as well as all sampled documents will have a feature vector to represent its topical position in the feature space, which is denoted as $z_i =< v_1, v_2, ..., v_{n'} >$. The absolute value of the LDA result is not used as a document's actual topic because we do not tune n' in this process. On the other hand, the vector representation of each document is taken as features used in the classification process to reveal the topical similarity between documents.

After dimension reduction, the training set, including $\mathcal{K}_i$ and documents from other communities, is supplied to Support Vector Machine (SVM) to train a binary classifier. All documents in $\mathcal{C}_i$ are classified and negative labeled documents are removed from $\mathcal{C}_i$.

From the previous introduction to the community discovery process it is observed that the necessary attributes we defined in Definition 1 (internal ranking, interdisciplinary members, and community cores) are constructed during the process, which do not require a second run to the dataset.

5. EVALUATION

In this section, we perform an empirical evaluation of our proposal. Section 5.1 describes the experiment settings and the testing methods. Section 5.2 reports our experiment results as well as analysis.

5.1 Experiment Settings

We conduct our evaluations based on an academic dataset extracted from the CiteSeer computer science digital library[6]. Papers and their associated citations are extracted for community discovery. For each paper d_i, its abstract, title, author, venue, as well as citation relations are collected. Table 2 gives the summary of the dataset as well as the default settings of parameters, which are used unless explicitly specified.

Number of papers	$575,598$
Number of references	$1,438,596$
Size of vocabulary	$238,028$
Filtering threshold (t)	14
Propagation iteration (n)	5
LDA parameter (n')	10

Table 2: Dataset summary for experiments.

We use two metrics, namely mining time overhead and $P\bar{C}S$, both discussed and used in Section 3. For each obtained result set, 10% of communities are randomly sampled. The topic label of a community (Z_i) is obtained manually by checking the metadata records of its core papers.

5.2 Experiment Results

5.2.1 Scalability Study

First, to test the scalability of proposed algorithm, we vary the size of datasets and filtering thresholds.

Figure 5, which plots the time cost corresponding to different sizes of datasets ranging from $50,000$ documents to $500,000$ documents, shows that the algorithm finished the community discovery task quickly despite of dataset sizes. It takes about $1,200$ seconds for the complete dataset ($500,000$ documents) and only around 400 seconds for a dataset of $50,000$ papers. Compared with LDA which takes about $20,000$ seconds to process the dataset of $50,000$ documents (as shown in Figure 1(a)), the proposed algorithm is much faster (i.e., in more than two orders of magnitude). Regardless the dataset size, our algorithm can quickly focus itself on a small subset of core documents and make textual analysis on them. Based on the trends of curves in these two figures, we also find our proposal is much more scalable than LDA. More importantly, when applied to the complete testing dataset, our algorithm does not suffer from memory limitation whereas LDA fails to return any result in our testing platform.

The follow-up study checks the algorithmic cost in finding communities for different filtering threshold settings on the complete dataset. The results are plotted in Figure 6. In this test, the filtering threshold t is varied from 11 to 134. Basically, a low threshold can create more community core candidates for the later merge process. Thereafter, the overhead in discovering communities with low thresholds is expected to be higher, which is confirmed in the figure.

[6]http://citeseer.ist.psu.edu/oai.html

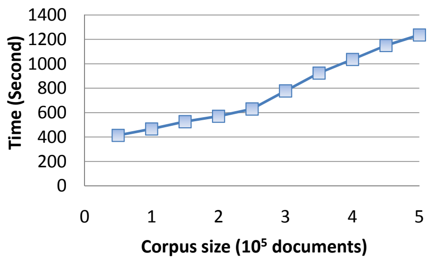


Figure 5: Scalability test to dataset sizes.

However, for the lowest t value in the figure, the overall mining time is only $1,502$ seconds. Figure 6 also shows the overhead in each of the four steps in community discovery. The step-wise comparison reveals the major cost difference is caused by the core probing phase and core merge phase. After merging, the overheads of the algorithm under different threshold settings are similar, suggesting the consistency in merged communities for different parameter settings.

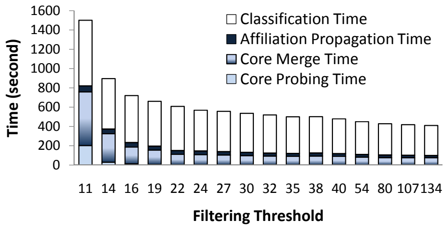


Figure 6: Time overhead to discover communities with different filtering thresholds.

5.2.2 Quality Study

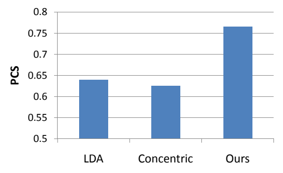
In this section, we try to compare our proposed algorithm with several typical textual data clustering approaches. Namely, we choose LDA as the representative of textual attribute clustering technique, which has been introduced in Section 3. Meanwhile, we choose a relational clustering model, Concentric-Circle Model [23], which identifies cores of communities only via relation study. Due to the scalability problem of LDA, we use the document set containing all $29,869$ papers published in 2001 as the test dataset. The proposed algorithm, together with the Concentric-Circle Model, uses the default parameter setting listed in Table 2. As of LDA, we use the topic number that generates best community topic precision in our tests in Section 3, 120, as the input parameter.

Our experimental results are given in Figure 7. From Figure 7(a), we find our proposed solution (labeled as *Ours*) generates the best quality communities. Its $P\bar{C}S$ is better than the other two approaches for more than 15%. One of the reasons for high $P\bar{C}S$ can be found in Figure 7(b), which suggests the proposed solution creates communities with smallest sizes. On the other hand, LDA predefined the number of topics whereas the Concentric Model includes too many non-relevant documents in communities because no attribute analysis is performed in its clustering process.

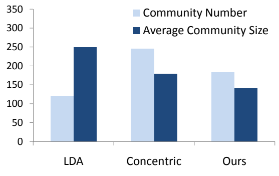
5.2.3 Parameter Sensitivity Study

In this section, we perform a sensitivity test of the returned community quality (in terms of topic precision) against various parameter settings.

First of all, we conduct an experiment to see the effect of



(a) $\bar{PCS}$.



(b) Sizes.

Figure 7: Algorithm comparison.

the filtering threshold t on the community quality as well as the effect of the merge process. Figure 8 gives the comparison results. As shown, although different filtering threshold t values result in different number of communities and different community precisions, the $\bar{PCS}$ values are very stable, achieving over 80% for most cases. Figure 8 also shows that the core merge process is very effective in reducing the negative effects of filtering threshold settings. For example, after merging, the number of cores is reduced from 4,264 to 2,209 ($t = 14$) and from 3,144 to 1,839 ($t = 16$). The gap between the sizes of the core collections returned by these two tests is greatly narrowed, suggesting that a very consistent community discovery regardless the initial t settings. Compared with the experiment on LDA (as shown in Section 3), our community discovery solution provides a larger *sweet zone* for its input parameters. On the other hand, if an improper filtering threshold is set, our algorithm can still promise a high consistency in popular communities due to Property 3 in Section 4.3.2.

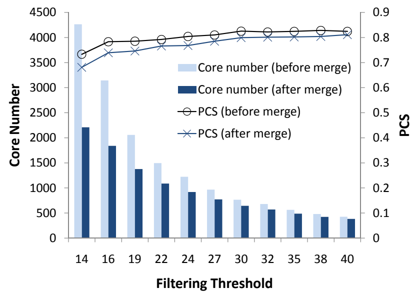
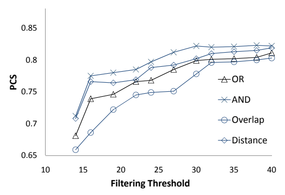


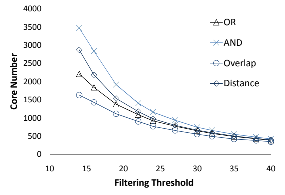
Figure 8: Effects of filtering threshold t and the merge process.

Meanwhile, it can be observed that the merge process may lower the topic precision slightly because documents that are originally assigned to different communities may be put into one community after merge. The topic coverage is thus larger and the distance of outlier documents to the community centroid is extended.

Next, we continue to study the impact of different merge policies to determine the best strategy in merging cores. In the experiment, four different core merge policies are put into comparison, which are (1) **OR Policy**, as introduced in Equation (6); (2) **AND Policy**, which requires both topic centroids of target cores to be within the overlapping topic area, as described in Equation (7); (3) **Overlap Policy**, which solely based on the co-occurrence of common members between two candidate cores and neglect the attributes. Two cores are merged as long as they have any common member; and (4) **Distance-Based Policy**, which check the distance $\bar{d_{ij}}$ between the two topic centroids of cores. To avoid introducing new distance threshold, we find a pair of documents from $\mathcal{K}_i$ and $\mathcal{K}_j$ respectively which have the longest distance between each other, denoted as d_{ij}^{max}. If $\bar{d_{ij}}$ is less than half of d_{ij}^{max}, the two cores are taken as relevant and are merged afterwards. The experimental results are given in Figure 9.



(a) $\bar{PCS}$.



(b) Number of cores.

Figure 9: Compare different merge policies.

From Figure 9(a) and Figure 9(b), it is observed that the AND-policy is the strongest, sacrificing the merge opportunities for higher precision. On the other hand, the Overlap-policy is the weakest policy and its overall topic precision is thus the lowest. The OR-policy and the Distance-based policy yields similar results. However, the Distance-based policy requires pair-wise distance computation between documents in cores, which is very expensive for high dimension feature spaces. Thereafter, since the OR-policy is very simple and effective, it is used as the default merge policy in the following experiments.

In the core merge process, as described in Section 4.3.2, each core document is transformed into a vector in a low-dimension feature space using LDA. Thus, a new variable, n', is introduced. A natural question arises correspondingly, "Will the selection of n' greatly impact the final community precision like what we already see in the LDA clustering analysis in Section 3?" To answer this question, a series of experiments are performed, using different n' values to test the community precision. The results are given in Figure 10.

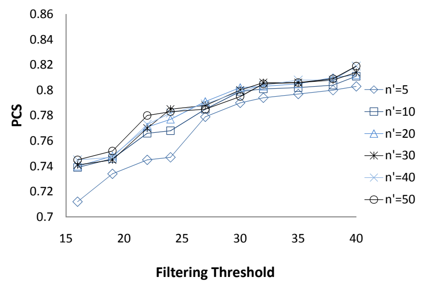


Figure 10: Effects of n' to community precision.

In this experiment, n' is varied from 5 to 50 to observe its impact on the precision of returned communities. Basically, as shown in the figure, the difference is very small and thus is negligible. Extremely small n' values ($n' = 5$) can lower $\bar{PCS}$ by 5%. For remaining tests, the community quality remains stable regardless n' values. It is also observed that higher n' does not necessarily generate better community results in terms of $\bar{PCS}$.

Finally, we show the effects of the member classification phase in community discovery. Figure 11 gives the change of $\bar{PCS}$ as well as the average size of communities. It is found the classification phase is very effective in promoting the topic consistency of a community by eliminating topically non-relevant documents. A significant improvement in the $\bar{PCS}$ value is found for each test (about 10%). From the average community size distribution in Figure 11, the average community size increases as the filtering threshold increases, whether it's before or after classification. This is because a higher filtering threshold can find *hot* community topics that attract many followers. However, after classification, the average size of communities becomes relatively stable for each test.

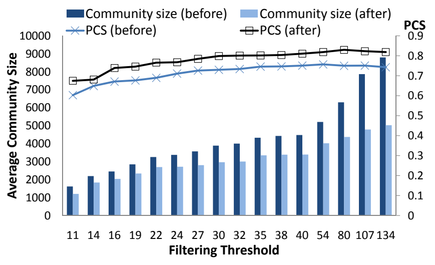


Figure 11: Effects of classification.

6. CONCLUSION

In this paper, we exploit the problem of current clustering techniques for community discovery on textual datasets. A new hierarchical community model is proposed to capture the observation that a community typically consists of a set of tightly relevant members with common latent topics. Based on this model, a community discovery solution is developed to discover communities from large-scale document corpus. Our solution overcomes the scalability issue by limiting analysis scope to a small subset using relation analysis. A core merge step is employed to make the solution less sensitive to initial parameter settings. Evaluation results validate our ideas and show that our solution outperforms existing community discovery techniques. The sensitivity tests further demonstrate our approach works very well in discovering useful and accurate communities, with low costs. In the future, we plan to study the dependency of communities and their causality, which can be used to predict future trends and transitions.

7. REFERENCES

[1] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In P. Buneman and S. Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington, D.C., 26–28 1993.

[2] E. Airoldi, D. Blei, E. Xing, and S. Fienberg. A latent mixed membership model for relational data. In *LinkKDD '05: Proceedings of the 3rd international workshop on Link discovery*, pages 82–89, New York, NY, USA, 2005. ACM Press.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.

[4] Y. Chi, S. Zhu, X. Song, J. Tatemura, and B. L. Tseng. Structural and temporal analysis of the blogosphere through community factorization. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 163–172, New York, NY, USA, 2007. ACM Press.

[5] D. Cohn and H. Chang. Learning to probabilistically identify authoritative documents. In *Proc. 17th International Conf. on Machine Learning*, pages 167–174. Morgan Kaufmann, San Francisco, CA, 2000.

[6] D. A. Cohn and T. Hofmann. The missing link - a probabilistic model of document content and hypertext connectivity. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *NIPS*, pages 430–436. MIT Press, 2000.

[7] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[8] Y. Dourisboure, F. Geraci, and M. Pellegrini. Extraction and classification of dense communities in the web. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 461–470, New York, NY, USA, 2007. ACM Press.

[9] G. Flake, S. Lawrence, and C. L. Giles. Efficient identification of web communities. In *Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 150–160, Boston, MA, August 20–23 2000.

[10] B. Gao, T.-Y. Liu, X. Zheng, Q.-S. Cheng, and W.-Y. Ma. Consistent bipartite graph co-partitioning for star-structured high-order heterogeneous data co-clustering. In *KDD '05: Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 41–50, New York, NY, USA, 2005. ACM Press.

[11] D. Gibson, J. M. Kleinberg, and P. Raghavan. Inferring web communities from link topology. In *UK Conference on Hypertext*, pages 225–234, 1998.

[12] T. L. Griffiths and M. Steyvers. Finding scientific topics. *Proc Natl Acad Sci U S A*, 101 Suppl 1:5228–5235, April 2004.

[13] D. Harel and Y. Koren. Clustering spatial data using random walks. In *Knowledge Discovery and Data Mining (KDD'01)*, pages 281–286, 2001.

[14] T. Hofmann. Probabilistic latent semantic analysis. In *Proc. of Uncertainty in Artificial Intelligence, UAI'99*, Stockholm, 1999.

[15] M. Kitsuregawa, M. Toyoda, and I. Pramudiono. Web community mining and web log mining: commodity cluster based execution. *Aust. Comput. Sci. Commun.*, 24(2):3–10, 2002.

[16] J. M. Kleinberg. Hubs, authorities, and communities. *ACM Comput. Surv.*, page 5.

[17] B. Long, Z. M. Zhang, and P. S. Yu. A probabilistic framework for relational clustering. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 470–479, New York, NY, USA, 2007. ACM.

[18] D. Mimno and A. McCallum. Expertise modeling for matching papers with reviewers. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 500–509, New York, NY, USA, 2007. ACM.

[19] F. Moser, R. Ge, and M. Ester. Joint cluster analysis of attribute and relationship data withouta-priori specification of the number of clusters. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 510–519, New York, NY, USA, 2007. ACM.

[20] A. Popescul, G. Flake, S. Lawrence, L. Ungar, and C. L. Giles. Clustering and identifying temporal trends in document databases. In *Advances in Digital Libraries, ADL 2000*, pages 173–182, Washington, DC, 2000.

[21] M. Rosen-Zvi, T. Griffiths, P. Smyth, and M. Steyvers. Learning author topic models from text corpora. Technical report, November 2005.

[22] X. Wang, N. Mohanty, and A. McCallum. Group and topic discovery from relations and text. In *LinkKDD '05: Proceedings of the 3rd international workshop on Link discovery*, pages 28–35, New York, NY, USA, 2005. ACM Press.

[23] W.-J. Zhou, J.-R. Wen, W.-Y. Ma, and H.-J. Zhang. A concentric-circle model for community mining in graph structures. Technical Report MSR-TR-2002-123, Microsoft Research Asia, Beijing, China, November 2002.