Knowl Inf Syst (2008) 17:335–354
DOI 10.1007/s10115-008-0135-5

REGULAR PAPER

Finding cohesive clusters for analyzing knowledge communities

Vasileios Kandylas · S. Phineas Upham · Lyle H. Ungar

Received: 29 October 2007 / Revised: 23 December 2007 / Accepted: 29 January 2008 /
Published online: 4 April 2008

Abstract Documents and authors can be clustered into “knowledge communities” based on the overlap in the papers they cite. We introduce a new clustering algorithm, Streemer, which finds cohesive foreground clusters embedded in a diffuse background, and use it to identify knowledge communities as foreground clusters of papers which share common citations. To analyze the evolution of these communities over time, we build predictive models with features based on the citation structure, the vocabulary of the papers, and the affiliations and prestige of the authors. Findings include that scientific knowledge communities tend to grow more rapidly if their publications build on diverse information and if they use a narrow vocabulary.

1 Introduction

A *knowledge community* [2] is an informal community of researchers who build on each other’s ideas and share similar interests. Such communities usually consist of people doing research on the same or closely related field. They are also known as *intellectual communities*, or *schools of thought* [19]. Belonging to such a community has the advantage for a researcher of making it easier to disseminate and gather new knowledge. Papers of interest are easier to find, new papers have an already existing audience and the content is familiar and thus accessible.

Within a knowledge community, most of the work is related. New results build upon previous discoveries, thus papers often cite other papers in the same community. Certain high impact or historical papers will be frequently cited within the community. Papers from

V. Kandylas · L. H. Ungar (✉)
Department of Computer and Information Science,
University of Pennsylvania, Philadelphia, PA 19104, USA
e-mail: ungar@cis.upenn.edu

V. Kandylas
e-mail: kandylas@seas.upenn.edu

S. P. Upham
Wharton School, University of Pennsylvania, Philadelphia, PA, USA

related communities will also be cited occasionally. Since patterns of citation are shared among papers in the same community, knowledge communities can be identified by clustering documents based on their citations.

Our goal is to discover and analyze knowledge communities. We wish to know what features of a community predict whether it will grow or shrink. To do this, we cluster papers based on the papers they cite. As explained in detail later, not every document is assigned to a community; many documents are assigned to a background cluster. Once the knowledge communities are discovered, we fit a supervised model and find which features of the clusters are statistically significant in predicting cluster growth. Features used include the citation patterns, (e.g., how many of the citations are to papers within or external to the community), vocabulary usage (e.g., how unique are the words used to the cluster), and exogenous measures such as the fraction of authors who have an industrial rather than academic affiliation.

We classify knowledge communities along a number of dimensions, including their *knowledge* use, as measured by what papers or communities they cite, and *rhetoric*, as measured by what vocabulary they use and how vocabulary changes over time or across communities. We find that successful communities use knowledge and rhetoric in opposite ways. Successful communities flexibly use broad *knowledge*. They have a unique area of focus, but cover new areas and incorporate knowledge from multiple domains. Also, successful communities use narrow, unchanging vocabulary, which is common across knowledge communities. Apparently, novel work that is described with new terminology is harder to understand; standardized terms can reduce this difficulty.

1.1 Foreground/background clustering for text mining

To reach these conclusions, we cluster papers into foreground clusters (knowledge communities), which cite many of the same papers, or into a more diffuse background. We take the notion of foreground and background from vision, where there are often cohesive foreground objects in front of a more diffuse background. In text mining, foreground clusters are cohesive communities or topics. The background consists of lower density regions, where there are fewer similar items. For example, when clustering documents, the foreground will consist of documents in some specific area (e.g., that cite the same papers), while the background will contain documents that cover several different areas on topics that are not widely cited.

Use of foreground and background is important for community analysis, where one wants to find the tight groups of papers or people. Clustering published documents according to what they cite, as we do in this paper, reveals knowledge communities, their sizes and how they evolve in time by drifting in nearby areas of scientific research, expanding or dying out. Co-citation analysis has long been used to systematically map and examine the network structures of research papers or patents and to isolate and identify the structure of scientific disciplines [9,20]. A wide variety of methods have been used since then to cluster documents based on citations, including k-means, co-clustering [3] and EM methods. However, none of these co-citation-based clustering algorithms used for text mining find foreground clusters against a background. Below, we propose and test two new algorithms aimed at foreground/background clustering: *background k-means*, a simple modification to k-means; and *Streemer*, a more elaborate algorithm.

This paper also differs from prior work in how it *uses* the clusters. Document clusters can be used as an aid for information retrieval, or for looking at the growth of communities over time [17]. We use the clusters that we find to build predictive models of community growth.

The properties of the clusters (e.g., size, growth rate, cohesiveness, etc.) are used both as features and as values to be predicted.

1.2 Analysis framework

The paper consists of two main parts. In the first half of the paper, in Sects. 2 and 3, we present the two clustering methods, background k-means and Streemer, and find foreground clusters against a background, constituting our knowledge communities. We evaluate the clusters in terms of their coherence, using external labelings of the clustered documents. In the second half, in Sects. 4 and 5, we describe the features and the model used to predict the future growth of the clusters from their past properties.

The Streemer[1] clustering algorithm is designed to maximize the coherence of the clusters found and to do so efficiently for large and high-dimensional data sets. Coherence is defined in terms of the cluster size, separation (distance) from neighboring clusters and cluster density. After clustering the papers, we use the results to generate features and use a generalized least squares model to predict the evolution of clusters over time. Data from different time periods are clustered separately. Finally, we examine which features are significant in this prediction and therefore are indicative of successful knowledge communities.

The rest of the paper is structured as follows: Sect. 2 presents the Streemer and background k-means clustering algorithms. Section 3 describes the data and evaluates the results of the clustering. Section 4 explains the generalized least squares model used for modeling the growth of communities. Section 5 presents the results for community prediction and analyzes the features that were most important. We conclude with a review of related work in Sect. 6 and a discussion in Sect. 7.

2 Clustering

We want to cluster the data so that observations that are not assigned to a standard cluster are assigned to a diffuse cluster that represents the background. This background will surround all (or almost all) of the foreground clusters, which are embedded like islands in the background. Thus, the background cluster is not only non-convex, but also non-compact, i.e., it contains holes. Such clusters cannot be found by many widely used methods such as k-means, which finds a tessellation of the space where every piece is defined by piecewise linear boundaries. The resulting clusters are always convex, and cannot represent foreground clusters embedded in a background.

As a simple example, consider the case where the data are generated from two mixture components with similar means, but different variances. Assuming, as shown in Fig. 1 for the 1-D case, that the separation between the cluster centers is small compared to the difference between the cluster variances, we would prefer the cluster with center $\mu_1 = 0$ to be surrounded by the other cluster, in accordance with the posterior probabilities. The best that we can expect from k-means, however, is to pick a decision boundary between the means of the two distributions. This effectively assigns all negative points to the left cluster, even though the probability of them belonging to the right cluster is higher.

Additionally, k-means implicitly assumes a prior belief of equal cluster sizes, which discourages finding clusters that are very small or very big. This follows from the view of k-means as a limiting case of a Gaussian mixture model with equal priors and equal cluster

[1] StreEMer is named for streaming EM, but this paper does not describe the EM interpretation.

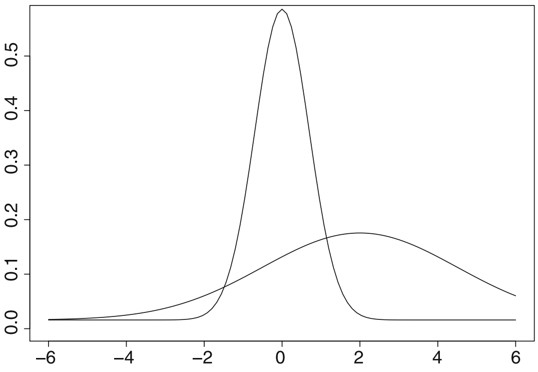


Fig. 1 Two Gaussian distributions with means $\mu_1 = 0$ and $\mu_2 = 2$ and very different variances. Cluster 1 is surrounded by cluster 2. k-means would assign the point $x = -3$ to the leftmost cluster, even though the posterior probability of the rightmost cluster is higher

variances, where k-means emerges in the limit of the variances going to zero [13]. However, when clustering documents into knowledge communities, we do not expect all of the communities to be of similar sizes. This is another argument for using some alternative clustering method, instead of k-means or a k-means variant.

2.1 Streemer algorithm

The input to *Streemer* consists of the data to be clustered, the fraction b of points in the background, and the number of clusters k. There are also some hidden parameters, which will be described later, but the result is not sensitive in their values. Additionally, Streemer requires a similarity function sim() between a data point and a cluster centroid, or between two centroids.

Streemer is a three-step algorithm that gives better results than simple streaming clustering at minimal extra computational cost. Streemer initially finds a large number of candidate clusters using streaming clustering. Then it selects k of them that are good in terms of size, density and position with respect to the other clusters. In the last step, the points are assigned to the k clusters, or to the background, if they are far enough from every cluster.

As described in detail in Fig. 2, Streemer performs streaming clustering in step 2 to generate a set of candidate clusters. Examining each point in sequence, it either adds it to an existing candidate cluster, or it creates a new cluster and assigns the point to it. This typically gives a large number of candidate clusters (a few thousand in our experiments). In steps 4–5, it selects k "good" clusters from the candidates, where "good" clusters are large and also either cohesive or isolated. Finally, in steps 6–7 it assigns each point to a cluster if it is sufficiently close to the cluster centroid, or otherwise to the background. The appropriate threshold is chosen to give the desired fraction of points in the background.

Streemer was inspired by Clustering By Committee (CBC) [16]. However, CBC has many parameters and there is no principled way to decide how to set them. The user of CBC ends up doing a blind search in the parameter space, which is time-consuming and not guaranteed to produce good results. Streemer, in contrast, has only the minimum number of parameters that are required and all the parameters have an intuitive interpretation that makes setting them easy.

Streemer is highly scalable. In its first pass over the data it performs streaming clustering, requiring time on the order of $|S|N$ operations ($|S|$ is the number of candidate clusters

```
Input: A point set $X = \{x_1, \ldots, x_N\}$, the number of clusters $k$, the fraction of points $b$
in the background, threshold $\theta$, a similarity function $sim(x, \mu_c)$ between a point $x$ and the
centroid $\mu_c$ of a cluster $c$.
Output: A partition of the points $X$ in foreground clusters $\{c_1, \ldots, c_k\}$ and a background
cluster $c_b$, with $|c_b| = bN$.
1. Initialize the set $S$ of candidate clusters to be empty.
2. For $i = 1 \ldots N$ do:
    – If $\max_{s \in S} sim(x_i, \mu_s) < \theta$, add a new cluster to $S$ containing $x_i$,
    – else add $x_i$ to $s' = \text{argmax}_{s \in S} sim(x_i, \mu_s)$ and update its centroid $\mu_{s'}$.
3. Set $minsize = (1 + |S|)/2$, where $|S|$ is the number of candidate clusters found in step 2.
   Initialize the set of final clusters $C$ to be empty.
4. For $j = 1 \ldots |S|$, let $s_j$ be the j-th candidate cluster in $S$ with centroid $\mu_{s_j}$.
   If $|s_j| > minsize$ (i.e. $s_j$ contains more than $minsize$ points), do:
    – If $\max_{s_l} sim(\mu_{s_j}, \mu_{s_l}) < \theta$ for all $s_l \neq s_j$, then add $s_j$ to $C$,
    – else compute the cluster cohesiveness:
      $D_{s_l} = \sum_{x \in s_l} sim(x, \mu_{s_l})/|s_l|$ for all $s_l$ with $sim(\mu_{s_j}, \mu_{s_l}) > \theta$ and add the most
      cohesive cluster $s' = \text{argmax}_{s_l} D_{s_l}$ to $C$.
5. If $|C| \neq k$, perform binary search for $minsize$ on the newly generated clusters, until
   $|C| = k$. (I.e., if $|C| < k$, set $minsize = (1 + minsize)/2$ and repeat step 4, similarly for
   $|C| > k$).
6. For $i = 1 \ldots N$, find the most similar cluster $\gamma_i = \text{argmax}_{c \in C} sim(x_i, \mu_c)$.
7. Set the similarity threshold $\theta_s$ such that $sim(x_i, \mu_{\gamma_i}) \leq \theta_s$ for fraction $b$ of the points.
   Assign each point $x_i$ with $sim(x_i, \mu_{\gamma_i}) > \theta_s$ to cluster $\gamma_i$; assign the remaining points to
   the background cluster $c_b$.
```

Fig. 2 The Streemer algorithm

found and N the number of data points). The part of the algorithm filtering the candidate clusters involves comparisons between themselves and is repeated approximately $\log(|S|)$ times. Finally, the last step makes one more pass over the data making kN comparisons. Overall, the complexity of Streemer is dominated by the first pass, which is that of streaming clustering [10].

By design, Streemer achieves what k-means cannot; it gives clusters without a strong bias towards equal sizes and variances. Also it can find a background cluster that can surround the foreground clusters. This background cluster is optional, in that it is possible to configure Streemer so that all the observations will be assigned to the foreground clusters. Streemer compares favorably to standard clustering methods. It is significantly faster and more efficient than EM. It is almost as fast as k-means, but it returns higher quality clusters with fewer structural restrictions. It also requires fewer (and more meaningful) parameters than CBC.

2.2 Setting parameters

A general question with any streaming clustering algorithm is how to specify a threshold for similarity (or distance). As each observation is examined, a decision is made whether to add it to an existing cluster if it similar enough, or to start a new cluster and put it there. Being a streaming algorithm itself, Streemer also requires a threshold on the similarity of items to cluster centroids. We have empirically found that the final clustering result of Streemer is not sensitive to the value of the threshold, θ. Varying θ from 0.0001 to 0.01 has almost no effect on the number of candidate clusters for a given dataset. (This, even though the computer science dataset yields about 4,500 candidate clusters while the management dataset yields

```
Input: A point set $X = \{x_1, \ldots, x_N\}$, the number of clusters $k$, the fraction of points $b$ in
the background, a similarity function $sim(x, \mu_c)$ between a point $x$ and the centroid $\mu_c$ of a
cluster $c$.
Output: A partition of the points $X$ in foreground clusters $\{c_1, \ldots, c_k\}$ and a background
cluster $c_b$, with $|c_b| = bN$.
1. Initialize the foreground cluster centroids, e.g. by random choice of points.
2. For $i = 1 \ldots N$ do:
    i.  For each foreground cluster $c$, find the similarity $sim(x_i, \mu_c)$ of $x_i$ to the centroid $\mu_c$
        of that cluster.
    ii. Pick the nearest cluster for point $x_i$:
        $\gamma_i = \mathrm{argmax}_c sim(x_i, \mu_c)$
        and store its similarity $\sigma_i = sim(x_i, \mu_{\gamma_i})$.
3. Set the similarity threshold $\theta_s$ such that $sim(x_i, \mu_{\gamma_i}) \leq \theta_s$ for fraction $b$ of the points.
   Assign each point $x_i$ with $sim(x_i, \mu_{\gamma_i}) > \theta_s$ to cluster $\gamma_i$; assign the remaining points to
   the background cluster $c_b$.
4. Re-estimate the centroids of the new foreground clusters and repeat the above procedure,
   starting from step 2, until there is no change in the assignments.
```

Fig. 3 The background *k*-means algorithm

35,000). Small differences in the number of candidate clusters have no effect on the final result, as steps 4–5 filter them out. In all of our experiments we used $\theta = 0.001$.

The threshold *minsize* is used in step 4 to select the most cohesive candidate clusters. Instead of setting it explicitly however, we perform binary search until the final number of clusters is k. Finally, in the last step Streemer assigns all points to the nearest cluster or, if no cluster is close enough, to the background. The algorithm selects the $N(1 - b)$th greatest distance and uses that as a threshold to assign points to their nearest foreground cluster or the background. Equivalently, the user could specify this threshold instead of b. In this case, our choice to specify b was motivated by some intuition on the structure of communities and a goal to understand the effect of belonging more closely to a community vs. being more dispersed.

Streemer looks very much like a streaming clustering algorithm augmented with a final step for generating a background cluster by trimming far points from the clusters. The difference is that Streemer finds a number of candidate clusters much larger than k, and uses a second step to select k of those candidates as clusters. This makes Streemer less greedy, which improves cluster quality. Additionally, in standard streaming clustering, users specify a threshold, and thus cannot specify the desired number of clusters. We find it more intuitive and useful to specify the desired number of clusters and let Streemer, by searching over the threshold values, find that number of clusters.

2.3 Background *k*-means algorithm

We also evaluated a simple modification of the *k*-means algorithm to include a background cluster (Fig. 3). We call this method *background k-means*. Like *k*-means, the first step in every iteration is to assign each point to its nearest cluster. The second step, which is new, moves to the background a pre-defined fraction of the points that have the greatest distance from the centroid of their cluster. The third step then re-estimates the cluster centroids using only the assigned points and the procedure repeats until convergence. As in *k*-means, all the foreground clusters are convex and have piecewise linear boundaries.

3 Validation of clustering methods

To test and validate the clustering methods, we used two datasets, one consisting of papers from computer science fields and the other from management. The computer science data were drawn from CiteSeer, a digital library of papers from conferences and journals in computer science. CiteSeer collects computer science papers posted on the Internet as well as by linking directly to publishers, conference sites, and journals, and then parses these articles to find the citations and descriptive information in each paper. We cross-referenced these papers with the DBLP Computer Science Bibliography, a database that indexes a similar group of computer science papers, in order to verify existing information and gather supplemental information on journals and conferences. The majority of papers in the version of these databases that we used were from between 1992 and 2003.

The management data were drawn from the Thomson ISI database. We selected 41 core journals and conference proceedings in management and collected complete sets of all articles and their citations for these 41 journals/proceedings since 1956. The list includes within the field of management both the macro (which is heavily influenced by economics and sociology) and the micro (which is heavily influenced by psychology) specialties.

The documents in both datasets were represented as boolean vectors, i.e., vectors with elements 0 or 1. The jth element of vector i is 1 if document i was cited by document j, and 0 otherwise. The vectors were L_2-normalized before clustering, to give documents with different number of citations equal weight. The computer science dataset consists of 341,458 documents, represented as boolean vectors of 197,163 dimensions. The vectors are sparse: the average number of nonzero elements (i.e. number of times a document was cited) is 5.19. The management dataset consists of 114,450 documents in 1,082,729 dimensions with 21.12 citations per document on average. We also extracted the text of the paper titles and keywords.

3.1 Validation method

We clustered our two data sets using k-means, background k-means, and Streemer and evaluated the clusters by computing two measures of how homogeneous are the class labels of the points in the clusters. As labels we used the journals and conference proceedings where the documents were published. Even though the mapping between journals/conferences and knowledge communities is not one-to-one, there is some correlation between them (for example, most researchers in computer architecture, graphics, and machine learning publish in different venues). Citeseer provides publication information, but because the annotation is automated, it contains many errors. To get more reliable publication data we mapped a sample of the papers to DBLP, which is of higher data quality. We found that the annotated documents belonged to 1,495 journals/proceedings. For the management dataset we used the titles of the 41 journals/proceedings where the documents were published.

We use two measures of cluster quality. The first is the weighted average entropy (WAE) [6]:

$$\mathrm{WAE} = \sum_{i=1}^{C} \frac{n_i}{N} E_i$$

where n_i is the number of points in cluster i, N is the total number of points and E_i is the entropy of the distribution of labels (i.e., journals or proceedings) for points in cluster i. The lower the WAE, the better the clustering matches the given data labels.

Table 1 Evaluation of the CiteSeer and management clusters for the three clustering algorithms. (fg) means that only the foreground clusters were used in the calculation. Better clusters have lower WAE and higher NMI

	k-means	Background k-means	Streemer
CiteSeer			
WAE (fg)	7.004	6.513	6.876
WAE	7.344	8.294	8.306
NMI (fg)	0.110	0.139	0.144
NMI	0.102	0.078	0.103
Management			
WAE (fg)	3.983	3.795	3.775
WAE	4.120	4.090	4.010
NMI (fg)	0.174	0.228	0.261
NMI	0.176	0.195	0.221

The second is the normalized mutual information (NMI) [22]:

$$\text{NMI} = \frac{I(Y, \hat{Y})}{\sqrt{H(Y)H(\hat{Y})}}$$

where Y and $\hat{Y}$ are random variables taking the values of the external and the cluster labels respectively. $I(Y, \hat{Y})$ is the mutual information between these two variables and $H(X)$ is the entropy of a random variable X. NMI is 1 when the clusters match the external labels exactly and 0 for a random clustering.

3.2 Validation results and discussion

In the experiments we compared k-means, background k-means and Streemer using the cosine as the similarity function. We used $b = 0.67$ for the computer science dataset and $b = 0.45$ for the management dataset. θ was set to 0.001 and k to 22 for both datasets. These values were deemed reasonable, based on our experience with the fields. In spite of doing an extensive search over the user-specified values, we were unable to get CBC to give anything close to the desired foreground/background split and number of clusters, so no CBC results are given. For k-means, which does not explicitly find a background cluster, we used the biggest of the clusters as background. Fortunately, for both datasets one cluster was much larger and heterogeneous than the rest. The largest k-means cluster for the computer science dataset contained 21% of the documents and for management 39%; all of the other clusters were approximately of equal sizes in both cases.

The evaluation of the clusters appears in Table 1. In terms of the average entropy for all the clusters (including background), k-means is either better or comparable to the background k-means and Streemer. This is not surprising, as the background cluster found by background k-means and Streemer is both big and of low cohesiveness by construction. Therefore its entropy is quite high and, coupled with the big size, it contributes significantly in the overall weighted entropy. The reason that background k-means and Streemer are worse for the computer science case but about the same for management is that the background cluster was bigger in computer science (67%) than in management (45%) and thus had a larger effect on the overall entropy. In terms of NMI, k-means was about the same as background k-means and Streemer for the CiteSeer data and worse for management.

A more relevant measure for the performance of the algorithms for our purposes is the average entropy and NMI of the foreground clusters only, denoted with "(fg)" in the table. Streemer is always best for NMI (fg), but background k-means is sometimes competitive for WAE (fg). Excluding the large, diffuse background cluster from the calculation can only improve the entropy; thus both background k-means and Streemer perform better than k-means for both WAE and NMI. As hoped, including a background cluster always leads to better foreground clusters.

4 Clustering over time

Most current popular clustering algorithms assume that clusters over time are static—that is, that all clusters exist at the beginning and end of the time period under consideration and that no new ones are formed in the interim [1,25]. This is a tolerable simplifying assumption for short periods in stable environments but more troubling for data over time in a dynamic environment where we must consider emerging clusters, merging clusters, and dying clusters. The cluster structure of computer science documents in 1975 is very different from that of 1995.

To address this, we developed an iterative clustering scheme ("rolling clustering") that successfully resolves this temporal confounding. We built an iterated "overlapping" clustering methodology into our algorithm that re-clusters in overlapping 5-year blocks, stepping forward by 1 year at a time. Therefore the elements in year 1990 would be clustered based only on the elements in 1985 to 1990, papers in year 1991 would be clustered based only on 1986 to 1991, and so on. We then match up clusters over time using the overlapping years of the two clustering runs (in the case above, 1986–1990). Our clustering algorithm looks only backward in time to determine clusters in a given year; "linking" is only a done after the fact. The temporal overlap ensures some consistency in cluster composition, while allowing new clusters to be created and existing clusters to merge or wither away.

Figure 4 graphically displays a hypothetical series of time windows. Clusters A and C appear throughout the three time windows. Cluster B disappears in the third time window. Cluster D appears in the second time window. Throughout the time windows the slow dynamic movement of the clusters as they change over time is visible. Essentially, we have chained together a series of overlapping clusterings so that we can create continuity while allowing for an evolving knowledge landscape.

This process allows all cluster assignments in each year to be backward-looking only, based on the previous 5-year frame—an appropriate "context" for knowledge development. At the

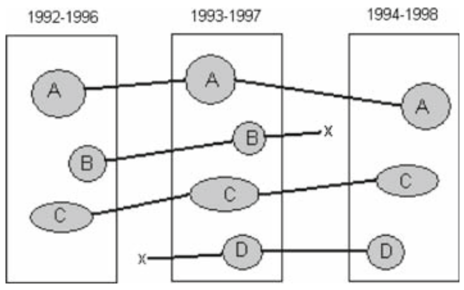


Fig. 4 Illustration of rolling clustering

same time we find very high continuity between clusters, since the knowledge landscape we created changes gradually. Another benefit of this method is that our measures of "centrality" at the paper and cluster levels refer to the appropriate frame rather than an aggregate over the entire time range, as with all other standard methods.

To do valid prediction of the success of knowledge community or of a paper, we need to use only clusters (and cluster features) based on earlier data. Our process assures that clusters at a given time only depend on prior history. The clusters capture what agents looking at that intellectual landscape would see at that time (i.e., in real time without the benefits of hindsight).

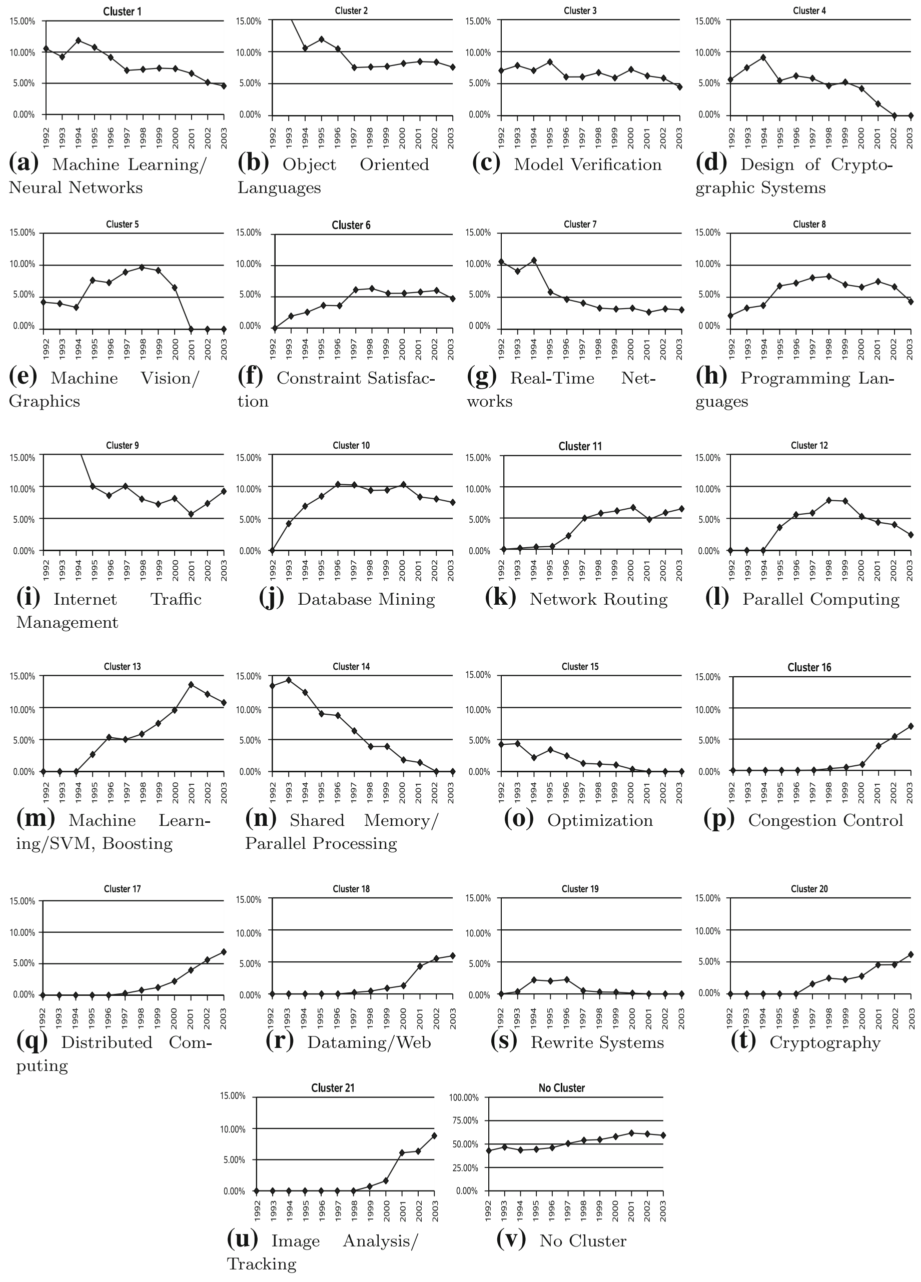
4.1 Analysis of clusters

The evolution of clusters over time can be seen as a result of the choices agents make as they, in aggregate, position themselves on this landscape. The year 1992–2003 marked a dramatic growth of computer science and radical evolution of its scope of use. To further test the validity of our method we look at the knowledge communities we found and see if they accurately reflect the changes in computer science during this period. Figure 5 and Table 2 give details on the 21 knowledge communities we identified. Figure 5 provides graphs of the growth of each knowledge community over time, as well as a sense of each community's appearances and disappearances. Table 2 details our proposed names for each knowledge community and also provides some details about them.

In 1992 we found 14 knowledge communities. Between 1992 and 1999 seven new knowledge communities formed and none disappeared. From 1999 to 2001 five knowledge communities disappeared and none were created. This finding is in keeping with the dramatic growth of computer science in the Internet boom and the subsequent collapse of the Internet bubble. The movement and rates of change of clusters also reflect these changes, with more activity during times of shake-up in 2000–2001 as knowledge communities collectively struggle to readjust to and survive in a period of dramatic correction in the sector.

The insights from literature on paradigm shifts and disruptive technologies can be seen in microcosm here. For example, clusters 5 and 21 are both on very similar topics—"machine vision/graphics" and "image analysis/tracking", respectively—but are very distinct communities. In the mid-1990s the first experienced a steep decline as a research community while the latter emerged from nowhere and became quite significant. Clusters 4 and 20 on "design of cryptographic systems" and "cryptography", respectively, experience the same pattern, with cluster 20 seeming to emerge and grab cluster 4's intellectual space. These seem to be examples of established communities of researchers being unable to absorb or compete with emerging research communities. During the mid-1990s as the Internet grew exponentially and broadband allowed video to be more easily transferred and stored, we also saw the emergence of two clusters on "congestion control" and "image analysis/tracking". At the same time we saw the decline of "distributed computing" and "shared memory/parallel processing". These trends seem to fit the intuitive understanding of trends in computer science.

Other clusters merge and split, potentially representing a schism of a knowledge community or the absorption of one knowledge community by another. Our dynamic clustering methodology allows some insight into how knowledge evolves and changes. For example, in 1996 the cluster representing the knowledge community researching "Internet traffic management," shattered to form several smaller clusters. Most fragments were below the threshold of size and cohesiveness to be knowledge communities, but there did remain remnants of the original cluster and a new cluster which represents "distributed computing". Both are signif-



(a) Machine Learning/ Neural Networks **(b)** Object Oriented Languages **(c)** Model Verification **(d)** Design of Cryptographic Systems

(e) Machine Vision/ Graphics **(f)** Constraint Satisfaction **(g)** Real-Time Networks **(h)** Programming Languages

(i) Internet Traffic Management **(j)** Database Mining **(k)** Network Routing **(l)** Parallel Computing

(m) Machine Learning/SVM, Boosting **(n)** Shared Memory/ Parallel Processing **(o)** Optimization **(p)** Congestion Control

(q) Distributed Computing **(r)** Dataming/Web **(s)** Rewrite Systems **(t)** Cryptography

(u) Image Analysis/ Tracking **(v)** No Cluster

Fig. 5 Cluster evolution by year from 1993 to 2003 for clusters 1–22 (as % of in-cluster papers)

icantly smaller—38 and 16%, respectively, of the size of the original cluster. In 2000 there was also a merger between the clusters representing the knowledge communities researching “constraint satisfaction” and “optimization”, respectively. These fields are clearly related, and both clusters were approximately the same size.

Table 2 Cluster descriptions

Cl.	Size	Proposed name	Five most common words (frequency)
1	7892	Machine learning/neural networks	Learn (1390), Network (691), Robot (649), Neural (606), Model (506)
2	9368	Object oriented languages	Type (835), Object (821), Program (800), System (709), Language (624)
3	7022	Model verification	System (1267), Time (826), Model (783), Verification (435), Specification (434)
4	4144	Design of crypto-graphic systems	System (557), Distribute (524), Protocol (279), Base (222), Fault (220)
5	6053	Machine vision/graphics	Image (717), Model (506), Base(483), Recognition (350), Motion (327)
6	6070	Constraint satisfaction	Model (471), Constraint (425), System (398), Base (389), Algorithm (380)
7	3968	Real time networks	Time (1146), Real (908), System (731), Schedule (638), Network (302)
8	7743	Programming languages	Logic (756), Program (700), System (595), Proof (485), Type (445)
9	9002	Internet traffic management	System (1253), Distribute (743), Network (651), Mobil (474), Perform (461)
10	10180	Database mining	Data (887), Queries (886), System (777), Base (767), Database (762)
11	5890	Network routing	Network (1060), Multicast (750), Service (444), Base (407), Protocol (404)
12	5990	Parallel computing	Parallel (1212), Perform (553), Distri-Computing (524), bute (533), System (458)
13	9566	Machine learning/SVM, boosting	Learn (1226), Model (911), Network (664), Base (597), Data (543)
14	3818	Shared memory/parallel processing	Parallel (479), Memory (466), Cache (297), Perform (281), Share (255)
15	950	Optimization	Algorithm (134), Genet (104), Problem (64), Optimization (57), Network (56)
16	2297	Congestion control	Network (514), Tcp (347), Control (324), Service (273), Congest (194)
17	2743	Distributed computing	Network (379), Web (352), Traffic (253), Cache (213), Service (196)
18	2428	Datamining/web	Mine (366), Data (342), Web (229), Base (206), Algorithm (197)
19	472	Rewrite systems	Rewrite (44), System (43), Program (42), Constraint (36), Logic (31)
20	3289	Cryptography	Secure (413), Key (238), Protocol (200), Computing (195), Scheme (184)
21	3043	Image analysis/tracking	Base (304), Image (298), Model (290), Recognition (245), Track (216)

We also see the emergence of a number of clusters that were not present at the start of our study. In 1996 a new cluster emerged on "Datamining/Web". One of its top three most cited papers is by Larry Page and Sergey Brin, the founders of Google. In 1996 the knowledge community representing “Datamining/Web” comprised only 0.23% of our computer science papers—in 2003 it represents 7.20%. Our main goal, however, is not to give us insight into what happened historically, but to predict what will happen going forward in an area of research and to define the attributes of knowledge communities that lead to their differential success. For the rest of the paper we analyze the computer science dataset. The same analyses on the management dataset gave qualitatively similar results [24].

5 Predicting knowledge community growth

In this section, we use the foreground and background clusters found previously as the basis for analyzing the growth of knowledge communities. We ask how the knowledge content, as measured by their citations, and the rhetorical content, as measured by the vocabulary they used, affect the success of these communities. We also examine the effect of community characteristics of cohesion, uniqueness, and adaptability, as described below.

5.1 Model for community growth

Our goal is to investigate the significance of a number of factors in explaining the success of a knowledge community. We look at attributes such as the cohesiveness and uniqueness of their vocabulary and the knowledge they draw on. Our dependent (predicted) variable is a measure of the *vigor* or *performance* of a community/cluster at a given time, as measured by the number of papers presented at conferences or published in journals in a cluster each year.

We used a generalized least squares (GLS) model, including robust standard errors for determining statistical significance, which allowed us to investigate the time trends within our data while also adjusting the standard errors for intra-group correlations. This is necessary because we believe the performance measures of any cluster will be correlated over time. Since the knowledge communities were clustered based on their similarity of citations, larger communities will tend to contain more diverse citations. We included a 1-year lag in the regression as well, thus controlling for the size of the cluster the previous year; this means that the results presented below are *not* due to community size.

More formally, we estimate our model as follows:

$$y_{it} = \beta x_{it} + b_i z_{it} + e_{it}$$

where i indexes the clusters, t indexes years (time), y_{it} denotes the number of papers in a cluster presented or published each year, x_{it} is the vector of features with corresponding coefficients β capturing effects that are the same across all clusters (the fixed effects), z_{it} the features with coefficients b_i which capture the variation across clusters (the random effects), and e_{it} represents the error term.

We describe below the features that we used: cohesiveness, uniqueness, flexibility, leadership/coordination controls, prestige controls and industry/academia controls.

Cohesiveness: We are interested in seeing if the intellectual “cohesiveness” of both the shared knowledge (papers cited) and shared rhetoric (words) of the knowledge community are significant for predicting its performance. Knowledge cohesiveness represents how widely (or narrowly) authors in the cluster searched for knowledge during that year.[2] It was computed as the average similarity between the citations of each paper and the overall citations

[2] Since clusters vary year to year, cohesiveness is for a given year.

of the cluster:

$$\sum_{i=1}^{n_c} \frac{\text{sim}\,(x_i, \mu_c)}{n_c}$$

where μ_c represents the centroid of cluster c for a given year; n_c is the number of papers in cluster c, x_i is the ith paper in c and sim() is the measure of similarity, which in our experiments was the cosine function. Rhetorical cohesiveness measures how similarly authors in a cluster use vocabulary. It was computed as the similarity of the stemmed words in the title and keywords of each paper to the average for its cluster. As is common, stop words were removed.

Uniqueness: We are also interested in how different an intellectual community is, either in the knowledge it generates or in the rhetoric (vocabulary) it uses, from other intellectual communities. Uniqueness of rhetoric represents how different the vocabulary of a knowledge community is at a given point in time compared to other clusters. Similarly, uniqueness of knowledge measures how different the sources of knowledge of a knowledge community are at a given point in time. For this feature we compare the average citations or vocabulary for a cluster to all other clusters' average citations or vocabulary. For example, if a cluster generally uses the same keywords or cites the same papers, it will have a low "uniqueness."

Flexibility: Flexible, or adaptable, clusters are defined as those that change more over time relative to other clusters. Change in vocabulary is measured by how much the word usage in a cluster changes from one year to the next; change in knowledge is measured by how much a cluster's average use of citations changes. In both cases, we compute the cosine similarity between the 3-year running averages of the citation structure and language centroids for each cluster and itself in the previous year.

Leadership/coordination controls: We test for the effects of leadership (or coordination) on three levels—from members of the community, for concentration of the institutions the members identify with, and for concentration in the venues the community publishes in. Space precludes a full description of these features; for details, see [24].

Prestige controls: We wish to test whether prestige is a powerful factor in explaining differential performance in knowledge communities. We control for prestige on the member, journal/conference, and employer/university levels. For members we wish to control for the prestige that would result from the "top" members of a field preferring to publish in some knowledge communities, leading to superior performance. We constructed a variable by counting the number of authors who were fellows of IEEE, ACM or the National Academy of Engineering and published in any of our communities. Next we constructed a variable that counted the number of papers coming from the 20 most prestigious universities in computer science as ranked by the US News and World Report graduate school rankings. By doing this we control for the tendency of some communities to be associated with prestigious institutions. Lastly, we constructed a variable that counted the number of papers published in the top ten most prestigious journals as ranked by impact factor in Thomson ISI's Impact Factors and the top ten most prestigious conferences, as ranked by citation impact by DBLP. We ranked these counts within year by cluster. The ranked list of clusters by year indicated the relative prestige of knowledge communities on multiple levels.

Industry/academia controls: Each author of each paper is coded as being affiliated with a firm or academic/research institution. We then code each paper as "academic" if all of its authors are affiliated with academic/research institutions, "industry" if all of its authors have firm affiliations, and "mixed" if some of its authors are affiliated with firms and some with

Table 3 Time series GLS Estimation (The dependent variable is the number of papers published by a community in a given year)

Cohesiveness	
Knowledge	−1.032**
Rhetoric	1.169**
Uniqueness	
Knowledge	−4.040*
Rhetoric	1.494***
Flexibility	
Knowledge	0.293*
Rhetoric	−0.272*
Control variables	
Lagged response	
One year	0.557***
Leadership controls	
Journal leadership	−3.240
School leadership	−0.394
Member leadership (eigenvector)	−0.004*
Prestige controls	
Journal prestige	−0.002
School prestige	−0.019***
Member prestige	0.011**
Industry/academy affiliation controls	
Pure industry affiliation	0.599*
Mixed industry/academy affiliation	−0.858*
Constant	0.108
N	231,000
χ^2	2,213.746
R^2	0.835

(*** $p < 0.001$; ** $p < 0.01$; * $p < 0.05$)

academic/research institutions. We entered this information into the regression by including the two categorical variables “mixed” and “industry”.

We arrive at our full model by analyzing variables systematically to examine their marginal effects as well as the end joint effects.

6 Community prediction results

Our model examines the effects of all of the features discussed above on knowledge community performance (Table 3). We find that cohesive rhetoric (low variance of vocabulary within a cluster) is associated with improved performance, while a broad use of knowledge (high variance of citations within a cluster) maximizes performance. Also, a knowledge community maximizes performance when it uses vocabulary that is similar to that of other clusters, while knowledge, as represented by citations, that is gathered from diverse sources predicts superior community performance. Community flexibility in citations predicts community growth, while changing vocabulary has the opposite effect (significant at $p < 0.05$).

Examining the coefficients of industry affiliation, we see that community performance is enhanced by a high percentage of purely industry-affiliated papers. On the other hand, a higher percentage of mixed industry-affiliated papers indicated a negative impact on community performance ($p < 0.05$). Clusters with higher proportions of purely industry-affiliated papers were associated with higher performance than clusters with elevated proportions of either purely academic or mixed-affiliation clusters, but the direction of causality is unclear.

Since our hypotheses examine the use of citations and rhetoric for the same three measures, we also examined the correlation between rhetoric and citation structures for each pair of similar variables. There was a significant, positive relationship between citation and rhetoric measures for all three knowledge community measures. It is not surprising that use of language and citations are related to each other, since authors are citing papers they learned from. On the other hand, they are not identical—a paper, while it relies on citations, is not only a function of them. Analysis shows that all the effects described above are unchanged, and remain statistically significant when correlation is taken into account.

In summary, successful knowledge communities have systematic characteristics. They use knowledge and rhetoric in diametrically opposite ways:

– Successful use of *knowledge* means using broad, rapidly repositioned and community-specific knowledge.
– Successful use of *rhetoric* means using narrow, unchanging language, which is common to many communities.

We also ran the same analysis on the management data set and found consistent results [24].

6.1 Discussion

We focus on the way knowledge communities use knowledge and rhetoric to help explain why some of these knowledge communities flourish and grow. We find that the patterns for knowledge and rhetoric use are very different. A broad-searching, far-ranging, and flexible use of knowledge maximizes community performance, while a shared, common, and stable rhetoric is most beneficial to community performance. We did not find support for the proposition that the use of unique knowledge benefits knowledge communities. Increased work by authors associated with firms had an overall positive effect on knowledge community performance, but an increase in work done jointly by researchers from firms and academic institutions led to an overall negative effect on knowledge community performance.

How do these characteristics lead to the functioning of knowledge communities? We speculate that in situations of large-scale collaboration and low coordination, a shared technical language helps minimize the cost and complexity of communication. Using a unified and consistent vocabulary, we believe, allows researchers to more efficiently exchange ideas and collaborate. Using terms that other communities know, makes it easier to be understood by these communities, too.

The search/positioning literature (S/P) for knowledge development [14] suggests that there is a tradeoff between exploration (searching for new ideas as measured by citing papers in diverse communities) and exploitation (making contributions based on papers central to one's own community). The data here indicate that knowledge communities which search broadly and remain intellectually nimble perform best. Particularly in the face of very diverse ideas, expressing these innovations in a unified rhetorical and intellectual framework allows many ideas to be absorbed by a successful community and translated into a unified, efficient and shared rhetorical framework.

7 Related work

Clustering has been used extensively for text mining [21]. In many cases it is based on the actual words of the text, but it has also been used to cluster documents based on their citation structure. Citation (and co-citation) analysis is quite old [23]. More recent research uses newer clustering algorithms, including spectral [4], information theoretic [3] or Bayesian models. Latent Dirichlet allocation (LDA) is also increasingly used to cluster documents, including their evolution over time [1]. Other papers deal with the similar problem of clustering web pages, where the links take the role of citations [7,8].

A well-known algorithm for clustering large datasets is BIRCH [26]. Streemer falls into the category of distance-based methods, as defined in [26]. By going over the points twice and finding a background cluster, Streemer avoids the problems mentioned there of frequently scanning the points and of treating them all equally. Much like BIRCH, Streemer first finds a large number of clusters which it later reduces. The difference is that Streemer does not employ an external clustering algorithm to do that. In [26] the authors use agglomerative hierarchical clustering. Additionally, Streemer can use any provided distance function, as opposed to BIRCH, which is limited to distances that can be computed by its "clustering feature" (CF) vectors.

Another algorithm with many similarities to Streemer is DBSCAN [5]. It finds clusters by repeatedly adding points that are close together and have a large number of neighbors. DBSCAN can find clusters of arbitrary shapes, but it requires the specification by the user of the parameters *Eps* and *MinPts* and is very sensitive to their values. Streemer also requires similar parameters, but we found that it is not sensitive to them. Furthermore, DBSCAN can suffer from robustness problems, because it operates on the whole set of points and does not do some form of preliminary clustering. If there is a string of points connecting two clusters, DBSCAN will merge the clusters. Streemer, on the other hand, first finds candidate clusters and then only merges them if the resulting cluster is highly cohesive.

A detailed analysis of streaming clustering algorithms appears in [10]. The paper examines several variations, such as sampling and finding first a large number of clusters which are then clustered down to the requested number k, which is reminiscent of Streemer's second step. The idea of cluster cohesiveness in Streemer was inspired by the clustering by committee (CBC) algorithm [16]. CBC finds a set of cohesive clusters, called committees, that are well separated and which initially include only a subset of the points. The algorithm proceeds by assigning points to their most similar committee.

Trying to identify communities or clusters and how they evolve with time has, of course, been studied in the past. This paper is unique in using clusters to build predictive models of how communities evolve, and how location in a cluster or in the background predicts how widely cited a paper will become. Agglomerative clustering is used in [11] to find co-citation communities that are strong and others that are essentially random, but they do not specifically use the notion of background in their clustering, and do not use their clusters in predictive models. Similarly, in [17] the authors search for temporal trends in hyper-linked document databases using clustering, but do not build predictive models of community growth. The authors in [15] identify research communities (and make paper acceptance predictions) from the citation patterns and text of papers, using relational learning techniques, but do not study the communities themselves.

This paper also differs from the prior work in that we cluster using foreground and background clusters, an idea that has been used in vision [12,18], but is rare when looking at documents. A common alternative algorithm with a similar property is a Gaussian mixture model with non-uniform variances. Estimating such a mixture model provides soft assignments of items to clusters, but is significantly more computationally demanding than Streemer or background *k*-means.

8 Conclusions

We have presented a new clustering algorithm, Streemer, with several useful characteristics. Streemer finds dense foreground clusters embedded in a more diffuse background cluster. It requires only two passes through the data, and unlike *k*-means or its variant, background *k*-means, Streemer is particularly good at avoiding local minima, and is able to find clusters of widely divergent sizes. This is useful, among other applications, for clustering documents into scientific communities.

Using a model in which many papers are not part of clusters (but rather fall in a background cluster) gives cleaner foreground clusters and allows important insights to be made. Knowledge communities, as defined by our clusters, produce a disproportionate amount of the knowledge in computer science. In our dataset of computer science publications in technical journals and conferences, 57% of citations are received by papers in clusters even though only 44% of papers are in clusters. Even more dramatically, 76% of citations of papers in a cluster refer to another paper in a cluster. On the other hand, papers not in a cluster cite almost proportionately to the ratio of papers in and out of clusters, with 41% of citations going to the 44% papers in a cluster and 59% of citations going to the 56% papers in the background.

We fitted a model to predict the evolution of the knowledge communities that were found from clustering. The coefficients of the features for the fitted model describe the properties of successful knowledge communities. We found that in order for a community to grow, its members should use broad, flexible and unique information in their publications. At the same time the vocabulary they use should be narrow, unchanging and common. This perhaps is not surprising; introducing new terms and definitions at the same time as novel ideas can make it hard for the readers to absorb such a large amount of information at once. New jargon obfuscates the content and limits its spreading. Further research into the differential success of knowledge communities can provide a better understanding of what guides the development and direction of innovation.

Acknowledgments Vasileios Kandylas was supported in part by the Greek State Scholarship Foundation (IKY).

References

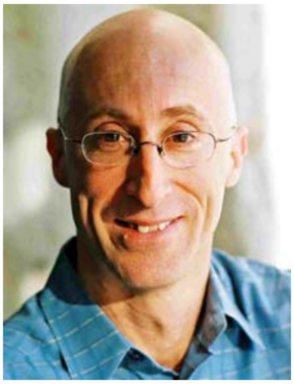
1. Blei D, Lafferty J (2006) Dynamic topic models. 23rd ICML, 113–120
2. Crane D (1972) Invisible colleges: diffusion of knowledge in scientific communities. University of Chicago Press
3. Dhillon I, Guan Y (2003) Information theoretic clustering of sparse cooccurrence data. ICDM 517–520
4. Dhillon IS (2001) Co-clustering documents and words using bipartite spectral graph partitioning. KDD, pp 269–274, ACM Press, New York
5. Ester M, Kriegel H, Sander J, Xu X (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. KDD, AAAI Press, Portland, OR, pp 226–231
6. Fern X, Brodley C (2003) Random projection for high dimensional data clustering: a cluster ensemble approach. ICML, pp 186–193
7. Flake G, Lawrence S, Giles C (2000) Efficient identification of Web communities. KDD pp 150–160
8. Gibson D, Kleinberg J, Raghavan P (1998) Inferring web communities from link topology. ACM Press, New York

9. Griffith B, Small H, Stonehill J, Dey S (1974) The structure of scientific literatures II: toward a macro- and microstructure for Science. Sci Studies 4(4):339–365
10. Guha S, Meyerson A, Mishra N, Motwani R, O'Callaghan L (2003) Clustering data streams: theory and practice. IEEE Trans Knowledge Data Eng 15(3):515–528
11. Hopcroft J, Khan O, Kulis B, Selman B (2003) Natural communities in large linked networks. KDD, pp 541–546
12. Huang Q, Dom B, Steele D, Ashley J, Niblack W (1995) Foreground/background segmentation of color images by integration of multiple cues. IEEE Int Conf Image Process 1:246–249
13. Kearns MJ, Mansour Y, Ng AY (1997) An information-theoretic analysis of hard and soft assignment methods for clustering. UAI, pp 282–293
14. McGann A (2002) The advantages of ideological cohesion a model of constituency representation and electoral competition in multi-party democracies. J Theor Politics 14(1):37–70
15. McGovern A, Friedland L, Hay M, Gallagher B, Fast A, Neville J, Jensen D (2003) Exploiting relational structure to understand publication patterns in high-energy physics. SIGKDD Explor Newslett 5(2):165–172
16. Pantel P, Lin D (2002) Document clustering with committees. SIGIR '02, ACM Press, New York, pp 199–206
17. Popescul A, Flake G, Lawrence S, Ungar L, Giles C (2000) Clustering and identifying temporal trends in document databases. Advances in digital libraries, 2000. ADL 2000. proceedings. IEEE, pp 173–182
18. Savakis A (1998) Adaptive document image thresholding using foreground and background clustering. Proceedings of international conference on image processing ICIP98
19. Small H (2003) Paradigms, citations, and maps of science: a personal history. J Am Soc Informat Sci Technol 54(5):394–399
20. Small H, Crane D (1979) Specialties and disciplines in science and social science: an examination of their structure using citation indexes. Scientometrics 1(5):445–461
21. Steinbach M, Karypis G, Kumar V (2000) A comparison of document clustering techniques. KDD workshop text mining 34:35
22. Strehl A, Ghosh J (2002) Cluster ensembles—a knowledge reuse framework for combining multiple partitions. JMLR 3:583–617
23. Sullivan D, White DH, Barboni EJ (1977) Co-citation analyses of science: an evaluation. Social Studies Sci 7(2):223–240
24. Upham SP (2006) Communities of innovation. PhD thesis, University of Pennsylvania
25. Wang X, McCallum A (2006) Topics over time: a non-Markov continuous-time model of topical trends. KDD, pp 424–433
26. Zhang T, Ramakrishnan R, Livny M (1996) BIRCH: an efficient data clustering method for very large databases. Proceedings of the 1996 ACM SIGMOD international conference on Management of data, pp 103–114

Author Biographies

Vasileios Kandylas received a B.E. degree from Aristotle University of Thessaloniki, Greece and an M.S. degree from University of Pennsylvania, USA. He is currently a graduate student at the Department of Computer and Information Science, University of Pennsylvania. His research interests include clustering, data mining and machine learning.

S. Phineas Upham earned his PhD from the Wharton School at the University of Pennsylvania and is currently a visiting scholar at Columbia University. He also holds an MBA from the Wharton School and a BA with honours from Harvard University. In addition to journal articles, he has edited three books published by academic and popular presses. His interests are in the study of schools of thought, paradigm development, the evolution of technical and organisational knowledge, and in the social construction of meaning.



Dr. Lyle H. Ungar is an Associate Professor of Computer and Information Science (CIS) at the University of Pennsylvania. He also holds appointments in several other departments in the Engineering, Medicine, and Wharton Schools. Dr. Ungar received a B.S. from Stanford University and a Ph.D. from M.I.T. He has published over 100 articles, and is co-inventor on nine patents. His current research interests include machine learning, data and text mining, and bioinformatics.