ONLINE CLUSTERING AND CITATION ANALYSIS USING STREEMER

Vasileios Kandylas

A DISSERTATION

in

Computer and Information Science

Presented to the Faculties of the University of Pennsylvania

in Partial Fulfillment of the Requirements for the

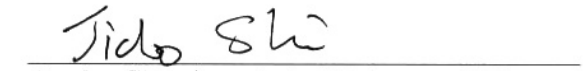
Degree of Doctor of Philosophy

2009

Supervisor of Dissertation

Lyle Ungar, Associate Professor, CIS

Graduate Group Chairperson



Jianbo Shi, Associate Professor, CIS

Dissertation Committee

Mitch Marcus, RCA Professor of Artificial Intelligence, CIS

David Blei, Assistant Professor, CS, Princeton University

Shane Jensen, Assistant Professor, Statistics

Ben Taskar, Assistant Professor, CIS

Acknowledgements

Even though this thesis bears my name, a lot of people have contributed, directly or indirectly, over the course of my graduate studies. First and foremost is my academic advisor, Dr Lyle Ungar. His supervision, guidance and ideas, his support, encouragement and motivation over the years were a sufficient and, more importantly, a necessary condition for completing my thesis. I would not have made it without his help and I owe him my deepest gratitude. Second only to my adviser are my collaborators. The work we did together has become a big part of the thesis. Dr Shane Jensen, who is also a member of my dissertation committee, helped me in presenting and clarifying the information in many chapters and contributed in the work of chapter 4. Dr Phineas Upham helped me a lot in gathering the data and analyzing the clusters in chapter 2. Ted Sandler (who is also to join the league of doctors in a few months - good luck in your defense Ted!) helped in chapter 4 and in many other places too numerous to count. And of course the members of my thesis committee, Doctors Mitch Marcus, Ben Taskar, David Blei and Shane Jensen whose insightful comments and suggestions helped make this dissertation so much better.

There are many more people who were not directly involved in the work presented in this thesis, but who were indirectly responsible for its completion. First, the faculty of the department of Computer and Information Science in the University of Pennsylvania. They accepted my application, taught me a lot about Computer Science and provided an environment with plenty of opportunities to improve, to the point that I could fulfill the requirements to become a Doctor of Philosophy.

I am grateful that they were willing to bet on my success as a graduate student. Second, the Greek State Scholarship Foundation (IKY) for providing partial financial support during my studies. It was an honor for me to receive that scholarship. Third, the many friends from within and without the department: George Fainekos, Ted Sandler, Stephen Tse, Michalis Kizirogloy, George Chrysochoides, Rosskyn D'Souza, Arvind Bhusnurmath, Dongyu Lin, Kilian Weinberger, Partha Talukdar, Axel Bernal, Abhishek Gupta, John Blitzer, Nikhil Dinesh, Dimitris Vytiniotis, Nate Foster. They are credited with providing encouragement, fun, stimulating conversations, a friendly environment, help, advice and generally making my life as a graduate student so much easier. I am very fortunate to have them as friends.

Finally, I would like to thank my parents and my brother who, from the sidelines, have been cheering me on and encouraging me. They have helped in more ways than I can express.

ABSTRACT

ONLINE CLUSTERING AND CITATION ANALYSIS USING STREEMER

Vasileios Kandylas

Lyle Ungar

Clustering algorithms can be viewed as following an algorithmic or a probabilistic approach. Algorithmic methods such as k-means or streaming clustering are fast and simple but tend to be *ad hoc* and hence hard to customize to particular problems, whereas the probabilistic methods are more flexible, but slower. In this work we propose online algorithms which combine the advantages of the two classes of approaches giving fast, scalable clustering, while allowing more flexible models of the data, such as foreground clusters interspersed within a diffuse background. These clusters are shown to be useful in modeling scientific citations.

We start the thesis by giving a non-probabilistic, few-pass algorithm, called Streemer. Streemer uses thresholds on similarities between points to find a large number of clusters on the first pass over the data. It then merges them to find larger and more cohesive clusters. In a final pass it assigns points to the clusters or to a diffuse background. Streemer avoids the standard k-means assumptions that clusters are of similar sizes. We also discuss the nature of the objective function that Streemer optimizes through its several steps and heuristics. At a cursory glance, Streemer appears to be an *ad hoc* algorithm, but in a subsequent chapter we develop a principled algorithm that emulates Streemer's steps and we make the connection between Streemer and online Dirichlet Process Mixture Models.

We use Streemer to cluster documents based on the documents they cite and find "knowledge communities" of authors that build on each other's work. The evolution over time of these clusters gives us insight into their growth or shrinkage. We also build predictive models with features based on the citation structure, the vocabulary of the papers, and the affiliations and prestige of the authors and use these models to study the drivers of community growth and the predictors of how widely a paper

will be cited. The analysis shows that scientific knowledge communities tend to grow more rapidly if their publications build on diverse information and use narrow vocabulary and that papers that lie on the periphery of a community have the highest impact, while those not in any community have the lowest impact.

We also present a probabilistic mixture model with a Dirichlet Process prior and Gaussian component distributions. This model allows for variable cluster numbers and sizes. We show how to use this model for clustering in an online fashion and also propose a two-pass algorithm, where the first pass clusters points in many clusters and the second pass clusters the output of the first pass. With the exception of foreground/background clustering, the model with the two-pass algorithm corresponds closely to Streemer.

Finally, we present an EM-based clustering method that can simultaneously cluster two or more variables using one or more tables of co-occurrence data. One application of this *multi-way* clustering algorithm is for constructing or augmenting ontologies. We test our algorithm by simultaneously clustering verbs and nouns using both verb-noun and noun-noun co-occurrence pairs. This strategy provides greater coverage of words than using either set of pairs alone, since not all words appear in both datasets. We demonstrate it on data extracted from Medline and evaluate the results using MeSH and Wordnet.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Clustering is perhaps the data mining technique most widely used on large data sets. Bayesian unsupervised learning methods are also the subject of vast amounts of research in recent years. The absence of a requirement to have observed response variables makes clustering attractive, but also creates difficulties. The proper choice of a similarity or a loss function is problem-dependent and there are not specific rules other than mostly empirical observations. Nevertheless, the great usefulness of clustering is evidenced by the many uses it has found in fields as diverse as language, marketing, information retrieval and bioinformatics [59, 6, 97, 102, 23, 67, 81].

This thesis focuses on clustering for large datasets, which are becoming increasingly common. Most current methods are either fast but simple (e.g., streaming methods), or sophisticated but slow (e.g., Bayesian models estimated with Gibbs sampling). We examine ways to combine the advantages of both by leveraging the power of a probabilistic model and the speed of a streaming algorithm. We show that using a Dirichlet Process Mixture Model (DPMM) and a 2-pass streaming approach we can achieve fast clustering, comparable to online distance-based approaches [48, 129], but with performance close to a much more expensive multi-pass algorithm like EM, or Gibbs sampling. We develop an algorithm called *Streemer*

that is very similar to a 2-pass DPMM with the addition of filtering out "background observations" and apply it in the identification and analysis of the evolution of knowledge communities. Finally, we develop a probabilistic clustering method for simultaneously clustering several variables. This *multi-way clustering* model uses as input more than one data table, which has the advantageous effect of allowing the incorporation of datasets for greater coverage over the range of values of the clustered variables.

1.1 Clustering approaches

Our approach to clustering draws from both distance-based and probabilistic clustering algorithms, which we now describe.

We call "distance-based clustering" algorithms those that operate using distance or similarity functions rather than probabilistic models. The typical example of such an algorithm would be k-means [24]. These algorithms tend to be simple and very fast, even though there are others (e.g. agglomerative and spectral clustering [44, 109]) that can be much slower. The main characteristic however is the use of a distance function and the lack of an explicit probabilistic model.

The second approach that we look at is "probabilistic clustering". These algorithms are based on probabilistic models and use probabilities instead of distances. Examples of this type would be mixture models (MM) and latent Dirichlet allocation (LDA) [9]. The use of a model gives a much greater power and freedom in choosing the structure or parameters, but also makes estimation and learning more complicated.

One thing worthy of note is that the two approaches are not unrelated. Any probability density function from the exponential family corresponds to a Bregman divergence and vice versa [3]. A mixture model with homoscedastic Gaussians (and equal mixture weights) corresponds to k-means with Euclidean distances. Other

densities give rise to other distances or divergences.

Despite this connection, the probabilistic methods have more degrees of freedom than the distance-based ones. For example, for the case of mixture models, one can have flexible priors on the distribution of cluster sizes and can use different probability densities (even for different mixture components in the same model). Moreover, even the model structure is unrestricted. One is not required to use a mixture model; any kind of hierarchical model is possible. This power means that the model has the ability to capture clusters with complicated boundaries, such as clusters within clusters. Distance-based algorithms are usually much simpler. For instance, k-means can only partition the space in convex shapes with piecewise linear boundaries. The advantage of their simplicity is that the algorithms can be extremely fast and have lower memory requirements. On the other end of the spectrum, the probabilistic clustering methods are in general more powerful and accurate but also slower than the distance-based methods.

Both approaches therefore have their advantages and disadvantages. The disadvantages become more significant as the datasets become more complex and larger. For example, in some kinds of large datasets, such as web-derived collections of articles or images, not every point fits well into a cluster. Instead, it is necessary to discard some points as noisy or spurious and cluster the rest. This distinction of points as belonging in a cluster (*foreground*) or not (*background*) is another capability that is lacking from the simpler distance-based algorithms. In general, current methods that are fast are too simple to capture the complexity of the data. And more sophisticated methods are not fast enough for practical purposes. Until recently, most clustering algorithms were applied to small or medium-sized datasets, but as larger and larger datasets are being used, these restrictions become more apparent.

There are two main avenues for dealing with large datasets that both compute a fast but approximate clustering. The first is to use approximation algorithms that

iterate over all data and the second to use streaming algorithms that only process the data once (or a small number of times).

An example of an approximate clustering algorithm is to select and cluster a random subset of the data and then use the clusters to assign the rest of the data. In essence, the speedup comes from fitting a model to a small subset of the data and then using that model to approximately cluster the large dataset [49]. A different approach is to use all the data points, but speed up the steps that are slow. An example of this case is Classification EM (CEM) [14], where the E-step of EM is simplified to find only the highest probability component. CEM makes hard assignments of clusters to points and so avoids computing the normalization constant for the assignment probabilities and also storing the fractional assignment value of each observation to every cluster. The benefits of less computation and memory are however counterbalanced by lower accuracy. The hard assignments produce biased estimates [68] and are also sensitive to the initialization.

In streaming clustering algorithms, on the other hand, the data are processed without approximating any step, i.e. as usual in a batch algorithm. The time savings come from not iterating over the points repeatedly, so in some sense the clustering stops before the algorithm has reached convergence. Most proposed streaming clustering algorithms are of the distance-based variety and carry restrictions similar to k-means [49]. Streaming algorithms for probabilistic models are less common and most are ad hoc and lack solid theoretical analysis regarding their performance. This thesis focuses on streaming algorithms for probabilistic models. Additionally, it uses two-pass streaming algorithms with an important difference: the two passes are not the same as two iterations of a batch algorithm. There are differences between the two passes (for example the number of clusters found) that help improve the final clusters.

A variety of other approaches have been used for clustering. For example, spectral

clustering uses the graph of point to point distances and computes the eigendecomposition of its Laplacian [94]. Even though the shapes of clusters that it finds can be complex, it is in fact a distance-based method and it does not find a probabilistic model of the data. Additionally, the decomposition becomes prohibitive for very large datasets.

The issues of clustering become compounded when the data reside in a high-dimensional space. A common approach is to project them to lower dimensions, either by a principled dimensionality reduction (e.g. PCA [61], LLE [104]), or doing random projections and taking advantage of the Johnson-Lindenstraus lemma [34]. Other more ad-hoc approaches follow regions of high density of points to find and expand clusters (e.g. BIRCH [128]). We do not investigate the problems and solutions of high dimensionality in this thesis.

1.2 Streaming probabilistic clustering

In this thesis we are investigating ways to combine the advantages of the distance-based and the probabilistic clustering approach. In order to take advantage of the power of probabilistic methods, we use probabilistic models as the underlying structure of our algorithms. And in order to speed up learning, we draw inspiration from the streaming clustering field of distance-based clustering. Probabilistic models are usually estimated using EM or Gibbs sampling, both slow enough that are impractical for large datasets. We therefore explore streaming alternatives, as more efficient ways to cluster large datasets using a probabilistic model and suffering just a small decline in the quality of the clusters.

We propose ways to combine two main components of the clustering approaches described previously. The first component is the Dirichlet Process Mixture Model (DPMM). The mixture model is more powerful than simple distance measurements and the Dirichlet Process (DP) deals with the unknown number of clusters which

is especially important for online methods. The second component is the online computation, which is faster than Gibbs sampling or variational methods. The goal is to get algorithms that are fast and have reduced memory requirements (like distance-based algorithms), but are also more powerful (like probabilistic algorithms). The next sections give more details of the contributions of this thesis.

1.2.1 Thesis contributions

The Streemer algorithm

We develop an algorithm, Streemer, that performs foreground/background filtering. Streemer is a more principled version of CBC [77], but is formulated to not require the setting of arbitrary parameters that CBC requires. We will also show that Streemer can be viewed as a variation of a two-pass DPMM and will give a probabilistic interpretation of Streemer.

A key aspect of Streemer is that points that cannot be assigned with high likelihood to a cluster are delegated to the background. Thus Streemer finds tight foreground clusters embedded in a diffuse background [50]. The removal of points of low fit (low quality) makes the fit of the rest of the points better and the resulting (foreground) cluster more cohesive. This can facilitate interpretation of the clusters.

We apply the Streemer algorithm to the discovery of knowledge communities. We use a dataset of published papers described by the papers they cite and cluster them in groups of papers that tend to cite each other. We can thus identify knowledge communities as foreground clusters of papers which share similar citation patterns. Subsequently, we cluster the papers in overlapping time periods and analyze the evolution of the communities over time and the factors that predict their growth or shrinkage. We also analyze the features that make a paper successful in terms of its vocabulary and its location in the community.

Online mixture model

We develop an online DPMM with Gaussian component distributions. Batch (i.e. non online) DPMMs have been the topic of prior work [103], but we are only aware of online DPMMs with multinomial distributions [127].

We investigate the operation of two-pass streaming clustering. One-pass streaming clustering is sensitive to the order that the points are processed and empirically does not give very good results. We apply the algorithm twice, with different parameters in each pass. In the first pass, we find a large number of clusters. In the second pass we treat these clusters as weighted points and merge them into fewer clusters if that increases the log likelihood [17]. Two-pass clustering gives significantly better results than one-pass clustering.

We compare the two-pass DPMM with Streemer and show the similarities between the two passes of the DPMM algorithm and the first and second step of Streemer. We also discuss the equivalence between the thresholds used in Streemer and the log likelihood value in the mixture model, as well as the relation between the Dirichlet process and the creation of new clusters based on a similarity threshold in standard streaming clustering.

Multi-way clustering

The last contribution is not about streaming clustering, but rather an algorithm for simultaneously clustering several variables. Simultaneous clustering is mostly known for the case of two variables, where it is called two-way clustering, co-clustering or biclustering [67, 19]. The advantage of simultaneous clustering is that the clustering of the one variable informs about the clustering of the other. Simultaneous clustering therefore shows the local structure of the data, which can not be revealed with traditional, 1-way clustering.

Our proposed method for multi-way clustering is based on a probabilistic model,

similar in that sense to the DPMM described previously. It extends the simple mixture model by introducing more variables and creating dependencies between them using a belief network (BN)[1]. Because of the underlying model, it solves many similar problems over competitive multi-way clustering algorithms, similar to the probabilistic one-way clustering algorithms that are more powerful than the distance-based ones. So for example, our model allows clusters of different sizes and makes it easier to define probabilistic dependencies between variables. An important improvement is that it allows the combined use of two or more data sets, each describing the clustered variables using different features. This can be used to gather data for more instances of the variables and thus increase the coverage of these instances.

We test our model by simultaneously clustering verbs and nouns using both verb-noun and noun-noun co-occurrence pairs, which we extract from Medline abstracts, and we evaluate the results by mapping them to MeSH and Wordnet. Potential applications of the algorithm include information or relation extraction. For example, it can be used to extend MeSH [92], or to create an alternate ontology to MeSH, for instance one where the relations are of different types.

1.3 Structure of the thesis

The rest of the thesis is structured in the sequence that the contributions were presented in the previous section. Specifically, in chapter 2 we present Streemer and also apply it to the discovery and analysis of knowledge communities. In chapter 3 we examine the online DPMM and its variations with two passes and foreground/background. Chapter 4 is devoted to our multi-way clustering algorithm and we conclude with a discussion in chapter 5.

[1]The model we present is used in a batch setting and thus does not use the Dirichlet process prior. It is possible to introduce that prior and perform simultaneous clustering in a streaming fashion

Chapter 2

Streemer

2.1 Introduction

Our goal is to discover and analyze knowledge communities, groups of researchers that are working on similar problems and are interested in the same research areas. We ask questions such as, "What features predict when a knowledge community will grow or shrink?" and "What kind of papers are usually more or less widely cited?" To answer these questions, we first cluster documents by their citations to find the knowledge communities and hence their authors. We use a new clustering algorithm, Streemer, which, doing only a few passes over the data, finds cohesive foreground communities whose members cite the same references, embedded in a more diffuse background [63]. We find a set of clusters which evolve over time, fit supervised models and find which features of the clusters are statistically significant in predicting their future growth or the level of citations of the papers the contain. The location of individual papers in foreground and background is also used to predict how widely they will be cited.

2.1.1 Analyzing knowledge communities

A *knowledge community* [15] is a set of people doing research on the same or closely related fields who build on each other's ideas and share similar interests. They are also known as *intellectual communities*, or *schools of thought* [112]. Belonging to a knowledge community has the advantage for a researcher of making it easier to disseminate and gather new knowledge, but as we will show below, drawing too narrowly from within a single community can be disadvantageous, both for the researcher and for the community. Since new research builds on previous work, which is often in the same field, papers often cite other papers in the same community, and knowledge communities can be identified by clustering documents based on their citations.

We aim to discover knowledge communities and to understand what drives their success, both in terms of recruiting more researchers and garnering more citations. To support this analysis, we cluster papers based on the papers that they cite. As explained below, only roughly half the documents are assigned to communities; the remainder are placed in a "background" cluster. We find clusters at multiple time periods and using only historical data. Once the knowledge communities are discovered, we fit supervised models and find which features of the clusters are statistically significant in predicting their growth. Features we look at 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 the words used are to the cluster) and exogenous measures such as the fraction of authors who have industrial or academic affiliations. Citation levels of individual papers are also predicted, using features such as whether the document is close to the center of a community, on its periphery, or does not belong in a community [95].

We characterize knowledge communities along a number of dimensions, including their use of *knowledge*, measured by what papers they cite, and *rhetoric*, measured by what vocabulary they use, and how knowledge and rhetoric change over time and

across communities. We discovered that successful communities exhibit flexible use of broad *knowledge*. Instead of concentrating on a narrow area of interest, they expand to new areas and draw knowledge from other domains. On the other hand, the vocabulary they use is restricted, tends to remain unchanged and it is common across communities. Trying to explain novel work using new terms can be confusing to the readers; a common vocabulary can facilitate the presentation of new knowledge.

Similarly, the impact which a paper will have can be predicted based on its centrality to a knowledge community. We find that a paper will have more impact (as measured by citation level, i.e. the number of citations that a paper has received) if it is within a knowledge community (in a cluster) and if it is toward the intellectual periphery of a community (edge of the cluster). Papers which draw on multiple areas are the most widely cited, as they can become the starting point for a new direction in research. In a related result, creators of new knowledge have greater impact if they actively engage in multiple communities over time, but only if they focus on one or two communities at a single time.

2.1.2 Foreground/background clustering for text mining

For our analyses we clustered papers into foreground clusters or into a diffuse background. The idea of foreground and background was taken from vision, where in image segmentation there are often cohesive foreground objects in front of a more diffuse background [57, 107]. In document mining, as in this work, the foreground clusters represent cohesive communities and mostly contain documents from specific scientific areas that tend to cite the same papers. The background consists of lower density regions, containing documents that cover several different areas or that focus on topics that are not highly cited. By clustering papers into knowledge communities, we can study how the communities grow, shrink, drift, split, merge or die out. Cluster membership can also be used to predict the citation levels of papers.

There is a long tradition of finding communities in small groups of people [10,

26, 29, 11, 37]. We address a problem of much greater scale, clustering millions of papers as opposed to a few tens or hundreds. The use of co-citation analysis also goes back many years. For example it has been used to map and examine the network structures of papers or patents [46] and to isolate and identify the structure of scientific disciplines [115]. Many other methods have been developed to cluster documents based on citations; these include k-means [59], co-clustering [22] and EM methods [18]. However, these co-citation-based methods make no use of a background. Below, we describe a new algorithm, Streemer, which was designed to operate on large, high-dimensional datasets and efficiently find foreground clusters of varying sizes embedded in a background. We combine Streemer with a "rolling clustering" procedure to allow clusters to evolve over time.

Another difference of our work compared to previous papers is in the way we use the clusters we find. Common applications of document clustering use the clusters as an aid for information retrieval, for looking at the growth of communities over time [101], or for profiling researchers and finding those with a given expertise [69]. In contrast, we use them to build predictive models of community growth and paper impact. We use the properties of the clusters, such as their vocabulary and knowledge cohesiveness, as features and train supervised learning models in order to predict the change in the cluster sizes and the number of citations of papers.

2.1.3 Analysis framework

This chapter is organized as follows. First, we describe the Streemer algorithm in section 2.2. In section 2.3, we evaluate the clusters by seeing how they divide up journals under the assumption that, in a good clustering, articles in a single journal would usually end up in the same cluster. Section 2.4 explains how we analyzed the evolution of the clusters with time. Examples of clusters are presented from the field of Computer Science, exhibiting growth, shrinkage, splitting and merging over time. We then use supervised learning in section 2.5 to predict the growth of

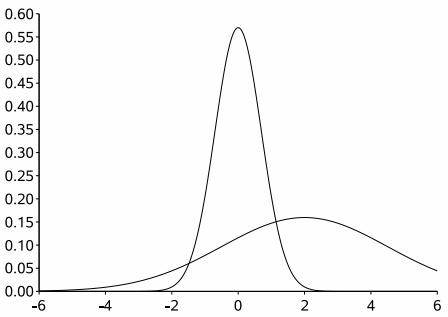


Figure 2.1: Two Gaussian distributions with means $\mu_1 = 0$ and $\mu_2 = 2$ and very different variances. Cluster 1 is surrounded by cluster 2. K-means would assign the point $x = -3$ to cluster 1, even though the posterior probability of cluster 2 is higher.

the communities that were found. We do a similar analysis for individual papers, studying how their position in a cluster affects their future citation counts in section 2.6. The chapter concludes with a discussion in section 2.7.

2.2 Clustering

2.2.1 Foreground and background

We intend to cluster papers so that they that are either assigned to a foreground cluster or to a diffuse background. The foreground clusters will be surrounded by the background, like islands in an ocean. For this to happen the background cluster cannot be compact – the foreground clusters will punch "holes" through the background. Some widely used methods cannot find clusters with these properties. For instance, k-means finds a tessellation of the space where every tile has piecewise linear boundaries.

Consider, for example, the case where the data are generated from a mixture of two 1-D Gaussians with similar means, but different variances (figure 2.1). In this example the component means are close and the variance of one is much higher than the other. In this specific example, we would prefer the cluster with center $\mu_1 = 0$ to

be surrounded by the other cluster, in accordance with the posterior probabilities. What 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 for some of them the probability of belonging to the right cluster is higher.

K-means also makes an implicit assumption about 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 variances, from which k-means emerges in the limit of the variances going to zero [66][1]. However, knowledge communities will not have similar sizes in general, making k-means a less appropriate algorithm for identifying them. This is an additional argument for using an alternative clustering method, instead of k-means or a k-means variant.

A natural generalization of k-means for foreground/background clustering would be a mixture of Gaussians with unknown variances. The ability of the components to acquire different variances and weights would allow the existence of clusters within clusters and one could use the surrounding cluster as background. However, such a model is more complicated and, for high-dimensional data, the number of parameters to fit would be prohibitive. For D dimensions, k-means requires the fitting of $O(D)$ parameters, but the Gaussian mixture would require $O(D^2)$. Additionally, the right choice of component distribution depends on the application and the Gaussian distribution is not the most appropriate for clustering documents. For other distributions there might be problems with defining a background. For example, for the Inverse Binomial distribution the notion of background is not well defined because the distribution is discrete and for the Poisson distribution it is mathematically impossible to create clusters within clusters by varying the cluster prior probabilities.

[1]This is in agreement with our empirical observations. In the experiments described in section 2.3 the cluster sizes we found using k-means and its variant algorithm, background k-means, were very similar; the size ratio of the largest to the smallest cluster was just 2.

2.2.2 Streemer

The clustering algorithm we use is Streemer [63]. Streemer[2] is designed to efficiently maximize the coherence of the clusters that it finds, making it suitable for large, high-dimensional data sets. We define coherence in terms of the size of the cluster, its separation (distance) from neighboring clusters and its density. The input to Streemer consists of the data to be clustered, the background fraction b and a parameter *minsize* that affects the minimum cluster size. (There are also two other parameters, which are mentioned in section 2.2.3, but we found that the clusters are not sensitive in their values, so they can be preset to some recommended values when using Streemer.)

In order to deal with a large number of examples, Streemer operates in a near-streaming fashion (figure 2.2), making only two passes over the data. It uses a three-step algorithm that gives better results than simple streaming at minimal extra computational cost. Streemer initially finds a large number of candidate, or seed, clusters using streaming clustering (step 2). Examining each point in sequence, it either adds it to an existing seed cluster, or it creates a new cluster and assigns the point to it. It thus finds seeds by collapsing close points into clusters. This typically gives a large number of seed clusters (a few thousand in our experiments). Then it selects from them those that are dense and not similar to each other; i.e., they have high inter-group similarity and low intra-group similarity (step 3). Finally, in steps 4 and 5, the points are assigned to the selected clusters, if they are sufficiently close, or to the background, if they are far enough from every cluster. Streemer chooses the appropriate threshold for distance from the cluster to give the desired fraction of points in the background.

One could envision re-clustering after the background points were removed. We do not do that for two reasons. First, re-clustering would make Streemer an iterative

[2]StreEMer is named for its similarity to streaming EM, but this chapter does not describe the EM interpretation.

Input: A point set $X = \{x_1, \ldots, x_N\}$, the number of clusters k, the fraction of points b in the background, threshold θ, candidate cluster minimum size $minsize$, a similarity function $\mathrm{sim}(x, \mu_c)$ between a point x and the centroid μ_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 size $|c_b| = bN$.

1. Initialize the set S of candidate clusters to be empty.
2. For $i = 1 \ldots N$ do:
   - If $\max_{s \in S} \mathrm{sim}(x_i, \mu_s) < \theta$, add a new cluster to S containing x_i,
   - else add x_i to $s' = \mathrm{argmax}_{s \in S} \mathrm{sim}(x_i, \mu_s)$ and update its centroid $\mu_{s'}$.
3. Initialize the set of final clusters C to be empty. For $j = 1 \ldots |S|$, let s_j be the j-th candidate cluster in S with centroid μ_{s_j}.
   If $|s_j| > minsize$ (i.e. s_j contains more than $minsize$ points), do:
   - If $\max_{s_l} \mathrm{sim}(\mu_{s_j}, \mu_{s_l}) < \theta$ for all $s_l \neq s_j$, then add s_j to C,
   - else compute the cluster cohesiveness:
     $D_{s_l} = \sum_{x \in s_l} \mathrm{sim}(x, \mu_{s_l}) / |s_l|$ for all s_l with $\mathrm{sim}(\mu_{s_j}, \mu_{s_l}) > \theta$ and add the most cohesive cluster $s' = \mathrm{argmax}_{s_l} D_{s_l}$ to C.
4. For each x_i, $i = 1 \ldots N$, find the most similar cluster $\gamma_i = \mathrm{argmax}_{c \in C} \mathrm{sim}(x_i, \mu_c)$.
5. Set the similarity threshold θ_s such that $\mathrm{sim}(x_i, \mu_{\gamma_i}) \leq \theta_s$ for fraction b of the points. Assign each point x_i with $\mathrm{sim}(x_i, \mu_{\gamma_i}) > \theta_s$ to cluster γ_i; assign the remaining points to the background cluster c_b.

Figure 2.2: The Streemer algorithm.

algorithm and we want to go over the data as few times as possible. Second, the previous steps have selected as cluster centroids points from dense areas, so the background is generally sparsely populated. Therefore, if we re-cluster, we do not expect to find clusters that are significantly different from those found in the first pass.

The inspiration for Streemer was Clustering by Committee (CBC) [96]. A big difference is that CBC uses many parameters and there is no principled way of setting them. This requires from the user to perform a grid search in the parameter space, which is time-consuming. Streemer on the other hand has only the minimum number

of parameters required and all the parameters have an intuitive interpretation, which makes setting them easier.

A significant advantage of Streemer is that it is scalable. For 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 found and N the number of data points). When it performs filtering of the candidate clusters it makes pairwise comparisons between themselves and this part 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 [48]. An iterative algorithm like k-means, on the other hand, performs order kN operations per iteration and, depending on the data, can require hundreds of iterations.

In summary, Streemer differs from k-means in several respects. It is able to give clusters that can have different sizes and variances, and it can also 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 foreground clusters. Streemer also compares favorably to standard clustering methods. It is significantly faster and more efficient than EM. Streemer is similar in speed to k-means[3] and it returns better clusters with fewer structural restrictions. It also requires fewer, more meaningful, parameters than CBC.

2.2.3 Setting parameters

Most streaming algorithms use an implicit or explicit similarity threshold [48]. While examining each observation, the algorithm decides whether to add this observation to an existing cluster, or to start a new cluster and put it there, based on that

[3]It is hard to compare different implementations of the algorithms. Streemer and k-means take approximately the same amount of time to execute on our dataset – 4-8 hours for non-optimized matlab code. Vectorization greatly reduces this, but still requires an hour while the gmeans [25] implementation of k-means in C runs in 2-3 minutes.

θ	# of candidate clusters
0.1	6492
0.01	4502
0.001	4426
0.0001	4426

Table 2.1: Examination of the effect of the threshold θ on the number of candidate clusters found by Streemer for the Computer Science dataset. Even when varying θ by several orders of magnitude, the candidate clusters found are roughly the same.

threshold. Therefore in all streaming algorithms the question arises how to specify that threshold. Streemer also requires a threshold on the similarity of items to cluster centroids. In our experiments we found that the final clustering result is not sensitive to the value of the threshold, θ. Table 2.1 shows some values of θ and the resulting number of candidate clusters that are found in step 2. The number of candidate clusters differs substantially across datasets, but, for a given dataset, θ has little effect. Small differences in the number of seeds generated in step 2 have no effect on the final result, as steps 4 and 5 filter them out. We used $\theta = 0.001$ in all of our experiments.

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 and size of communities and the intention to understand the effect of belonging more closely to a community vs. being more dispersed.

One can compare Streemer with a streaming clustering algorithm augmented with a final step for generating a background cluster by trimming far points from the clusters. Streemer looks very much like such an algorithm, except for the fact that it finds a number of candidate clusters larger than k in step 1, and uses a second

step to select k of those candidates as clusters. This makes Streemer less greedy than a streaming algorithm, which improves cluster quality.

2.2.4 Objective function

Given the heuristics employed in Streemer when selecting the final clusters, it is not obvious what objective function Streemer attempts to optimize. In this section we obtain this objective function and compare it with other similar functions that are used in clustering problems.

To help with notation we define the following indicator functions:

$$\delta_{ik} = \begin{cases} 1 & \text{if } \mathbf{x}_i \text{ is in candidate cluster } k \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

$$\gamma_k = \begin{cases} 1 & \text{if candidate centroid } \boldsymbol{\mu}_k \text{ is a foreground cluster centroid} \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

$$\beta_i = \begin{cases} 1 & \text{if } \mathbf{x}_i \text{ is assigned to the background} \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

The indicator function δ_{ik} is needed for the first step of Streemer where candidate clusters are formed. γ_k is defined for the second step of Streemer when the centroids of the final foreground clusters are selected from the candidates found in the first step. Finally, β_i is used to define which points are assigned to the background in the last step of Streemer.

For the case of clustering where the number of clusters, K, is fixed and there is no background, such as in k-means, the objective function that is being optimized is the sum of the within cluster squared distances with constraints that force every

observation to belong to one and only one cluster. It takes the simple form

$$\begin{aligned} &\min_{\delta,\boldsymbol{\mu}} \sum_{k=1}^{K}\sum_{i=1}^{N} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|^2 \delta_{ik} \\ \text{s. t. } &\sum_{k=1}^{K} \delta_{ik} = 1 \\ &\delta_{ik}(1-\delta_{ik}) = 0 \end{aligned} \tag{2.4}$$

The minimization over δ affects the assignments of the observations to clusters, whereas the minimization over $\boldsymbol{\mu}$ affects the estimates of the cluster centroids. It should be noted that the simultaneous minimization over both the assignments and the centroids gives a natural description of the optimization problem and is amenable to the iterative minimization procedure used by algorithms such as k-means, but it is not absolutely necessary for the formulation of the problem. In fact, we can dispense with $\boldsymbol{\mu}$ if we rewrite the squared differences to be between points in the same cluster. Equation (2.4) then becomes

$$\begin{aligned} &\min_{\delta} \sum_{k=1}^{K} \frac{1}{2N_k} \sum_{i=1}^{N}\sum_{j=1}^{N} \|\mathbf{x}_i - \mathbf{x}_j\|^2 \delta_{ik}\delta_{jk} \\ \text{s. t. } &\sum_{k=1}^{K} \delta_{ik} = 1 \\ &\delta_{ik}(1-\delta_{ik}) = 0 \end{aligned} \tag{2.5}$$

where $N_k = \sum_{i=1}^{N} \delta_{ik}$ is the number of points in cluster k. This transformation is possible for all the objective functions given below, but we will not make it explicit.

If we introduce a background cluster of fixed size b into the problem and maintain

a fixed K, the objective function becomes

$$\begin{aligned} & \min_{\delta,\boldsymbol{\mu},\beta} \sum_{k=1}^{K}\sum_{i=1}^{N} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|^2 \delta_{ik}(1-\beta_i) \\ \text{s. t. } & \sum_{i=1}^{N} \beta_i = b \\ & \beta_i(1-\beta_i) = 0 \\ & \sum_{k=1}^{N} \delta_{ik}(1-\beta_i) = 1(1-\beta_i) \\ & \delta_{ik}(1-\delta_{ik})(1-\beta_i) = 0 \end{aligned} \tag{2.6}$$

In the above equation the first constraint sets the size of the background to b and the second ensures that points belong fully either to foreground or the background. The last two constraints make the foreground points belong to one and only one foreground cluster. Using the method of Lagrange multipliers we can move the constraint for the β_i in the function to be minimized, which then becomes

$$\begin{aligned} & \min_{\delta,\boldsymbol{\mu},\beta,\lambda} \sum_{k=1}^{K}\sum_{i=1}^{N} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|^2 \delta_{ik}(1-\beta_i) + \lambda\left(\sum_{i=1}^{N}\beta_i - b\right) \\ \text{s. t. } & \beta_i(1-\beta_i) = 0 \\ & \sum_{k=1}^{N} \delta_{ik}(1-\beta_i) = 1(1-\beta_i) \\ & \delta_{ik}(1-\delta_{ik})(1-\beta_i) = 0 \end{aligned} \tag{2.7}$$

This minimization will give a solution with a fixed background size b, similar to what the background k-means algorithm finds. If we do not want to specify the exact background size but instead give a parameter that signifies the importance of the background, or in other words the cost of placing an observation in the background,

then we can drop the size b and make λ user-defined:

$$\begin{aligned} &\min_{\delta,\boldsymbol{\mu},\beta} \sum_{k=1}^{K}\sum_{i=1}^{N} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|^2 \delta_{ik}(1-\beta_i) + \lambda \sum_{i=1}^{N} \beta_i \\ \text{s. t. } &\beta_i(1-\beta_i) = 0 \\ &\sum_{k=1}^{N} \delta_{ik}(1-\beta_i) = 1(1-\beta_i) \\ &\delta_{ik}(1-\delta_{ik})(1-\beta_i) = 0 \end{aligned} \tag{2.8}$$

This modification exploits the duality between a threshold λ and a fixed background size b, similar to the duality between the threshold θ of streaming clustering and the number of clusters k in k-means.

We now look at the case of no background and variable number of clusters. We introduce a cost α of starting a new cluster, which is similar to but not exactly the same as the α parameter of the Dirichlet process.

$$\begin{aligned} &\min_{\delta,\boldsymbol{\mu}} \sum_{k=1}^{N}\sum_{i=1}^{N} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|^2 \delta_{ik} + \alpha \left\| \sum_{i=1}^{N} \delta_{i\cdot} \right\|_0 \\ \text{s. t. } &\sum_{k=1}^{N} \delta_{ik} = 1 \\ &\delta_{ik}(1-\delta_{ik}) = 0 \end{aligned} \tag{2.9}$$

Since the number of points that are clustered is N, the number of clusters K selected by the model will lie between 1 and N. Thus, we set the summation boundaries for k to be these values and use the indicator δ to ignore values for $k > K$. In the equation above we used the L_0 norm to compute the number of clusters from δ.

Finally, if we introduce the background again, the above equation becomes

$$\begin{aligned}
&\min_{\delta,\boldsymbol{\mu},\beta} \sum_{k=1}^{N}\sum_{i=1}^{N} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|^2 \delta_{ik}(1-\beta_i) + \alpha \left\| \sum_{i=1}^{N} \delta_{i\cdot} \right\|_0 + \lambda \sum_{i=1}^{N} \beta_i \\
&\text{s. t. } \beta_i(1-\beta_i) = 0 \\
&\quad \sum_{k=1}^{N} \delta_{ik}(1-\beta_i) = 1(1-\beta_i) \\
&\quad \delta_{ik}(1-\delta_{ik})(1-\beta_i) = 0
\end{aligned} \tag{2.10}$$

There is one important difference between equations (2.9) – (2.10) and a simple streaming clustering algorithm that uses a threshold θ. The streaming clustering algorithm uses K inequality constraints of the form $\|\mathbf{x}_i - \boldsymbol{\mu}_k\|^2 \leq \theta$ and when the existing constraints cannot be satisfied a new cluster is created and a new constraint is introduced. The above objective functions, on the other hand, use a single penalty α and the creation of new clusters is determined by the trade-off between the cost α of a new cluster if an observation is assigned to a new cluster versus the increase in the sum of squared distances if the observation is assigned to an existing cluster. This allows some of the clusters to have a radius smaller than θ and others greater than θ (for some value of θ that is determined by α), depending on how cohesive they are and therefore how large their sums of squared differences are.

Recall how Streemer deals with clusters in its second pass. As it processes a cluster found in the first pass, it either keeps it as a cluster, merges it with another existing cluster, or assigns it to the background. If a new cluster is formed, this incurs cost α plus whatever the sum of squared distances is for the points in that cluster. If the points are merged with an existing cluster, then there is no cost α, but there is an increase in the sum of squared distances of the cluster that they merge with. If they are assigned to the background then there is a cost of λ per point. All these choices of Streemer can be modeled by equation (2.10), which corresponds to a clean objective for what Streemer attempts to optimize. Streemer itself, however, is not as clean and the many comparisons employed by the algorithm, such as comparing the

cohesiveness of the candidate clusters or their distances to each other, are heuristics whose aim is to avoid local minima. Next we attempt to gradually introduce these heuristics in the final objective function for Streemer.

In order to arrive at the final objective function being optimized by Streemer, we start by first defining separate objective functions for each of the three steps. Each function uses the optimized parameters of the previous step as fixed parameters. Once all the individual objective functions are defined, we will combine them to get a final objective function for Streemer.

The objective function for the first step is trying to maximize the within cluster similarity while keeping the number of clusters as small as possible:

$$\begin{aligned} \max_{\delta,\boldsymbol{\mu}} & \sum_{i=1}^{N}\sum_{k=1}^{N} \mathrm{sim}(\mathbf{x}_i, \boldsymbol{\mu}_k)\delta_{ik} - \alpha \left\| \sum_{i=1}^{N} \delta_{i\cdot} \right\|_0 \\ \text{s. t. } & \mathrm{sim}(\mathbf{x}_i, \boldsymbol{\mu}_k) \geq \theta \\ & \sum_{k=1}^{N} \delta_{ik} = 1 \\ & \delta_{ik}(1 - \delta_{ik}) = 0 \end{aligned} \tag{2.11}$$

The first constraint forces the candidate clusters to have a minimum within cluster similarity of θ. The other two constraints make sure every point belongs to one and only one cluster.

At the beginning of the second step δ and $\boldsymbol{\mu}$ are fixed to the values found in the previous optimization. Let c_k be the cohesiveness of candidate cluster k:

$$c_k = \frac{\sum_{i=1}^{N} \mathrm{sim}(\mathbf{x}_i, \boldsymbol{\mu}_k)\delta_{ik}}{\sum_{i=1}^{N} \delta_{ik}}$$

The second step then selects those centroids of the candidate clusters that are the

most cohesive and thus maximize the overall cohesiveness:

$$\begin{aligned} &\max_{\gamma} \sum_{k=1}^{N} c_k \gamma_k \\ \text{s. t. } &\sum_{i=1}^{N} \delta_{ik} \gamma_k \geq minsize \\ &\gamma_k(1-\gamma_k) = 0 \\ &(\gamma_k + \gamma_j)\mathrm{I}[\mathrm{sim}(\boldsymbol{\mu}_k, \boldsymbol{\mu}_j) \geq \theta] \leq 1 \\ &c_k \gamma_k \geq c_j \mathrm{I}[\mathrm{sim}(\boldsymbol{\mu}_k, \boldsymbol{\mu}_j) \geq \theta](1-\gamma_j) \end{aligned} \tag{2.12}$$

The first constraint above forces the selected clusters to have at least *minsize* points and the second makes a candidate cluster to be either selected or not. The third constraint makes sure that at most one centroid is chosen between two neighbors and by extension at most one centroid is chosen in the whole neighborhood. The last constraint ensures that the most cohesive candidate in the neighborhood is chosen (or, if the candidate cluster is isolated then it is the only cluster in its neighborhood and is thus also the most cohesive). Suppose $\boldsymbol{\mu}_k$ and $\boldsymbol{\mu}_j$ are two centroids in the same neighborhood. If $c_k \geq c_j$, the last constraint is satisfied if $\boldsymbol{\mu}_k$ is chosen and $\boldsymbol{\mu}_j$ is not. If neither are chosen the constraint is violated because cluster cohesiveness is always positive. If $\boldsymbol{\mu}_j$ is chosen but $\boldsymbol{\mu}_k$ is not, then the specific constraint is satisfied, but the corresponding constraint with the indices j and k swapped is violated.

For the last step of Streemer the values of γ and $\boldsymbol{\mu}$ are considered known and fixed. That step then reassigns points to foreground clusters or the background so as to maximize the within cluster similarity of the foreground clusters while placing

b points to the background:

$$\begin{aligned} & \max_{\delta,\beta} \sum_{i=1}^{N} \sum_{k=1}^{N} \text{sim}(\mathbf{x}_i, \boldsymbol{\mu}_k) \delta_{ik} (1 - \beta_i) \\ \text{s. t. } & \sum_{i=1}^{N} \beta_i = b \\ & \beta_i (1 - \beta_i) = 0 \\ & \sum_{k=1}^{N} \delta_{ik} (1 - \beta_i) = 1(1 - \beta_i) \\ & \delta_{ik} (1 - \delta_{ik})(1 - \beta_i) = 0 \end{aligned} \tag{2.13}$$

The first constraint forces b points to the background, the second makes every point belong either to a foreground cluster or the background and the last two make the foreground points belong to one and only one foreground cluster.

Note that the last objective function re-optimizes δ, which is also a variable being optimized in the first function. Therefore, if we combine it with the first function, there will be a conflict in the way the δ values are set by Streemer. We could make δ a new variable (say δ'), make the third step objective independent of that of the first step and combine all three optimizations into one (albeit a complex one). Instead, below we combine the first two objective functions into one (since they do not optimize the same variables) and leave the last function separate for ease of presentation. This is not a serious problem because the last step of Streemer that finds the background is optional and not integral to the algorithm. In addition, the last step only performs a reassignment of points without affecting the cluster centroids.

By combining the first two functions, we thus get the following objective function

for the first two steps of Streemer:

$$\max_{\delta,\boldsymbol{\mu},\gamma} \sum_{i=1}^{N}\sum_{k=1}^{N} \mathrm{sim}(\mathbf{x}_i, \boldsymbol{\mu}_k)\delta_{ik} - \alpha \left\| \sum_{i=1}^{N} \delta_{i\cdot} \right\|_0 + \sum_{k=1}^{N} c_k\gamma_k$$

$$\begin{aligned}
\text{s. t. } & \mathrm{sim}(\mathbf{x}_i, \boldsymbol{\mu}_k) \geq \theta \\
& \sum_{k=1}^{N} \delta_{ik} = 1 \\
& \delta_{ik}(1-\delta_{ik}) = 0 \\
& \sum_{i=1}^{N} \delta_{ik}\gamma_k \geq minsize \\
& \gamma_k(1-\gamma_k) = 0 \\
& (\gamma_k + \gamma_j)\mathrm{I}[\mathrm{sim}(\boldsymbol{\mu}_k, \boldsymbol{\mu}_j) \geq \theta] \leq 1 \\
& c_k\gamma_k \geq c_j\mathrm{I}[\mathrm{sim}(\boldsymbol{\mu}_k, \boldsymbol{\mu}_j) \geq \theta](1-\gamma_j)
\end{aligned} \tag{2.14}$$

In the first step of Streemer, $\gamma_k = 0$, making the $\sum_{k=1}^{N} c_k\gamma_k$ term zero and trivially satisfying the last four constraints. In the second step, Streemer holds δ and $\boldsymbol{\mu}$ constant and only optimizes γ.

2.3 Validation of Streemer

2.3.1 Data used

In our experiments we used a dataset consisting of papers from computer science fields. The data were drawn from CiteSeer [42], 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. It has over 700,000 indexed papers in its database. We cross-referenced these papers with the DBLP Computer Science Bibliography [75],

a European database with over 600,000 papers 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 are from between 1992 and 2003.

The documents in our dataset were represented as sparse Boolean vectors, i.e. vectors with elements 0 or 1. The j-th element of vector i is 1 if document i cited document j and 0 otherwise. The vectors were L_2 normalized before clustering to give documents with different number of citations equal weight. The dataset consisted of 341,458 documents, represented as Boolean vectors of 197,163 dimensions[4]. We also extracted the text of the paper titles and keywords.

2.3.2 Validation method

In order to see how well Streemer finds clusters, we compared it to k-means as a baseline. For the comparison to be meaningful, the algorithms should both find the same number of clusters. We therefore modified Streemer to perform in step 3 a binary search for the value of *minsize* that gives k clusters. However, k-means does not find a background cluster, therefore we developed a simple algorithm, based on k-means, that does find one. This algorithm, which we call *background k-means*, takes two parameters: the percent background b and the number of clusters k and returns $k-1$ foreground clusters and a single background cluster. It differs from standard k-means in that at each iteration only the nearest $(1-b)N$ points are assigned to clusters; the rest belong to the background. Only these $(1-b)N$ points are used for estimating the cluster centroids. As in k-means, all the foreground clusters are convex, have piecewise linear boundaries and equal prior probabilities.

We compared the three algorithms by clustering the Computer Science data set and evaluating the clusters by measuring how homogeneous the documents in the

[4]The number of dimensions is less than the number of documents because some papers in our dataset are not cited by the others. We removed these columns of all zeros from our sparse matrix.

clusters were in terms of the journals and conference proceedings where they were published. The mapping between journals/conferences and knowledge communities is, of course, not one-to-one, but they are highly correlated. For example, most researchers in computer architecture, graphics, and machine learning publish in different venues. CiteSeer provides publication information, but because it is automatically extracted, 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 our evaluations we used weighted average entropy (WAE) [34] as a measure of cluster purity:

$$WAE = \sum_{i=1}^{C} \frac{n_i}{N} E_i \tag{2.15}$$

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., conferences and journals) for points in cluster i. The lower the WAE, the better the clustering matches the given data labels.

We also computed the normalized mutual information (NMI) [118] and found results consistent with the WAE results, so we do not report them.

2.3.3 Validation results and discussion

In the results presented below, the cosine measure was used as the similarity function in all three methods. We used $\theta = 0.001$ for Streamer. For k-means, which does not explicitly find a background cluster, we found the k clusters and as a final post-processing step we assigned to the background the bN points that were most distant from their cluster centroids.

We tested each algorithm for $k = 10, 20, 40$ and $b = 0, 0.33, 0.5, 0.67, 0.9$. For each value of k and b we ran the algorithms ten times starting from a random initialization. Figure 2.3 shows detailed results for one choice of parameter values,

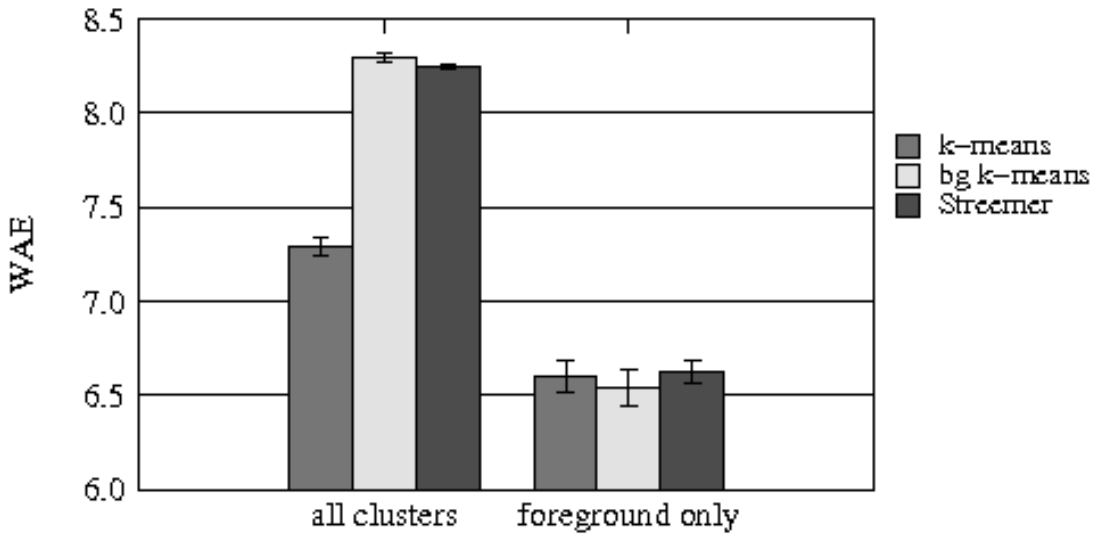


Figure 2.3: Average WAE over 10 runs evaluated on all or only on the foreground clusters ($k = 20$, $b = 0.67$). The error bars are plus or minus one standard deviation. Lower WAE is better.

$k = 20$ and $b = 0.67$. The bar chart shows the average WAE and one standard deviation error bars for the ten runs (lower WAE is better). The bar for k-means (all clusters) is before the post-processing step of creating the background cluster, whereas the bar for k-kmeans (foreground only) is computed after generating the background and excluding the points in it. We note that the WAE for all the clusters, including background, is much better for k-means than for Streemer and background k-means. This is not surprising, as the background cluster is of low cohesiveness by construction, therefore its entropy is quite high and it negatively affects the overall weighted entropy. On the other hand, the WAE for the foreground clusters only is lower for every algorithm. Searching for a background cluster has the effect of finding better foreground clusters. All the results of the sensitivity analysis are shown in table 2.2.

Our robustness analysis for different values of k and b found that k-means always performs the best in terms of all clusters (including background). This is expected, since the background is by construction of low quality. However, in this work we are

	$k = 10$		$k = 20$		$k = 40$	
b	bgkm	Strm	bgkm	Strm	bgkm	Strm
0	7.69	8.39	7.17	7.95	6.71	7.45
0.33	7.46	7.79	6.91	7.28	6.44	6.77
0.5	7.30	7.72	6.75	7.09	6.28	6.42
0.67	7.11	7.35	6.54	6.63	6.07	5.99
0.9	6.39	6.13	5.75	5.67	5.22	5.16

Table 2.2: Sensitivity analysis. Reported is the mean of the WAE over 10 runs for the foreground clusters (points in the background are ignored). bgkm is background k-means, Strm is Streemer.

not interested in clustering every point, so we do not report these results in table 2.2. Also, when finding a background, the foreground clusters for k-means were always worse than background k-means and so we omit these results as well. Streemer is slightly worse than background k-means for small b and k. As the background size or the number of clusters increases, both Streemer and background k-means perform better, but Streemer improves faster. At $b = 0.67$ they are not significantly different, except for $k = 40$, where Streemer is significantly better than background k-means.

Even though background k-means performed better than Streemer for some parameter settings, we still chose to use Streemer for our analyses in the rest of the paper for several reasons:

- Streemer only makes two passes over the data, whereas background k-means iterates over them until convergence. In our experiments Streemer was approximately an order of magnitude faster, so it is preferred for large datasets. Of course, Streemer suffers a penalty because of this, but, as we saw, the penalty is relatively small.

- Streemer is agnostic about the size distribution of clusters, unlike background k-means which assumes an equal size prior. In our tests the foreground clusters found by Streemer had very different sizes. The ratio of the biggest to the smallest cluster was about 20, while for background k-means the same ratio was

only 2. For the application area we examine, we expect that some knowledge communities are small and others are big. The clusters returned by Streemer match better this size variability.

- Our goal is to examine the evolution of clusters over time, so we want clusters to be created or destroyed. Algorithms that use a fixed k are not amenable to this, because they will always find the same number of clusters in every time period. Streemer uses a threshold, so if we keep the threshold value fixed over all time periods, we can get different numbers of clusters and the results will be comparable.

There is vast literature on clustering using both iterative and streaming algorithms. Iterative methods, such as spectral [19, 109, 21], or information theoretic clustering [111, 20, 4] are generally too slow for large datasets. A faster alternative is to use methods which make either a single-pass (a streaming algorithm) or a small number of passes over the data.

Streaming clustering, which is used in the first step of Streemer, has many attractive theoretical properties [48]. However, first finding a large number of clusters which are then reduced to the requested number k, as is done in Streemer's second step, gives significantly higher quality clusters than single pass algorithms [17]. The idea of cluster cohesiveness in Streemer was inspired by the Clustering By Committee (CBC) algorithm [96]. 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.

Streemer falls into the category of distance-based methods, as defined in [128]. By making two passes over the points and finding a background cluster, Streemer avoids the problems mentioned there of frequently scanning the points and of treating them all equally. Much like the well-known BIRCH algorithm for clustering large datasets [128], Streemer first finds a large number of clusters which it later reduces. BIRCH differs in that it first builds a tree whose nodes keep statistics about subsets of the

data and then uses a clustering algorithm (in the paper agglomerative) to cluster the tree leaves. Because BIRCH only keeps clustering feature (CF) vectors about subsets of the data and not the actual data, in order to reduce memory requirements, not everything can be computed from the data that was possible originally. Therefore, unlike Streemer, BIRCH can only use certain distance functions.

DBSCAN is another algorithm with many similarities to Streemer [28]. 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 [49]. 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 any preliminary clustering. If two distant clusters are connected by a string of points, then 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.

Our work can be viewed as following in the tradition of citation analysis, but differs from the historically more descriptive work using unsupervised learning in that we also use supervised learning models on top of our unsupervised clustering. Citation (and co-citation) analysis is quite old [119, 113, 114, 12], but has seen a revival with newer clustering algorithms, such as Latent Dirichlet Allocation (LDA) [9, 47] and its variants [7]. Other papers deal with the similar problem of clustering web pages, where the links take the role of citations. [36, 41] find web communities by using the connections between the observations. [84] use co-citation structure between web pages to discover new topics.

The Streemer algorithm was introduced in [63], where it was compared in detail to k-means and background k-means. Some results regarding the evolution of knowledge communities were also presented there. In the present paper we provide extended descriptions of the features used for community growth prediction and

give specific examples and in-depth analysis of the results. We also look at emerging, splitting, joining and dying clusters that were found. Additionally, we apply Streemer to the new problem of predicting citations of papers and analyzing when new knowledge has greater impact.

2.3.4 Another application - finding stop words

The ability of the Streemer algorithm to cluster the observations into foreground or background opens the possibilities of other uses besides finding knowledge communities. The defining characteristic of the background cluster is that it contains those observations that don't fit well with any of the foreground cluster. When we apply Streemer to community analysis in this chapter, we are interested in the foreground clusters, while the background is only used to improve the foreground quality. However, in other cases, one may be interested in the observations that actually get assigned to the background.

Consider, for example, the automatic identification of stop words. Stop words are words that are usually filtered out before processing text because they are considered common and not conveying any useful information. Lists of stop words are usually manually created, so it is desirable to generate them automatically for other languages. A defining characteristic of stop words is that they tend to appear equally randomly in most documents. Content words on the other hand convey most of the information of a document and are thus related to the topic of document. For example, sports articles will contain many words related to sports and financial news stories will contain many terms that have to do with finance and economics. Every document however will contain articles, pronouns, modal verbs and other stop words.

One might propose to use Streemer to cluster the words in a collection of documents according to the documents they appear in with the expectation that each foreground cluster will contain words about a specific topic, whereas the background cluster will have all the stop words since they don't fit well to any specific topic.

This turns out not to be the case.

We tested Streemer on a collection of words extracted from 50,000 Medline abstracts. The words were represented by vectors whose elements were the number of times the word appeared in a document. The number of vector features was equal to the number of documents. To eliminate spurious results we stemmed the words and removed those with fewer than 10 overall occurrences, leaving approximately 4,700 words. Table 2.3 shows some foreground clusters that were found. The identified topics seem quite good, however the goal is to find stop words in the background. The background is given in table 2.4 and, surprisingly, it does not contain any stop words. Instead the words in the background are mostly specialized medical terms that do not seem to have any apparent connection. As to what happened to the stop words, the majority of them appear in a foreground cluster (table 2.5).

Why did we get such unexpected results? The words were clustered according to the documents they appear in, so words whose distributions of document appearances are similar were assigned to the same clusters (for example the words *embryo* and *egg* that appear in abstracts about reproduction were in the same cluster). Stop words also have similar distributions; they appear in most documents almost uniformly! Because of their strong distributional similarity, they are all assigned to a foreground cluster. The question that remains is "What are the words in the background?" We hypothesize that they are words that either did not appear enough times to be tied strongly to an existing cluster, or they occur in documents from two (or more) topics. They are therefore somewhat similar to both topics (i.e. clusters) but not very similar to either. Also, Streemer has a preference for larger clusters, so many small clusters are just thrown to the background.

This example shows that one should be careful when applying Streemer (and in general any algorithm) to a problem. What at first glance might seem plausible, it may fact lead to unexpected outcomes. A deep understanding of the problem at hand and the inner workings of the algorithm is important for a successful application.

retrospect prospect thromboembol studi incid risk who pill user review survei hospit oc out predispos data associ complic epidemiolog etiolog pregnanc mere statist evalu assess match colombia morbid surveil admiss complaint perform hemorrhag 1972 perinat outcom diagnos embol criteria consecut pariti done remark diagnost select british 49 ectop hip cardiovascular prognosi woman advers trait categori physician comment comput tabul random 1968 62 reflux neurolog defin resum uncertain literatur elsewher deliveri methodolog atresia accuraci labour 71 smoke anticoagul illiter causal evacu accid quit exclud 46 obstetr acquir colleg fatal meaning roentgenograph submit chart roentgenogram
speech intellig db hear leader acoust palat languag tape auditori judgment sound cleft word instruct score invest discrimin tone hemispher rehabilit comprehens prophylact inadequ tendenc flap nois set hysterectomi 96 judg item psycholog speak parkinson laryng wider esophag mask heroin sibl scale thought development adjust expenditur confus skill dph placement regim emot bia verbal gather confid tongu hyperact freedom
dystrophi muscular polyribosom muscl motor ribosom atrophi neuromuscular proportion intim sharp weak belong abnorm notabl elast thicken isometr patholog disord granular recess anomali eros hereditari pelvi
noradrenalin adrenalin catecholamin adrenoceptor phentolamin isoprenalin ap sympathet dopamin anaesthet infus inotrop potenc acetylcholin strip action shorten prolong output morphin propranolol medulla contract hz content guinea transmitt agonist biphas na injur contractil puls arous ca 300 diminish plateau plexu accompani longitudin hypoglycemia unlik da

Table 2.3: Examples of foreground clusters from the stop word identification experiment.

surround ecolog ascertain boundari adenoma offspr ac conflict intrins interf calor renin metropolitan workshop occupi t3 evolv allogen ask vasoconstrict horn recognit accur 108 septum hypertroph framework lifetim configur taurochol childbirth gastric prenat color domin duoden avian summer advic lytic academ canadian union neglect elig unexpect brazil gallbladd postmenopaus wish attenu alpha1 cisterna lysat circuit ribonucl encount dmso alkaloid amphetamin resin discov glaucoma facial oocyt assur hexos sent epilepsi cyst ligament viabil leav ileum polynucleotid check phe pilot disrupt preserv inde correctli retinol infanc endonucleas bile bill referr benefici recruit opinion esterifi iodin briefli lumbar easili mycobacterium carefulli infer virul bud ru sudden ruptur notic distress uncompl degen lidocain clinician voltag medicin pouch karyotyp nurs specul hl naproxen congest oligonucleotid fatigu dissect habit furosemid galactosidas ly vii phenylalanin hemodialysi book hard methadon max hybrid calcitonin uret calibr glomerulonephr 1956 decompress lose vena unless primer fate salivari tempor smoker cytogenet ward crystal hdl devis contraind prosthet thaw ethanolamin ch dysfunct exact me haemorrhag cea chlorophyl progeni anxieti scheme favour bp cranial secretin tyr exhaust search micro endotheli mention intercours leu practition eighti porcin mathemat polymorph dissolut shorter stand father abstin diazepam gtp dermal unsuccess wast met streptomycin parenchyma charact 94 press felt jaundic sle brown methotrex sphere safeti subtyp plasmid mesenchym undergon tubulin z dt chemotact dy simplex millilit lymphoblast specialist periodont finger uninfect quantifi thiamin understood rrna unrespons sclerosi oblig elucid indigen mul prl classic miss fish dba sterol enamel autolog let lend supervis usag pathogenesi tm tool drosophila nonpregn cerebrospin egypt artifici coverag perceiv breed sea satisfact illumin distort inabl york postcoit weigh profound gc handl illustr handicap parenchym territori glycol attract trh tent unclear transcrib curettag vacuum antitrypsin olfactori genotyp heterolog peer ic mastectomi lactic midwiv caution bactericid heavili seros suspend centr mononuclear know scientist genu english vesicular phosphatidylethanolamin mellitu la estriol leukaemia pellet undesir cadmium hyperparathyroid menarch lengthen apart propag laser transloc arachidon mens pup goiter ti ppd foci nineti prescrib addict enumer moment ethic influenza hla radioiodin seizur suit mission student meat f1 amend recal therapist gentamicin insemin helper tropic thyrotropin formal virus televis allograft satellit calcif psychiatrist specifi antihypertens intravascular sporul radionuclid rhythmic adenoviru b6 csf multipar bsa 99mtc fundu stai sheet ria umbil leaflet criterion granulocyt faculti glc pathophysiolog nicotin diuret quench autosom c3 verifi hematoma headach twin 110 instabl eeg yr radio crimin spring counselor angiotensin rs defens rodent rubella juvenil aggress measl cs stepwis schizophren altitud peculiar empti fibrinolyt intent copi le modal disk ambient ldh herd shoulder genom client imped b12 stillbirth mi c4 gynecologist dental twitch blast oscil pepsin rheumat intercellular gastrin britain duodenum ultrason bilirubin titl sr paramed gangliosid file a1 q detach vi trigger convuls sv40 pentagastrin pra disintegr cattl sheath streptococci tendon ala spermicid antenat birthweight splanchnic hsv suicid club camera ser bradykinin gly choroid pharmacist mlc librari

Table 2.4: The background cluster from the stop word identification experiment.

treat untreat with treatment in of were wa and patient to anim a control after or from on no that rat not result for therapi these had effect thi group but an at 1 5 2 3 signific 6 receiv show 4 compar increas all follow significantli normal dose 10 observ when did level there onli both also 7 those surviv daili suggest given induc found inject 8 howev administr case three less 12 occur decreas week initi time other us reduc month 50 respons dure kg 20 either 100 previous indic while chang four administ present although befor surgic serum 15 remain greater lower total per period demonstr same without caus five recurr 9 first hr wherea evid six complet higher well remiss vitro i pretreat produc report alon clinic delai subsequ low seen eight conclud alter successfulli find further 22 possibl rabbit sarcoma could blood examin due within recov earli twenti up system author elev ani di long symptom investig chemotherapi none each function involv cortison methylcholanthren affect mean non assai 19 mark 13 fail 21 irradi therapeut whole cancer thu cyclophosphamid progress note primari experi nine hypothyroid 60 prevent requir thirti then lymphocyt cours persist contrast least histolog releas 56 second whether 40 36 ten confirm markedli trial old intraperiton seri phospholipas immun nor prior conserv few seven lesion 26 enhanc regimen cure respond side usual peripher phenobarbit srbc influenc thyroid whom frequent exposur disappear 37 capac 27 35 salin return maintain 70 32 except impair durat sensit cold 29 melanoma good experiment twice necropsi slightli taken probabl intramuscularli failur doubl cytotox metastat 42 39 spontan lymph becam 45 34 phytohemagglutinin thyroxin absenc 63 mix aliv cervix percent similarli x pronounc wash mate 53 blastocyst 55 expos abdomin secondari fifteen diabet 61 our 72 replac bird syngen 33 adenocarcinoma start highest placebo suffer 31 presum rosett fourth unaffect tetracyclin repeat lutea fifti intervent prove augment 75 sec twelv 005 93 might nervou ceas entir indomethacin altern gal median blind nodul earlier definit syndrom moder satisfactori 64 histopatholog interfer t4 lactat randomli onset nodular varieti theta tsh protocol healthi resorpt numer 67 seventi intramuscular excel conceiv disloc v c3h best sham menopaus hyperthyroid relaps asymptomat outpati lysosom spong proven pg sixti lumen render latent litter diethylstilbestrol cessat prednison wk regular 88 indirect span ms rifampicin splenic strike schedul hundr prognost rest cy interferon beyond mpa grade 800 abdomen again histiocyt epilept disturb preliminari tc withdraw cytolog eleven endometriosi rh thyroglobulin ip tuberculosi

Table 2.5: A foreground cluster that actually contains mostly stop words.

2.4 Clustering over time

For the validation experiments in the previous sections, we applied the clustering algorithms to the whole dataset. Documents published in early and later years were all clustered together. This is acceptable when evaluating the clustering algorithms, but not when the intention is to analyze the evolution of knowledge communities. Such communities change over time, they can grow or shrink as more or less papers are published about their topics of interest. They can even disappear, either because of waning interest, or because they split in smaller, more specialized communities. Other communities may merge into one and new communities may be formed about topics that did not exist in earlier years.

In such cases, where the data come from a dynamic environment, the clustering method must allow the clusters to evolve in the ways described previously. A simple clustering of all the data would incorrectly assume that the clusters are static over time, which we know is not the case for knowledge communities. Since we could not perform a straightforward clustering of all of our data, we had to develop an iterative clustering scheme (which we call "rolling clustering") that successfully addresses the requirements of the dynamic environment.

We divided the papers by publication date into 5-year groups and clustered the papers of each group separately. The 5-year periods were shifted by 1 year and consecutive groups were partially overlapping for 4 years. For example, we clustered the documents published from 1985 to 1990 based on what they cite. Then we clustered the documents published between 1986 and 1991, then those between 1987 and 1992 and so on. We then manually matched up clusters from consecutive groups using the common documents in the groups. For example, the clusters found for 1985-1990 were matched to the clusters for 1986-1991, by looking at the common papers in 1986-1990. A cluster from 1986-1991 that contained mostly the same papers as a cluster from 1985-1990 was considered to represent the same community. The temporal overlap ensures some consistency in cluster composition, while allowing

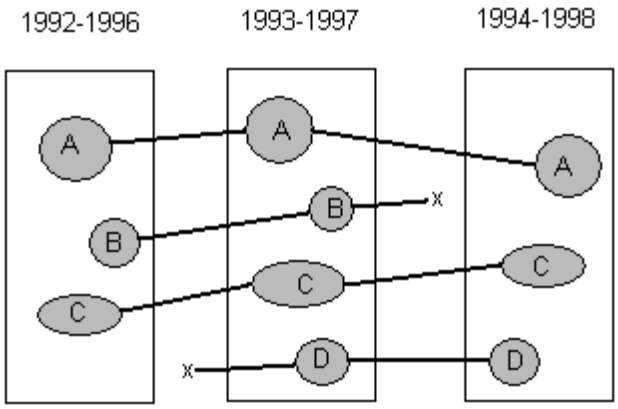


Figure 2.4: Illustration of rolling clustering with cluster B disappearing and cluster C forming in the 1993-1997 time period. By using the common documents from the overlapping time periods we can match the clusters and track their evolution.

new clusters to be created and existing clusters to merge or wither away.

A hypothetical example of rolling clustering is shown in figure 2.4. The three rectangles represent three consecutive 5-year periods. The round shapes are communities found by clustering the documents in each time period. In this example, clusters A and C are present throughout the three time periods. Cluster B disappears in the third time period and cluster D appears in the second. Throughout the time periods one can observe the dynamic movement of the clusters as they evolve over time. Essentially, what we did was to chain together a series of overlapping clusterings so that we can create continuity while allowing for an evolving population of communities.

We used Streemer to perform rolling clustering, without the modification in the previous section, that is we used a threshold and the number of clusters found was not fixed. The threshold value was the one found in section 2.3 using binary search which gave 20 clusters; the background fraction was $b = 0.67$. Both these values were selected by examining the clusters we found in section 2.3 and getting experts' opinions about the cluster numbers and their sizes.

Clustering based on words or citations has been used extensively for text mining (for example [72, 116]). Trying to identify communities or clusters and how they evolve over time has also been studied in the past. For example, [120] use an algorithm with no user-defined parameters to cluster dynamic networks of interactions and track their evolution. [74] explore and model the properties of graphs evolving over time. Their findings can provide inspiration for improving in the future the simple "rolling clustering" methodology that we used. There is some work on dealing with evolving topics over time (see for example [7, 124]), but the reasons we used the "rolling clustering" methodology were two-fold. First, it is much simpler, and second, it is necessary for our later analysis that the clusters formed at a given time depend only on older data, since we will be using the clusters to predict the future. Our methodology requires all cluster assignments in each year to be backward-looking only. At the same time, we find very high continuity between clusters, since the knowledge landscape we created changes gradually. Additionally, we can easily use our measures of "centrality" at the paper and cluster levels that refer to the appropriate time frame.

In sections 2.5 and 2.6 we use the "rolling clusters" to predict the success of knowledge communities and of papers. For our predictions to be valid we need to use only clusters (and cluster features) based on earlier data. Because of the procedure we follow, clusters at a given time only depend on prior history. We only use the information available at that time; no knowledge of the future is used.

2.4.1 Analysis of clusters

In this section we look at the "rolling clusters" we found. These evolving clusters give an interesting view of the field of computer science and the changes in the knowledge communities within it.

According to the literature on search and positioning for knowledge development, the evolution of clusters over time is the result of the choices agents make as they

position themselves on this landscape [52, 51]. The period from 1992 to 2003 marked a dramatic growth of computer science and a radical evolution of its scope of use. To 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 2.5 and table 2.6 give details on the 22 knowledge communities we identified. Figure 2.5 shows the evolution of the sizes of the knowledge communities with time, as well as a timeline of when each community appeared and disappeared. Table 2.6 gives for each cluster the number of documents assigned to it, a name given manually based on inspection of the papers assigned to the cluster and using our expertise and the 5 most common words in the abstracts of the papers in the cluster together with a count of the word occurrences.

There is no ground truth in the data we are clustering. As authors research and publish, their areas of interest can drift, or the topic of their work may require citing quite different papers than their prior or subsequent publications. In the end, what we are trying to cluster is a cloud of diffuse communities that are formed by complex interactions. We do not believe that the 22 clusters we found are in any way definitive. The hypothesis we are exploring is that a clustering which is meaningful, in the sense that it captures some of the community structure, will prove useful. The verification that the specific clusters we found are reasonable is that in the next two sections they are used to build models that make good predictions about the community growth and the impact of papers. Our goal in this section therefore is not to present clusters of papers and claim that we have found the existing scientific knowledge communities, but to illustrate how our analysis can uncover structure in the citations. The plots are useful examples in exploring the citation connections in our dataset and an informal verification that what we have found is at least reasonable.

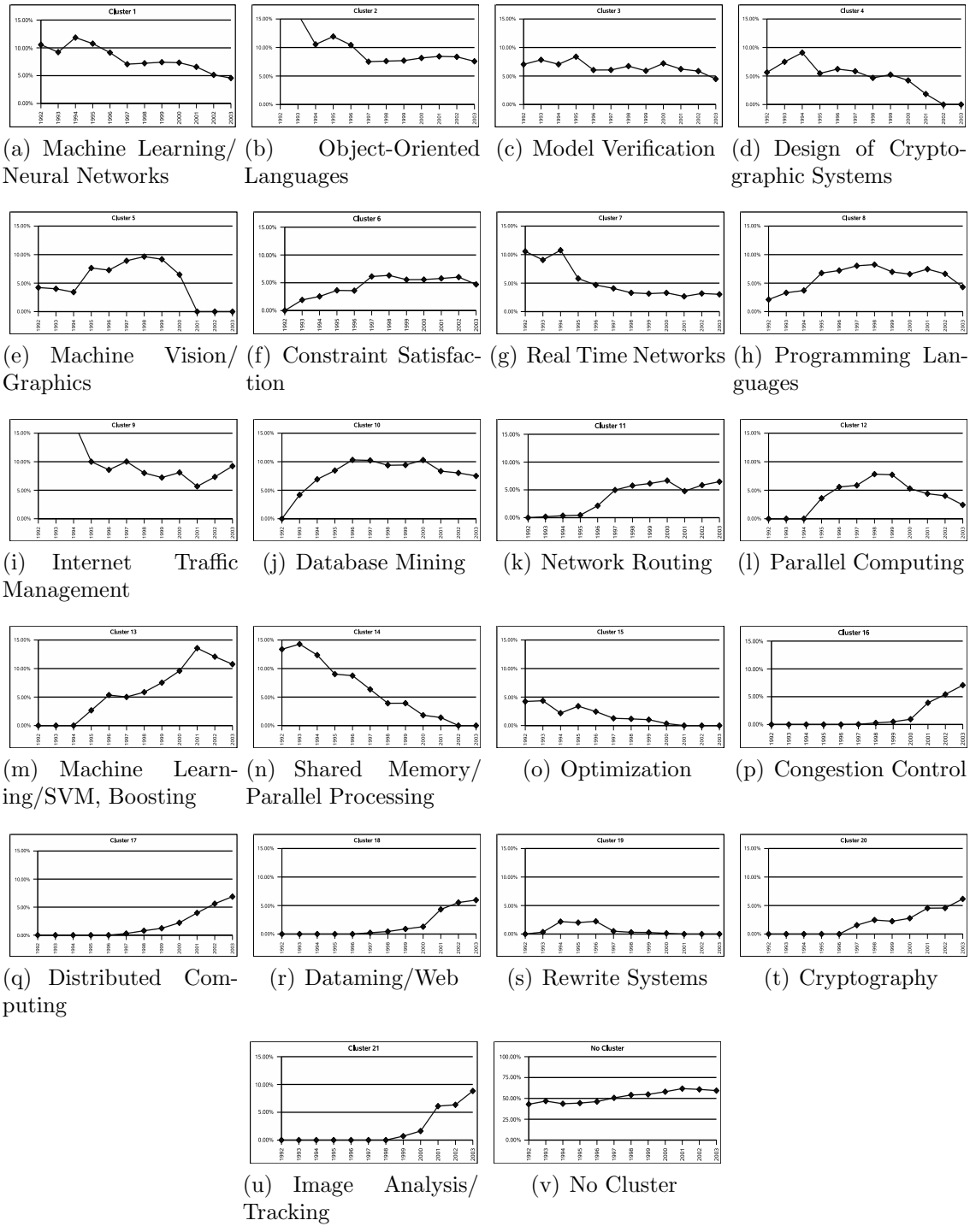
One thing to note is that the clusters represent knowledge communities and not subjects. For example, computer architecture is a subject that is very wide and

encompasses several communities. Computer architecture is not cohesive enough to be represented as a single cluster. It is instead split in several clusters in our results. Other subjects that do not appear in the 22 clusters we found may be too small, or too diffuse to show up, or they may be a part of a larger community.

Our findings give a sense as to how such an analysis can help us explore the collection. For example, notice that in 1992 we discovered 14 knowledge communities. Between 1992 and 1999 we observed seven new knowledge communities forming and none disappearing. From 1999 to 2001 we found that 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.

We were also able to identify cases of paradigm shifts. 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. 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".

Besides birth and death of communities, we also observe merging and splitting. For example, in 1996 cluster 9, representing the knowledge community researching "Internet traffic management," shattered to form several smaller clusters (figure 2.6).



(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

Figure 2.5: Cluster evolution by year from 1993-2003 for clusters 1-22 (as % of in-cluster papers).

Cl.	Size	Proposed cluster name	Five most common words (and counts)
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 Cryptographic 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), Distribute (533), Computing (524), 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)

Table 2.6: Cluster descriptions.

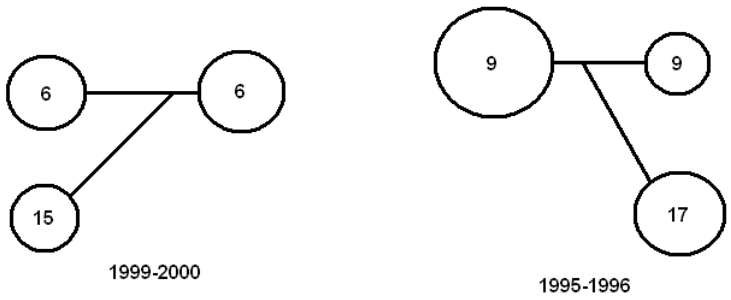


Figure 2.6: How clusters merge or split. Cluster 6 was labeled "Constrained Satisfaction" and cluster 15 "Optimization". Cluster 9 was "Internet Traffic Management" and 17 "Distributed Computing".

Most fragments were below the threshold of size and cohesiveness to be knowledge communities, consisting of a few loosely related papers, but there do remain remnants of the original cluster 9, and a new cluster, 17, which represents "distributed computing". Both are significantly smaller – 38% and 16%, respectively, of the size of the original cluster 9. In 2000 there was also a merger between clusters 6 and 15 (figure 2.6), representing the knowledge communities researching "constraint satisfaction" and "optimization", respectively. These fields are clearly related, and both clusters were approximately the same size. Cluster 15 was on the tail end of a gradual decline, and cluster 6 was recently formed and consistently growing, so we labeled the resulting knowledge community as cluster 6.

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%.

2.5 Predicting the growth of knowledge communities

Our goal, however, is not to gain insight into what happened historically (even though that is also interesting), but to predict what will happen going forward in a knowledge community and to define the attributes of successful communities. We will use the clusters we found in the previous section to analyze the growth of knowledge communities. We ask how the knowledge content, as measured by their citations, and rhetorical content, as measured by the vocabulary they used, affect the differential success of these communities [99, 112]. We also examine the effect of community characteristics of cohesion [91], uniqueness [53] and adaptability [82], as described below.

2.5.1 Model for community growth

We wish to find out how effectively the growth of knowledge communities can be predicted using 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 cluster at a given time, as measured by the number of papers presented at conferences or published in computer science journals in a cluster each year.

We estimated our models using Generalized Least Squares (GLS), including robust standard errors for determining statistical significance [86]. This approach allows us to investigate the time trends within our data while also adjusting our 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.

The empirical goal of our model is to explore the extent to which we can measure community attributes and use them to predict performance. The central question we face is how the community uses vocabulary and draws on knowledge, both of which are unrelated to the explanatory power of the community, to enable it to be successful. Specifically, we consider how the community draws on past knowledge and generates persuasive rhetoric by measuring the cohesiveness and uniqueness of both. More formally, we estimate our model as follows:

$$y_{it} = \beta x_{it} + b_i z_{it} + e_{it} \tag{2.16}$$

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.

Below we describe the features that we used: cohesiveness, uniqueness, rate of change, leadership/coordination controls, prestige controls and industry/academia controls. The feature values were normalized by their standard deviations.

Cohesiveness: We use this feature to measure how the intellectual "cohesiveness" of both the shared knowledge (papers cited) and shared rhetoric (words) of the knowledge community are significant for predicting its performance. For the Knowledge Cohesiveness variable we represent how widely the cluster as a whole searched for knowledge during that year in the intellectual landscape vs. how focused (coordinated) that search was. Knowledge Cohesiveness was computed as the average similarity between the citations of each paper and the overall citations of the cluster. More formally, it is

$$\sum_{i=1}^{n_C} \frac{\text{sim}(e_i, cen_C)}{n_C} \tag{2.17}$$

where C represents a cluster for a given year; i indexes papers in cluster C; n_C is the number of papers in cluster C and sim() is the measure of similarity as previously defined in the clustering methodology. This represents how widely (or narrowly) authors in the cluster searched for knowledge during that year. (Since clusters vary year to year, cohesiveness is for a given year.) Rhetorical Cohesiveness, which measures how similarly people in a cluster use vocabulary, was computed in the same way. In this case, the similarity of the stemmed words in the title and keywords of each paper to the average for its cluster was computed. As is common, stop words were also removed.

Uniqueness: We are also interested in how different an intellectual community is from other communities, either in the knowledge it generates or in the rhetoric (vocabulary) it uses. 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 school of thought are at a given point in time. The variable is computed in the same way as Uniqueness of Rhetoric, but using citation structure rather than words. For this feature we compare the average citations or vocabulary for a cluster to the average citations or vocabulary of all other clusters. For example, if a cluster generally uses the same keywords or cites the same papers, it will have a low "uniqueness."

Flexibility: Flexibility or adaptability for a cluster is an important measure of how much a cluster changes over time. We are interested in the tendencies of a cluster to change or remain stable. We assume that over time in a changing environment, flexible clusters move more than less flexible clusters. Since knowledge changes as a function of other knowledge, we use a relative measure of change in constructing this variable. Given our averages or centroids for citation structure and language per cluster, we construct a cosine similarity between each cluster and itself in the previous year. The difference between the cluster average from year t to t+1 is a

measure of the "rate of change" of a cluster over time. We computed the 3-year running average (smoothings over 1, 2 and 5 gave comparable results). The change in rhetoric represents how much the words that a cluster uses change from one year to the next; the change in knowledge represents how much a cluster's average use of citations changes from one year to the next.

Leadership/Coordination Controls: A common way to explain differential performance in firms is to look at the level of leadership or coordination. To attempt to measure leadership in knowledge communities, 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. A knowledge community may have very influential members who can act as intellectual leaders, coordinating the knowledge community's chief concerns, methodologies and areas of research. We identify influence ties between authors of papers and the authors of those papers that they cite and thereby construct an influence network for each cluster. We then run centrality measures on these networks to measure the clusters' eigenvector, degree, and in-degree centrality. We found eigenvectors to be the most useful measure of centrality, since it measures both direct and indirect influence, though all measures led to similar results.

We also wish to control for the potential coordinating influence of institutions. For example, a school such as MIT or a company such as Google might be home to a significant number of members of a knowledge community and thus the formal control, social network, institutional norms and institutional organization these institutions exhibit may contribute to the *de facto* coordination of the school of thought. To construct a variable to measure this we first identified the institutions that the authors in the database identified with in their papers. We then found the percent of papers for each cluster that came from the most common 10 schools, research institutions, or companies, in order to see if a cluster had concentrated influences by a few institutions (concentration of the top 1, 2, 3, and 5 institutions gave similar

results). We assume that a higher concentration of control by a few players in a knowledge cluster increases the potential for cluster coordination.

Lastly, we believe a coordinating or leadership role might be played if an intellectual community is dominated by a powerful venue that controls distribution for that cluster – such as a journal or conference that acts as a gate-keeper for the community. Such knowledge gate-keepers can implicitly or explicitly influence the level of homogeneity and coordination of a school of thought by both lending legitimacy to work (by certifying it has passed a rigorous review process) and making it available to an interested audience. In this case we look at the percent of articles published in the 10 most common venues of the authors (either journals or conferences).

Prestige Controls: Prestige is a powerful factor in explaining differential performance in organizations; we wish to test whether this also holds true of knowledge communities. As with leadership, we therefore control for prestige on the member, journal/conference, and employer/university levels of analysis. For members, we wish to control for the prestige that would result from the "top" members of a field preferring to publish in some intellectual communities leading to superior performance. We constructed this variable by finding the authors who had been nominated to the prestigious post of fellow by three top societies in computer science – the Institute for Electrical and Electronics Engineers, the Association of Computing Machinery, and the National Academy of Engineering – from 1975 to 2005 and counting the number of these fellows who published in any of our intellectual communities by cluster and year. Next we constructed a variable that counted the number of papers coming from the most prestigious 20 universities in computer science as ranked by the US News and World Report graduate school rankings of academic programs. By doing this we help control for the tendency for some intellectual communities to be associated with prestigious institutions. Lastly, we constructed a variable that counted the number of papers published in the top 10 most prestigious journals as ranked by impact factor in Thomson ISI's Impact Factors, which ranks the influence

of journals, and the top 10 most prestigious conferences, as ranked by citation impact by DBLP. We ranked these counts within year by cluster. This rank-ordered list of clusters by year indicated the relative prestige of knowledge communities on multiple levels.

Industry/Academia Controls: We encode each author of every paper as being affiliated with a firm or academic/research institution. We then encode 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 academic/research institutions. We entered this information into the regression by including the two categorical variables "mixed" and "industry".

2.5.2 Community prediction results

The parameter values of our trained model are shown in table 2.7. Looking at the magnitude, sign and significance of the parameters we find that cohesive rhetoric (low variance of vocabulary within a cluster) and a broad use of knowledge (high variance of citations within a cluster) are associated with improved performance. Also, a knowledge community maximizes performance when it uses vocabulary that is similar to that of other clusters and knowledge, as represented by citations, that is gathered from diverse sources. Community flexibility in citations predicts community growth, while changing vocabulary has the opposite effect (significant at $p < 0.05$). We did not find significant effect of concentration (leadership) in publications on community growth, either by source (school or institution) or by venue.

The variable for mixed industry/academy affiliation becomes statistically significant at the $p < 0.1$ level in this model. 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

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	

Table 2.7: Time Series GLS Estimation. The dependent variable is the number of papers published by a community in a given year. (*** $p < 0.001$; ** $p < 0.01$; * $p < 0.05$.)

industry-affiliated papers indicated a slightly negative, though statistically insignificant, impact on community performance. This suggests that the effect of higher proportions of purely academic-affiliated papers is indistinguishable from that of mixed-affiliation papers. 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 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. For knowledge community cohesiveness, regressing the similar measures for rhetoric on citations yielded an R^2 of 0.846, indicating that approximately 85% of the variation in rhetorical cohesiveness is attributable to changes in citation cohesiveness. Similarly, for knowledge community uniqueness the R^2 was 0.401, so approximately 40% of the variation in rhetorical uniqueness is explained by changes in citation uniqueness. Lastly, an R^2 of 0.9 for knowledge community flexibility indicates that about 90% of the changes in rhetorical flexibility are explained by corresponding changes in citation flexibility.

It is important to note that the trends identified by these measures are consistent throughout the dataset; the extremely high correlations are chiefly due to extreme values. When examining the same regression for knowledge community cohesiveness as above, but including only the central 80% of points, we found that a more reasonable 50% of the total variation in rhetorical cohesiveness is attributable to changes in citation cohesiveness. To ensure that these results retain their significance when controlling for the other previously identified covariates in our full model, we refit this model and saw that our relationships remained large, positive and statistically significant.

We also evaluated a variety of plausible model estimation methods, chiefly a Generalized Estimating Equations (GEE) approach, an explicit panel-data GLS model, and a Random Effects specification estimated via maximum likelihood, which confirm the robustness of our model [122].

Summarizing the results, we found that successful intellectual communities have systematic characteristics. They use knowledge and rhetoric, which are linked, 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.

To provide an intuitive example of how this analysis might be understood in a given environment we identified two clusters with similar recent growth histories, one of which is on the verge of growth while the other is on the verge of shrinking. We identified cluster 4, which focuses on the design of cryptographic systems, and cluster 13, which focuses on machine learning. For 1996 both clusters have similar numbers of papers and have remained stable from the prior years. Underlying this apparent similarity we observe that cluster 13 has sharply increased in its rhetorical cohesiveness and stability while becoming less rhetorically unique. It has also become more diverse in its use of knowledge while remaining stable in knowledge uniqueness and flexibility. Cluster 4 on the other hand has become much more rhetorically diverse, and grown more focused in its knowledge cohesiveness and decreased in knowledge flexibility. Overall, our analysis predicts that cluster 13 is primed for growth while cluster 4 is not. The performance of these clusters over the next 3 years bears this out, as cluster 13 grows approximately 79% from 1996 to 1999 while cluster 4 shrinks by about 12%. While this is a relatively extreme case, and our findings in this paper speak of average tendencies not necessarily applicable to

every case, this example illustrates how our methods might be used practically to understand a cluster's performance.

2.5.3 Discussion

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, 57% of citations are received by papers in clusters when only 44% of papers are in clusters. 76% of citations of papers in a cluster go 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 not in a cluster.

We looked at how knowledge communities use knowledge and rhetoric to help explain why some of these knowledge communities flourish and grow and found 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.

There is a question as to how 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 allows researchers to exchange ideas and collaborate more efficiently. Using terms that other communities know makes it easier to be understood by these communities, too.

Research in the field of Search/Positioning for knowledge development 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 your own community) [87]. We found that this tradeoff must also take into account the differences between the use of language and knowledge. The data 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 school and translated into a unified, explanatory, efficient and shared rhetorical framework.

Patterns of success and failure in science have explanatory consequences for the way science as a whole develops. For example, we were able to note the rise of the knowledge community for search technology in 1996 that preceded the growing importance of this technology in the evolution of computer science – a community that from its founding exhibited extremely focused rhetoric and very wide patterns of knowledge exploration. Further research into the differential success of knowledge communities can give us a better understanding of what guides the development and direction of innovation. Most importantly, continued understanding of the underlying causes of differential innovation in large-scale network structures should make it easier to encourage successful collaboration between researchers and improve the functioning of such communities and lead as well to an increase in the overall velocity of research and innovation.

This paper is unique in using clusters to build predictive models of how communities evolve and how position in a cluster or in the background predicts how widely cited a paper will become. Agglomerative clustering has been used in [56] and [55] to find co-citation communities that are strong and others that are essentially random, but the authors do not specifically use the notion of background in their clustering, and do not use their clusters in predictive models. Similarly, [101] searched

for temporal trends in hyper-linked document databases using clustering and [121] found communities that change with time in the setting of social networks, but neither build predictive models of community growth. [88] 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. [60] use both text mining and citation analysis to find communities in bioinformatics and track their evolution through time, again without making any future predictions.

2.6 Predicting the impact of individual papers

A second major question we look at regards modeling paper impact. How much will a paper be cited? To answer this, we again used a supervised learning model, this time to predict the number of citations a paper will receive from characteristics of the paper and its position in the cluster.

2.6.1 Model for paper impact

The data used for our analyses are non-negative counts of the number of citations received in future years. As with previous measures of publication citation counts, the data exhibit a variance in the number of citations larger than would be expected from a Poisson distribution. We considered using a simple negative binomial (NB) model to account for the excess variance [54, 1, 38]; however, the high number of papers that have received zero citations in our data further indicates that a zero-inflated negative binomial regression (ZINBR) model would be preferred [45, 123]. The use of two-stage ZINBR models is helpful when there may be a distinct process influencing the occurrence of a proportion of data points with the value of zero.

Our model was the following:

$$Pr(y = 0) = p + (1 - p)(1 + \frac{\lambda}{\alpha})^{-\alpha} \tag{2.18}$$

$$Pr(y > 0) = (1 - p)\frac{\Gamma(y + \alpha)}{y!\Gamma(\alpha)}(1 + \frac{\lambda}{\alpha})^{-\alpha}(1 + \frac{a}{\lambda})^{-y} \tag{2.19}$$

where p (the probability of a structural zero count i.e. no citations) and λ are modeled as

$$\ln(\frac{p}{1 - p}) = c_p + \sum_{i=1}^{I} a_i v_i \tag{2.20}$$

$$\ln(\lambda) = c_\lambda + \sum_{i=1}^{I} b_i w_i \tag{2.21}$$

v and w are the independent variables (the features in our model), a and b are the corresponding regression coefficients and c_p, c_λ are the regression constants (intercepts). Here v and w are labeled differently, though they coincide in our models. The over-dispersion parameter α is determined by the iterative maximum-likelihood procedure used to fit the model. Thus, the predicted mean number of citations for a paper, given its features, is $\lambda(1 - p)$. Note this is independent of α.

Unfortunately, only some of the articles had journal or conference information because of limitations in our data source – despite our attempt to make this variable more reliable by getting additional data from the DBLP Computer Science Bibliography. For the analysis in the previous section, aggregating the information on journals to see "journal coordination" and "journal prestige", a partial random sample was sufficient to differentiate between knowledge communities. But for a paper-by-paper level of analysis, since the majority of these data lack explicit journal or conference assignments and the rest were highly dispersed among a large number of journals and conferences, they would not have contributed meaningfully to the model. We therefore excluded this control variable from our regression. Other necessary data such as authors, years of publication, and bibliographies were available and sufficiently complete to use in the model. All other variables were generated using these component variables.

Our unit of analysis here is individual papers. The dependent variable measures the total impact of a paper as the number of citations it has received subsequent to its publication through 2003. Our features describe characteristics of papers, both descriptions of them and also how they relate to other papers published at that time. The features we used, which were normalized by their standard deviations, were the following:

Bibliography Size: The average bibliography size increased steadily over the time spanned by our data. To control for the increasing number of citations made in papers, we controlled for the number of entries in a paper's bibliography.

Year of Publication: We created dummy variables for each year from 1992 to 2003.

Coauthorship (Binary): We included a binary variable for whether a paper has more than one author (1 if coauthored, 0 if single author). This helps control for the differences in the process of joint and individual knowledge production.

Cluster (Binary): This binary variable was coded 1 if the paper was in a cluster of other similar papers (representing a knowledge community) when it was published and 0 if the paper was not in a cluster when it was published.

Distance: A paper is central to its cluster when its citation structure is very similar to the mean of the citation structure of all papers in its cluster (the centroid). Papers that are typical of their clusters will have small distances. Papers that differ from the rest of the group by citing outside sources or by citing uncommonly cited papers will have larger distances. We measured distance as the angle of the vector representations between the citation structure of a paper and the centroid of its cluster. Papers not in a cluster were not assigned a distance.

Because clusters are generated with data from the year of the paper and the 4 previous years, distance represents the centrality of the paper to its cluster historically at the time of publication, not the centrality of the paper after publication. We also included a second-order effect for distance.

Diversity of Publications: In order to find a proxy for an author's more general tendency to seek a diversity of knowledge and viewpoints, we counted the number of clusters in which an author has published throughout our dataset. We believe this gives an estimate of an author's tendency to stay in a school of thought or move between schools of thought. For papers with multiple authors we averaged their diversity measures.

Total Number of Papers Published: We counted an author's total number of publications, which is potentially correlated with the *Diversity of Publications* measure. For papers with multiple authors we summed their publication counts.

We tested the following three hypotheses:

- Hypothesis 1: New knowledge has more impact if it is within a knowledge community than if it is not.

- Hypothesis 2: A position toward the intellectual periphery of a knowledge community results in greater new knowledge impact.

- Hypothesis 3: Creators of new knowledge who actively engage in multiple/few schools of thought, over time, have greater impact.

To this end, we built three models, using a different set of features for each one. To capture as much of the variance in paper citations as possible before testing our hypotheses, we also created a Base Model that predicts a paper's total citations based on external paper and field characteristics without including any cluster-specific information. For the Base Model we constructed a ZINBR model and included bibliography size, coauthorship and year effects as our explanatory variables. The year effect is necessary because papers that are published earlier tend to have accumulated a greater number of citations. We therefore control for publication year with dummy variables. In predicting the inflated zero counts we utilized the same explanatory variables.

For Model 1 we built upon the Base Model to explore Hypothesis 1, which asks whether a paper benefits from membership in a cluster. To this end we augmented our Base Model with the binary cluster membership variable identifying whether or not a paper belongs to a cluster.

Model 2 investigates Hypothesis 2, which argues that a paper at the semi-periphery of its cluster is more likely to be highly cited. We included in the analysis, in addition to the Base Model, the variable's distance from center and the squared term of distance from center. Papers that are not in a cluster were excluded from consideration when fitting this model since they have no meaningful measure for distance. Consequently, the binary cluster membership variable used in Model 1 was not included in this model.

For Model 3 we adjusted Model 1 to account for the extent to which an author has benefited or has been harmed by publishing in many schools of thought (represented by clusters) throughout his or her career. We aimed to determine whether individual papers receive more citations if the author has a diverse experience with multiple or within few knowledge communities in our data – a proxy for an author's more general exploratory tendencies. To do this we included in the analysis a diversity measure to capture author publication diversity in addition to the independent variables included in Model 1.

2.6.2 Paper impact results

Table 2.8 shows the coefficients for the negative binomial component of the three models. The zero-inflated part (not shown) is qualitatively similar.

Our Model 1 reveals that we have constructed a sound basis for modeling the number of citations received by papers. The coefficients for all included independent variables were significant ($p < 0.001$) in both the zero-inflation and NB portions of the model (zero-inflated part not shown). Model 1 includes significant coefficients for cluster membership in both portions, fully supporting Hypothesis 1. Cluster

Model:	**Base**	**1**	**2**	**3**
Cluster (binary)		0.537 *		0.234 *
Distance			3.080 *	
Distance Squared			-3.527 *	
Diversity of Sources				0.526 *
Diversity of Sources Squared				-0.253 *
Year of Publication Dummies	sig. *	sig. *	sig. *	sig. *
Bibliography size	0.003 *	0.045 *	0.040 *	0.033 *
Coauthorship (binary)	0.103 *	0.149 *	0.143 *	-0.096 *
Pure Industry Affiliation	0.132 *	0.099 *	0.131 *	0.153 *
Mixed Industry/ Academy Affiliation	0.122 *	0.111 *	0.125 *	0.080 *
Cluster Sum				0.007 *
Constant	1.020 *	0.853 *	1.169 *	0.956 *
log(alpha)	1.307 *	1.348 *	1.353 *	1.339 *
Log-likelihood	-403,615	-429,260	-388,102	-347,678
# Observations	190,982	190,982	164,980	144,909

Table 2.8: Coefficients and significance values for the negative binomial models. (* $p < 0.001$, sig. = significant)

membership, on average, is associated with receiving 4.27 more citations, holding all other variables unchanged. Furthermore, there is a strong relationship between membership in a cluster and receiving zero citations.

When computing Model 2, we excluded from consideration all papers that were not assigned to a cluster, since they have no meaningful distance measure, and used a normalized measure of distance for papers that were within a cluster such that they range from 0.0 (very central) to 1.0 (extreme periphery). Model 2 includes our variable representing distance, allowing us to test Hypothesis 2. Within the NB portion of the model, the significant coefficients for distance squared and for distance indicate a potentially curvilinear relationship between distance and total citations. Within the zero-inflation portion of the model we found that both distance and

distance squared were non-significant. Increasing distance from the core of a cluster is initially beneficial and beyond a certain point a further increase in distance is associated with relatively fewer expected citations. Based on Model 2, moving away from the optimal point in the semi-periphery by two standard deviations towards either the core or the periphery and holding other variables constant, the number of expected citations decreases by 1.33. The variables held over from Model 1 retain significance in the same direction, leaving their interpretations unchanged.

Model 3 allows us to examine diversity of publications as a predictor of total citations. The initial results support the hypothesis that a diverse publication pattern does indeed lead to higher citations. Results are significant for both the NB and zero-inflation portions of Model 3. The positive coefficient in the NB portion indicates that increased diversity in an author's publication pattern, which is associated with the author's interaction with very diverse knowledge and perspectives, is associated with higher citation counts. Similarly, in the zero-inflation portion we found that the more diverse an author's citation pattern, the less likely he or she is to receive zero citations. On average, an increase in diversity by two standard deviations is associated with 0.86 fewer citations, holding all other variables constant.

2.6.3 Discussion

Functionally, knowledge communities provide "small world" advantages to the process of knowledge development. Communities provide the local dense connection networks that lend themselves to learning and reputation. At the same time, the incentives toward semi-periphery positioning encourage community boundary-spanning. Our findings nicely reinforce the small world findings in the arena of knowledge creation and provide a new perspective and additional explanatory analysis of the social and intellectual underpinnings of this process in the knowledge creation context.

By applying performance measures to positioning in and around knowledge communities, we reveal that where knowledge is positioned has a significant impact on its

performance. We found that new knowledge which is positioned within a community can expect to get a higher number of citations on average. Knowledge positioned in the semi-periphery of a community (representing knowledge that builds on a mix of knowledge common and unusual in that community) rather than at its center or periphery results also in additional citations. It appears that new knowledge was positioned by its creators under the stress of two search tensions – being a part of an identifiable community and simultaneously reaching beyond that community to draw on outside knowledge.

We believe that this sort of knowledge creation, where new knowledge developers share knowledge and coalesce into cohesive and distinct intellectual and social groups, is crucial in new knowledge development. We created a quantitative framework for analyzing a paper's positioning incentive structure that, when aggregated across all papers, shapes how knowledge develops. We speculate that the robust incentive structures we found in clusters are maintained through selection forces within the knowledge environment; clusters that encourage too much exploration lose their integrity and fail to develop strong internal paradigms, while clusters that are too internally focused may not attract sufficient attention or become too stagnant to gain momentum. Clusters that balance these two extremes in the way we describe seem to have survived to populate our dataset.

Our results agree with the broader research which argues that knowledge creation occurs through a moderated combination of exploration and exploitation [82]. While we support the basic premises of positioning theory, we take into consideration previously ignored key social dimensions of intellectual positioning. By looking at new knowledge creation from the author-level positioning perspective, we quantitatively test the actual benefits of membership and position with a knowledge community at time of publication for a creator of new knowledge.

2.7 Conclusions

Viewing communities as tight foreground clusters embedded into a diffuse background facilitates community analysis, since the future levels of citation of papers within communities are significantly higher than for those in the background. We used a new algorithm, Streemer, for this clustering, which has several useful characteristics. Streemer works by first finding a large number of potential or candidate clusters. Then it filters those and chooses the best in terms of their size, separation from neighboring clusters and density of points. Finally it assigns all the points to their nearest cluster or the background. Streemer finds dense foreground clusters embedded in a more diffuse background cluster. It compares favorably in speed to standard clustering methods such as k-means, while offering the advantage of not forcing items to be in a foreground cluster. When many items (e.g. papers) do not fit cleanly into a cluster, it is preferable not to distort the clusters by adding these "outliers" to them. 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. Streemer is useful 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.

We also compared Streemer to a simple modification to k-means, called background k-means, which also produces foreground and background clusters. Background k-means is as fast as k-means and shares most of its characteristics. It often performs as well as Streemer or even slightly better, but it is iterative and thus slower. Streemer has lower computational complexity because it requires only two passes over the data, one to find candidate clusters and one to assign the points to foreground/background. It also does not make as many restrictive assumptions about the cluster, like their size or number, as background k-means does.

In the second half of the chapter, we fitted supervised models, based on the knowledge communities that were found from clustering. A number of authors (e.g., [98, 73, 40]) have used models to predict the citation rates of papers as a function of various features, such as keywords or number of publications of the authors. Unlike this paper, they do not perform any clustering and they do not examine the effect of membership or position in a knowledge community. Using a variety of features we predicted the evolution of the communities and the number of citations a paper will receive. The coefficients of the features for the fitted models describe the properties of successful knowledge communities and of widely cited papers. 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. Regarding individual papers, we found that for a paper to be cited it must draw on work from multiple fields, but use the language of the field where it is published. Papers which are within knowledge communities have significantly more impact than those that are in the background.

For both knowledge communities and individual papers, these two properties, broad content and standard vocabulary, are related to success. Combined, they allow a paper to present new knowledge and extend the bounds of current research while explaining it using familiar vocabulary. This is perhaps 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. It is thus striking that knowledge communities vary significantly in how variable their use of vocabulary is, in how broadly they cite outside their field and in how rapidly they grow.

Chapter 3

Online Dirichlet Process Mixture Model for citation analysis

3.1 Introduction

Most clustering methods fall in one of two categories: they are either simple and fast or powerful and slow. For example, k-means is a simple algorithm and due to its simplicity is very fast. Latent Dirichlet Allocation (LDA [9]) on the other hand uses a complicated model, but its power comes with a cost because inference requires the use of variational methods with the respective computational burden. It would be beneficial then to have a clustering algorithm that falls in between these two categories, one that is fast enough to be used with large datasets and powerful enough to find clusters with few restrictive properties. In this chapter we present such a method. It processes the data online and uses a mixture model with a Dirichlet process (DP) prior. The online approach makes it fast and the DP mixture model allows the removal of certain assumptions about the clusters, such as their number and sizes.

Dirichlet Processes have been proposed and used in several kinds of models, including mixtures of distributions [103]. However, they have rarely been used with

online algorithms, even though they are amenable to that setting. Zhang et. al. [127] analyze a model with multinomial distributions for the mixture components. Other online methods that have been proposed are based on variational Bayes [106, 43], or particle filtering [31]. In this chapter we present an online DPMM that uses Gaussian component distributions and give an algorithm for the online learning of the component and DP parameters. In order to improve the quality of the found clusters, we also investigate two-pass clustering. Prior work gives a theoretical justification for employing a second pass for clustering and shows that subsequent passes offer only small improvement in the cluster quality [17]. This is in agreement with our empirical observations with Streemer and with one- vs. two-pass clustering on synthetic datasets, that a single-pass algorithm makes several wrong decisions for the early points it processes, due to unreliable estimates of the model parameters. These estimates affect the assignment decisions for the later points and the overall clustering can be poor. We propose the use of a second pass for improvement. In the first pass, the algorithm finds a large number of small clusters; in the second pass, these clusters are treated as weighted points which are clustered into fewer clusters.

When clustering documents it is customary to use cosine similarities instead of Euclidean distances [117]. The main reason is that the cosine similarity, due to being scale invariant, allows comparisons between documents with different counts of words or citations – what is important is the angle the vector representing the document forms, which depends only on the document composition and not the total counts. The probability distribution that corresponds to the cosine similarity is the von Mises-Fisher (vMF) distribution [83], which is used to model directional dissimilarities between vectors. However, there are difficulties in computing analytical estimates of the vMF concentration parameter [2], which make the use of vMF less attractive. For the algorithm presented in this chapter we use Gaussian distributions in our mixture model which simplify the computations for the updates.

The approach we propose has many similarities with Streemer (chapter 2) in

that it is a few-pass algorithm, it uses a mixture model of Gaussian distributions, which corresponds to a distance function and it can find variable cluster numbers and cluster sizes due to the DP prior. The similarities between this algorithm and Streemer are also highlighted in this chapter. The original description of Streemer in chapter 2 is based on the use of thresholds on distances. In order to emphasize the similarities, we will replace the distances with probability calculations and model the clusters with a mixture model. Under these assumptions we will give a probabilistic interpretation of the Streemer algorithm.

In the next sections we describe the model, the online algorithm and the parameter updates. We give a probabilistic interpretation for Streemer and conclude with a discussion on foreground/background clustering in online settings and other possible priors besides Dirichlet Processes that one can use.

3.2 The DP mixture model

The major components that define our approach are the Dirichlet process, the mixture components and their prior distributions.

3.2.1 Dirichlet process

The Dirichlet Process (DP) [33] is a distribution over distributions. It is parameterized by a base distribution G_0 and a concentration parameter α. A random distribution G is distributed according to $\mathcal{DP}(\alpha, G_0)$ if for any finite partition $(A_1, \ldots, A_r)$, the random vector $(G(A_1), \ldots, G(A_r)$ is distributed as a Dirichlet distribution:

$$(G(A_1), \ldots, G(A_r) \sim Dir(\alpha_0 G_0(A_1), \ldots, \alpha_0 G_0(A_r)) \tag{3.1}$$

Draws from the DP exhibit a clustering effect, because previously observed values have a non-zero probability of occurring again. The probability of the n-th draw,

conditioned on the previous $n-1$ draws is

$$p(\theta_n|\theta_1,\ldots,\theta_{n-1},\alpha,G_0) \propto \sum_{i=1}^{n-1} \delta_{\theta_i}(\cdot) + \alpha_0 G_0(\cdot) \tag{3.2}$$

This means that θ_n has a positive probability of being equal to one of the previous draws. Additionally, the more frequently a point is drawn, the higher the probability is that it will be drawn again in the future.

We can rewrite the previous equation using the number of times the draws take certain values. This gives rise to the view of the DP known as the Chinese Restaurant Process (CRP) [8]. Consider a Chinese restaurant with an infinite number of tables. Each draw θ_n corresponds to a customer. The first customer sits at the first table, whereas the n-th customer sits at a table drawn from this distribution:

$$p(\theta_n = j|\theta_1,\ldots,\theta_{n-1}) = \begin{cases} \frac{n_j}{\alpha+n-1} & \text{if table } j \text{ is occupied} \\ \frac{\alpha}{\alpha+n-1} & \text{if the table is empty} \end{cases} \tag{3.3}$$

where n_j is the number of previous customers sitting at table j.

3.2.2 The DP mixture model

The DP mixture model (DPMM) [79] can be thought of as a standard mixture model, with uncertainty about the prior distribution G. For observations $n = 1,\ldots,N$:

$$\mathbf{x}_n \sim f(\mathbf{x}_n|\boldsymbol{\theta}_n) \tag{3.4}$$

$$\boldsymbol{\theta}_n \sim G \tag{3.5}$$

$$G \sim DP(\alpha, G_0) \tag{3.6}$$

In our case, $\boldsymbol{\theta} = (\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and $f(\cdot|\boldsymbol{\theta})$ is the multivariate Gaussian density function with parameters $\boldsymbol{\theta}$:

$$f(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right). \tag{3.7}$$

The index n ranges over the observations and $\boldsymbol{\theta}_n$ corresponds to the parameters of the component that generated $\mathbf{x}_n$.

We will use a diagonal covariance matrix $\mathbf{\Sigma}$ with possibly unequal variances in different dimensions (heteroscedastic). The reason is that in high dimensions there is a problem in fitting all the parameters of a full covariance matrix due to lack of enough data. By restricting ourselves to diagonal $\mathbf{\Sigma}$ we reduce the number of parameters we must fit from $O(d^2)$ to $O(d)$.

The conjugate prior for the Gaussian distribution is a multivariate Gaussian - inverse Wishart [35]. The priors for $\boldsymbol{\mu}$ and $\mathbf{\Sigma}$ are conditionally independent. Specifically, given $\mathbf{\Sigma}$, the prior for $\boldsymbol{\mu}$ is a multivariate Gaussian with parameters $(\boldsymbol{\mu}_0, \mathbf{\Sigma_0})$. The prior for $\mathbf{\Sigma}$ is an inverse Wishart distribution with parameters (Ψ, M_0):

$$\mathbf{\Sigma} \sim W^{-1}(\mathbf{\Psi}, M_0) \tag{3.8}$$

The probability density function is

$$\frac{|\mathbf{\Psi}|^{M_0/2} \ |\mathbf{\Sigma}|^{-(M_0+d+1)/2} \ e^{-trace(\Psi\mathbf{\Sigma}^{-1})/2}}{2^{(M_0 d)/2}\Gamma_d(M_0/2)} \tag{3.9}$$

where d is the dimensionality of $\mathbf{\Sigma}$.

There are no hard rules for setting the values of $\boldsymbol{\mu}_0, \mathbf{\Sigma_0}, \mathbf{\Psi}, M_0$. Usually $\boldsymbol{\mu}_0$ is set equal to 0 or the mean of the observations in the data set $\bar{\mathbf{x}}$ (effectively centering the data). $\mathbf{\Sigma_0}$ and $\mathbf{\Psi}$ are set to large values (either in the absolute sense, or for example five times the sample covariance), so that the covariance prior is sufficiently non-informative. For similar reasons M_0 is usually chosen to be small [58].

Given a sample $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, the posterior distribution of $\mathbf{\Sigma}$ is again an inverse Wishart with updated parameters:

$$\mathbf{\Sigma}|\mathbf{X} \sim W^{-1}(\mathbf{A} + \mathbf{\Psi}, N + M_0) \tag{3.10}$$

where $\mathbf{A} = (\mathbf{X} - \bar{\mathbf{X}})(\mathbf{X} - \bar{\mathbf{X}})^T$ is N times the sample covariance matrix of $\mathbf{X}$. The posterior distribution of $\boldsymbol{\mu}$ is a Gaussian with parameters $(\frac{N_0\boldsymbol{\mu}_0 + N\bar{\mathbf{X}}}{N_0+N}, \frac{\mathbf{\Psi}+\mathbf{A}}{N_0+N})$.

The graphical representation of the DPMM is shown in figure 3.1. The base measure G_0 is our Gaussian - inverse Wishart prior and α is the concentration parameter of the DP that affects the probability of sampling a new value or one of

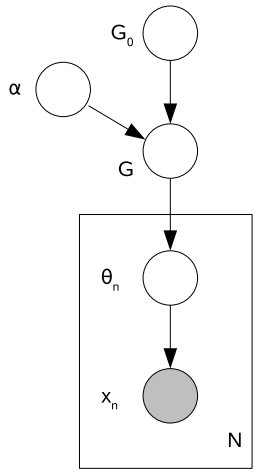


Figure 3.1: The Dirichlet process mixture model.

the previously sampled values. The distribution G consists of an infinite number of Dirac's delta functions from the base measure G_0, weighted by the DP according to the number of times each value appeared if we imagine infinite samples drawn. For each of the actual observations $\mathbf{x}_n$ we draw a parameter $\boldsymbol{\theta}_n$ from the sum of infinite delta functions that is G.

3.3 Online clustering algorithm

In the Chinese Restaurant Process interpretation of the Dirichlet Process, each arriving customer can be viewed as an incoming observation and each restaurant table as a mixture component. Under this view it is straightforward to develop an algorithm for online clustering that uses a DP as a prior. We first describe a 1-pass algorithm that clusters individual points and later add a second pass that clusters collections of points taken from the output of the first pass.

3.3.1 One-pass clustering

In the 1-pass algorithm (figure 3.2) the observations arrive one at a time and are assigned either to an existing or to a new cluster. The decision for the best assignment

is made based on which assignment improves the most the log likelihood of the data seen so far. After the observation is assigned, the cluster parameters are updated and the next observation is processed.

Let K be the number of clusters found so far, when observation $\mathbf{x}_i$ is about to be processed. The point $\mathbf{x}_i$ is potentially assigned to an existing cluster c_j ($j = 1, \ldots, K$), or a new cluster c_{K+1}. If $\mathbf{x}_i$ is assigned to c_j then the log likelihood of the data seen so far will change both because of the inclusion of the new observation and also because of the change in the model parameters due to the new observation. If $\mathbf{x}_i$ were to be assigned to c_j which already contains $|c_j|$ observations, the mean $\boldsymbol{\mu}'_j$ of c_j would become

$$\boldsymbol{\mu}'_j = \frac{N_0\boldsymbol{\mu}_0 + \sum_{\mathbf{x}_l \in c_j} \mathbf{x}_l + \mathbf{x}_l}{N_0 + |c_j| + 1} \tag{3.11}$$

and the covariance

$$\boldsymbol{\Sigma}'_j = \frac{\boldsymbol{\Psi} + \sum_{\mathbf{x}_l \in c_j} (\mathbf{x}_l - \boldsymbol{\mu}'_j)(\mathbf{x}_l - \boldsymbol{\mu}'_j)^T + (\mathbf{x}_i - \boldsymbol{\mu}'_j)(\mathbf{x}_i - \boldsymbol{\mu}'_j)^T}{M_0 + |c_j| + 1} \tag{3.12}$$

The probability of cluster c_j after the assignment of $\mathbf{x}_i$ becomes

$$P'(c_j) \propto \alpha |c_j|! \tag{3.13}$$

The denominator can be omitted because, due to the exchangeability of the DP, it is always the same regardless of the cluster to which $\mathbf{x}_i$ is assigned.

The log likelihood then is:

$$ll' = \sum_{j=1}^{K} \left[\left(\alpha + \sum_{l=1}^{|c_j|-1} l \right) + \sum_{\mathbf{x}_l \in c_j} \log f(\mathbf{x}_l | \boldsymbol{\mu}'_j, \boldsymbol{\Sigma}'_j) \right] + \text{const} \tag{3.14}$$

The first term is due to the cluster probability, the second due to the points assigned the cluster and the constant comes from the normalizing constants.

If, on the other hand, $\mathbf{x}_i$ starts a new cluster c_{K+1}, then that cluster will have

Input: A point set $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, priors $N_0, \boldsymbol{\mu}_0, \boldsymbol{\Psi}, M_0$ and DP parameter α.

For $i = 1 \ldots N$ do:

1. For every possible assignment of $\mathbf{x}_i$ to an existing or new cluster, compute (and store temporarily) the new cluster parameters:
   - If $\mathbf{x}_i$ was assigned to an existing cluster, use equations (3.11) - (3.12).
   - If $\mathbf{x}_i$ was assigned to a new cluster c_{K+1}, use equations (3.15) - (3.16).
2. For each potential assignment to a cluster, compute the new log likelihood ll', using equations (3.14) or (3.18).
3. Assign $\mathbf{x}_i$ to the cluster that will yield the highest log likelihood: $c(\mathbf{x}_i) = \text{argmax}_k ll'_k$.
4. Update the parameters for the selected cluster according to the computations in step 1.

Figure 3.2: One-pass online clustering.

parameters

$$\boldsymbol{\mu}'_{K+1} = \frac{N_0 \boldsymbol{\mu}_0 + \mathbf{x}_l}{N_0 + 1} \tag{3.15}$$

$$\boldsymbol{\Sigma}'_{K+1} = \frac{\boldsymbol{\Psi} + (\mathbf{x}_i - \boldsymbol{\mu}'_j)(\mathbf{x}_i - \boldsymbol{\mu}'_j)^T}{M_0 + 1} \tag{3.16}$$

and

$$P'(c_{K+1}) \propto \alpha \tag{3.17}$$

In this case the log likelihood would become:

$$ll' = \sum_{j=1}^{K+1} \left[\left(\alpha + \sum_{l=1}^{|c_j|-1} l \right) + \sum_{\mathbf{x}_l \in c_j} \log f(\mathbf{x}_l | \boldsymbol{\mu}'_j, \boldsymbol{\Sigma}'_j) \right] + \text{const} \tag{3.18}$$

The observation $\mathbf{x}_i$ will be assigned to the cluster that gives the highest new log likelihood ll' and the parameters will be updated as per the appropriate equations above.

Even though we have priors on the cluster parameters, we do not try to estimate

them. Prior work has dealt with these parameters by either introducing hyper-priors for them and estimating their hyper-parameters [103], or using empirical Bayes methods to find nonparametric point estimates [85]. Such techniques can be used in our case, too and we refer the reader to the cited works. We choose instead to treat the prior parameters $\alpha, \boldsymbol{\mu}_0, N_0, \boldsymbol{\Psi}, M_0$ as user-specified, both for simplicity and also for better correspondence to the Streemer algorithm. For example, the threshold in the first step of Streemer is equivalent to the DP parameter α. That threshold is set by the user (but could have been estimated from the data) and similarly α is also pre-set.

3.3.2 Two-pass clustering

Theoretical work [48, 17] and our empirical observations, both with Streemer as well as with the algorithms in this chapter applied to synthetic datasets, suggest that a single-pass algorithm makes several wrong decisions for the early points, due to unreliable estimates of the cluster parameters. These decisions affect the assignment decisions for the subsequent points and the overall clustering can be poor. A way to improve the quality of 1-pass online clustering is to execute two clustering passes over the data. In the first pass the algorithm finds a large number of small clusters; in the second pass, these clusters are treated as weighted points which are grouped into fewer clusters. Experiments show a significant improvement in the cluster quality when using the two-pass approach.

The second pass of our 2-pass clustering algorithm operates similarly to the first pass, so the algorithm given in figure 3.2 is fundamentally the same. The difference is that in the second pass, instead of processing individual points, the algorithm processes groups of points, where each group is a cluster found in the first pass. The parameter estimation equations (3.11) - (3.16) are modified to use in place of the single point $\mathbf{x}_i$ the sum of the points $\sum_{\mathbf{x}_l \in c_j} \mathbf{x}_l$ in the first-pass cluster c_j. The cluster probabilities use the number of points in the first-pass cluster as if all the

points from the first-pass cluster were assigned individually to the same second-pass cluster. Under these changes, the groups of points found in the first-pass are assigned to the (second-pass) cluster that will maximize the complete log likelihood.

3.4 Analysis

3.4.1 Probabilistic interpretation of Streemer

The operation of the Streemer algorithm is very similar to a two-pass online DPMM followed by a third pass to find the background. The original description of Streemer is based on the use of thresholds on distances, however the correspondence between distances and probabilities allows us to describe Streemer with respect to thresholds on probabilities arising from an appropriately defined model.

The first pass of Streemer (steps 1 and 2 in figure 2.2) finds a large number of candidate clusters. Each incoming observation is assigned to an existing cluster depending on its similarity to the cluster centroids. If the similarity is below a threshold then the observation forms a new cluster. This procedure is very similar to the single pass algorithm of figure 3.2. Instead of a similarity function of Streemer, this algorithm uses the log likelihood values from equation (3.14). Potential assignment to an existing cluster versus a new one is compared relative to the improvement in the log likelihood. A low similarity of an observation to current cluster centroids corresponds to a small improvement in the log likelihood.

The output of the first pass of Streemer is a set of candidate clusters. These clusters are then processed in the second step in sequence and are either kept and become the centroids of the final clusters, or they are dropped. The decision to keep them or not is based on a complicated heuristic that takes into account the size of the candidate cluster, its isolation from nearby candidate clusters and the cohesiveness of itself and the neighboring candidates. Once the final cluster centroids have been selected, the points are assigned to the most similar clusters. One thing

to note is that there is no re-estimation of the cluster centroids. The centroids of the final clusters are the centroids of the candidates that were selected in the second step. So, the second step of Streemer performs some sort of selection over the candidate clusters and the goal of the heuristic process is to pick the best centroids but avoid being overly greedy and risking getting trapped in local minima. This step of Streemer is emulated by the second pass of the DPMM clustering algorithm. The first pass returns a large number of candidate clusters, like Streemer does. These candidates are then treated as undivided collections of points clustered in the second pass. The final centroids are estimated by averaging all the points in the candidate clusters which were clustered together. The process by which the final centroids are computed is therefore different from Streemer. The overall goal however is the same, to examine a large number of candidates and use them to compute good final centroids. The DPMM algorithm attempts to avoid local minima not by cluster selection, but by pre-clustering the data in the first pass and reducing the number of clusters in the second.

After the final clusters have been found, Streemer executes the final step which generates a background cluster. The approach is quite simple and involves finding the points that are most dissimilar to their cluster centroids, removing them from their respective clusters and placing them in the background. The DPMM algorithm does not find a background as described, however a similar step as the last step of Streemer could be used to generate one without any substantial differences. If one wishes to avoid using dissimilarities for the background step it is possible to use the log likelihood instead. For example, we could compute the improvement in the log likelihood for the foreground clusters that would be achieved by assigning a point to the background. Then actually assign to the background the points that improve the log likelihood the most.

One final difference is the type of similarity used in Streemer. While the similarity function is not defined in Streemer, for all our experiments it was effectively the

cosine measure. Taking into account the equivalence between Bregman divergences (and consequently of the cosine) and exponential family distributions [3], the cosine similarity would correspond to a DP mixture model of von Mises-Fisher (vMF) distributions. The model presented in this chapter uses Gaussian distributions, because of the the analytical difficulties of computing estimates for the vMF and would therefore correspond to Streemer with a Mahalanobis or Euclidean distance.

3.4.2 Relation to Gibbs sampling and particle filtering

Our single-pass clustering algorithm presented earlier in this chapter has several similarities with both Gibbs sampling and particle filtering. These methods are commonly used for probabilistic inference and are easily applicable for models with conjugate priors such as the Gaussian mixture model of this chapter.

Gibbs sampling (as well as other Monte Carlo methods) can be used to estimate expectations from the posterior distribution of a DPMM. By sampling T points $\boldsymbol{\theta}^{(t)}$, $t = 1, \ldots, T$ from the posterior distribution of $\boldsymbol{\theta}$, we can estimate the predictive distribution of a new observation $\mathbf{x}_{n+1}$ as $(1/T)\sum_{t=1}^{T} f(\boldsymbol{\theta}_{n+1}^{(t)})$, where $\boldsymbol{\theta}_{n+1}^{(t)}$ is drawn from the posterior distribution G. There are several variations of Gibbs sampling in the context of DPMMs. The reader is referred to the excellent paper by Neal [93] and the references therein.

Here we will only look at one of them, where the cluster parameters are integrated out, eliminating them from the algorithm and leaving only the assignments of the observations to clusters $\mathbf{c} = (c_1, \ldots, c_n)$. This is possible because of the conjugate prior and desirable because it simplifies the algorithm. The state of the Markov chain in that case is just the c_i. Let $\boldsymbol{\phi}_c$ be the distinct values that the $\boldsymbol{\theta}_i$ take. The index c ranges from 1 to the number of clusters in the model at any given time. With Gibbs sampling, every point is assigned randomly to a cluster in turn, keeping the assignments of the other points fixed to their previously sampled values. The assignments of points to clusters are made according to the following probabilities:

$$P(c_i = c_j \text{ for some } j \neq i | c_{-i}, x_i) = b \frac{n_{-i,c}}{n-1+\alpha} \int f(x_i, \boldsymbol{\phi}) dG_{-i,c} \tag{3.19}$$

$$P(c_i \neq c_j \text{ for all } j \neq i | c_{-i}, x_i) = b \frac{\alpha}{n-1+\alpha} \int f(x_i, \boldsymbol{\phi}) dG_0 \tag{3.20}$$

Here, b is a normalizing constant, c_{-i} denotes every c_j except $j = i$, $n_{-i,c}$ is the number of c_j excluding c_i that are equal to c and $G_{-i,c}$ is the posterior distribution G of $\boldsymbol{\phi}$ based on G_0 and all the observations x_j in cluster c excluding observation x_i.

The above conditional probabilities are easily derived by multiplying the likelihood $f(x_i, \boldsymbol{\phi})$ with the conditional prior (treating observation i as the last observation, which is possible due to the exchangeability of the Dirichlet process) and integrating out the $\boldsymbol{\phi}$. The Gibbs sampling algorithm works by repeatedly sampling, for $i = 1, \ldots, n$, new values for c_i given c_{-i} and x_i according to equations 3.19.

Both Gibbs sampling and the the probabilistic version of Streemer which was proposed in this chapter assign the points to clusters one at a time. However, Gibbs sampling is a batch algorithm and uses all the points except the one under consideration, whereas our algorithm uses only the points seen so far.

The assignments in Gibbs sampling use the conditional probabilities for c_i given the other assignments and observations and the actual assignment is decided probabilistically. Every potential assignment has a probability of being selected in the current or a subsequent sampling and thus the space of assignments will be eventually visited completely, given enough iterations. Our algorithm, on the other hand, makes the assignment decisions based on the change in the log likelihood after the assignment and the algorithm's decisions, even though they are based on probabilities, are deterministic. The algorithm behaves greedily, looking for the mode in the log likelihood, which makes it much faster but also likely to get stuck in a local maximum. Also, once the assignments are made they are never revisited, regardless of what the new data arriving afterwards would dictate. This is necessary because of the online nature of our algorithm; the second pass was introduced to ameliorate

the negative effects of not revisiting the cluster assignments.

Another disadvantage of Gibbs sampling and other Markov chain Monte Carlo (MCMC) methods is that they are not well-suited to online settings. As each new observation arrives, the posterior distribution changes and the MCMC algorithm should be run again to obtain representative samples from it. However MCMC methods require several iterations, which makes their continuous execution after every point computationally costly. As a result of this problem, sequential Monte Carlo methods have been developed, which are known as particle filters [31].

A particle filter uses a set of weighted particles to approximate the posterior distribution. The particles and their weights are updated sequentially as each new observation arrives. In the DPMM setting with conjugate priors, it is again possible to integrate out the cluster parameters. In this case every particle is just a specific assignment of the observations to clusters. The number of particles P is kept constant during the execution of the algorithm. Assume that there are K clusters when a new observation is considered. This observation can be assigned to one of the existing or a new cluster, giving rise to $K+1$ possibilities. Each particle then generates up to $K+1$ potential particles. In order to keep the number of particles from increasing exponentially a resampling step follows which reduces the number of particles to P and updates their weights so that the approximation of the posterior remains close.

Specifically, after seeing observation $\mathbf{x}_n$, the particle filter approximates the posterior distribution $p(c_n|\mathbf{x}_1,\ldots,\mathbf{x}_n)$ by a set of P particles $\{\mathbf{c}_n^{(i)}\}$, $i=1,\ldots,P$, with respective weights $\mathbf{w}_n^{(i)}$ normalized to sum to 1. The posterior $p(c_n|\mathbf{x}_1,\ldots,\mathbf{x}_n)$ is approximated by a discrete distribution that takes as values the particles with probabilities proportional to the particle weights. With this approximation, the expectation of any function $g(c_n)$ can be computed as

$$E[g(c_n)|\mathbf{x}_1,\ldots,\mathbf{x}_n] \approx \sum_{i=1}^{P} w_n^{(i)} g(\mathbf{c}_n^{(i)}) \tag{3.21}$$

For example, the posterior is estimated as follows:

$$p(c_1, \ldots, c_n | \mathbf{x}_1, \ldots, \mathbf{x}_n) \approx \sum_{i=1}^{P} w_n^{(i)} p(c_1, \ldots, c_n | \mathbf{c}_n^{(i)}, \mathbf{x}_1, \ldots, \mathbf{x}_n) \tag{3.22}$$

When the next observation x_{n+1} arrives, for every particle $\mathbf{c}_n^{(i)}$ with K_i distinct clusters there are K_i+1 potential assignments for the new observation. Thus the posterior $p(c_1, \ldots, c_n, c_{n+1} | \mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{x}_{n+1})$ is approximated by a discrete distribution that can take as values the new particles $\{\mathbf{c}_n^{(i)}, j\}$, $j = 1, \ldots, K_i+1$ with probabilities respectively proportional to

$$w_n^{(i)} \frac{p(\{\mathbf{c}_n^{(i)}, j\} | \mathbf{x}_1, \ldots, \mathbf{x}_{n+1})}{p(\{\mathbf{c}_n^{(i)}\} | \mathbf{x}_1, \ldots, \mathbf{x}_n)} \tag{3.23}$$

This approximation is basically based on $\sum_{i=1}^{N}(K_i+1)$ new particles. These new particles are resampled to reduce their number to P. There are several proposed methods to perform the resampling. The simplest is the keep the P particles with the largest weights. Another possibility is to sample from them according to their weights. More information about resampling algorithms are given by Fearnhead [31] and the papers he references.

Particle filters were developed for analyzing dynamic problems where the state changes over time and they are appropriate for online clustering with a DPMM. They process the observations sequentially and the updates are not prohibitively expensive as in the case of MCMC. A particle filter maintains P assignments of all the points so far, which constitute the particles and uses them to approximate the posterior as a weighted sum. In comparison, the first pass of our algorithm, as was described earlier in this chapter, only keeps one assignment, the final assignment of points to clusters with an implicit weight of 1. It is thus similar to a particle filter with a single particle. As in the case of MCMC, the P particles permit the computation of approximations of the posterior or functions thereof, while our online algorithm is mode-seeking and cannot be used for similar computations.

Both particle filters and our algorithm do not change the assignments of previous

points when new points are considered. Of course this is only true for the particles and the posterior $p(c_1, \ldots, c_n | \mathbf{x}_1, \ldots, \mathbf{x}_n)$ can change drastically as the particle weights are being updated. MCMC algorithms on the other hand will theoretically visit every potential assignment given enough time.

Finally, the assignments of points to clusters in the particles are made according to the sampling step. While this can be deterministic for the simplest algorithm, in most cases the assignments are decided stochastically using importance sampling or some variation. This may help avoid the problem of local extrema that our algorithm faces, at the expense of having to keep a large enough number of particles. Our algorithm on the other hand makes the assignment that will improve the log likelihood the most in a deterministic fashion.

3.5 Discussion

Unlike other methods for citation analysis [80, 90], ours is, to our knowledge, the first principled probabilistic online approach. We combine a mixture model and a DP prior with an online, few-pass algorithm to achieve fast clustering with fewer structural restrictions than k-means. The model we present makes use of Gaussian distributions with full covariance matrices. While this is fine for the purpose of exposition, it may cause problems when used in practice. The reason is that many real datasets are high-dimensional. For example, our citation dataset lies in a space with more than 100,000 dimensions. For a D-dimensional dataset, the covariance matrix $\mathbf{\Sigma}$ of the components would be $d \times d$ and the clustering algorithm would be attempting to fit $O(d^2)$ parameters. For our citation dataset that would mean fitting hundreds of billions of parameters, something impossible to achieve with a few million data points without severe overfitting. When the dimensionality of the data is a problem, one solution is to use a simplified $\mathbf{\Sigma}$. One simplification is to assume that the features are independent and use a diagonal covariance matrix, thus reducing

the number of parameters to $O(d)$. If one wants to use even fewer parameters, $\mathbf{\Sigma}$ can be modeled with a single parameter as $\sigma^2\mathbf{I}$, making all the clusters spherical.

Another potential implementation issue has to do with the probability distributions used and how well they match the actual data. Our citation data consist of Boolean vectors whose coordinates are non-negative and the cosine similarities between them (and the computed cluster centroids) range between 0 and 1. All the vectors therefore (as well as the cluster centroids) lie on the surface of the unit hypersphere and specifically in the first orthant. Under these conditions, a mixture model of von Mises-Fisher distributions would be the most appropriate [16, 2]. Of course one *could* use the given model with Gaussian instead of vMF components, but would incur some penalty in the modeling accuracy.[1].

An issue with online clustering algorithms is that a single pass cannot distinguish whether a point that differs from the rest belongs to a new foreground cluster or the background. An algorithm must execute at least two passes in order to cluster points as belonging to the foreground or the background. The first pass can accumulate statistics about the distribution of points and the second pass, given that information, can decide whether a point falls near a component or not. In our work we have treated the foreground/background filtering as an add-on step and thus both the algorithm in this chapter augmented with a background filtering post-processing step, as well as Streemer, would perform three passes over the data. As future work, it may be possible to execute a first pass that will find a large number of potential clusters and also learn the distribution of the data. The second pass then will have all the information required to both cluster the results of the first pass as well as filter ill-fitting points to the background.

[1]It would be beneficial in the Gaussian case to normalize the data vectors to have norm one and thus lie on the unit hypersphere. Then the estimates for the mean in both the Gaussian and the vMF distributions are the same, except for the lack of normalization in the Gaussian. If normalization is applied every time the centroid vectors are estimated, then the two distribution estimates will be identical. The estimates for the variance in the Gaussian and the related concentration parameter in the vMF will be different, however.

Other directions for future work lie in prior choices other than the Dirichlet process. If we forget for a moment its requirement for a fixed number of clusters, k-means tends to find clusters of equal sizes. DPs on the other hand find clusters with a distribution similar to a power-law and other types of processes, such as the Pittman-Yor process [100], find different size distributions. Depending on the problem at hand and the expected cluster sizes for that problem, one could modify our algorithm to use a different prior that "best" matches the data.

Chapter 4

Multi-way clustering

4.1 Introduction

Word clustering can be used to automate, or partially automate tasks such as generating semantic classes, term-sets, or ontologies such as MeSH [92]. These sets of related terms are then used to index documents (e.g., in Medline), as features for entity tagging [105], or for relation extraction [13]. There has been extensive work in automating the process of generating term-sets, both in general natural language processing [77] and specifically for biomedical and bioinformatic uses [30, 126]. These methods typically cluster words under the "distributional similarity" assumption that words that occur in the the same contexts are semantically related [125, 76, 108]. In this chapter, we present a method for word clustering that combines information from different types of relations in which words occur (e.g., co-occurrence with verbs, and with other nouns) and achieves greater coverage of words and qualitatively different clusters. The resulting word clusters can be used for relation extraction [27], term identification [71], or for creating a new ontology from scratch in a new field (or subfield) [65, 110]. The clusters we obtain are broader than the usual MeSH categories and may be more useful for tasks such as relationship extraction.

Common clustering methods, such as k-means, agglomerative clustering or spectral methods are procedures designed for clustering a single variable. Methods that can cluster two variables simultaneously are known as biclustering and co-clustering algorithms [81, 22]. These cluster, for instance, nouns by verbs and verbs by nouns. Because clusters of the first variable depend on clusters of the other and vice versa, biclustering can reveal more information by showing the association structure between the two types of clusters. For cases where we want to cluster based on data from multiple sources, we need multi-clustering algorithms, such as [5, 39], that can cluster multiple variables.

Our contribution is a mixture model-based approach for finding word clusters by integrating information from different data sources. Specifically, we concentrate on integrating the co-occurrence information of word pairs extracted using different patterns, such as noun-noun and verb-noun co-occurrences. Since not all words appear in every type of pattern, taking advantage of side information helps both to improve (or as discussed below, at least alter) the quality of the clusters and also to expand the coverage of clustered words. Additionally, the different sources of information introduce new relationships between terms and the resulting clusters capture concepts that are different from those found only from one type or co-occurrence pattern.

The usefulness of using side information for extended coverage, is exhibited in the dataset shown in Table 4.1. The dataset was constructed by extracting two kinds of pairs: one of the form (verb, noun) and another of the form (noun, noun). Even though the two datasets are very different, each kind of relationship expands the coverage of nouns for both datasets. The nouns that appear in both types of pairs can be used to provide the common "ground" used to combine the information for the uniquely appearing nouns. If one attempts to cluster the nouns using only the verb-noun, or only the noun-noun pairs, a large number of them will be missed. This is because not all nouns appear in a single type of co-occurrence relationships. For

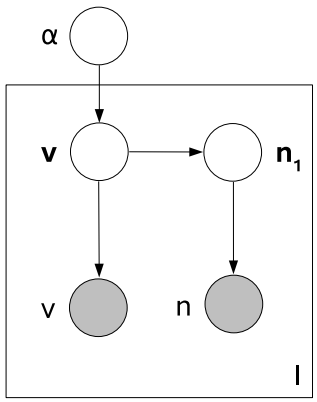
Dataset	Nouns in verb-noun set	Nouns in both sets	Nouns in noun-noun set
Medline	549	7602	1357

Table 4.1: Coverage of nouns by the two types of pairs.

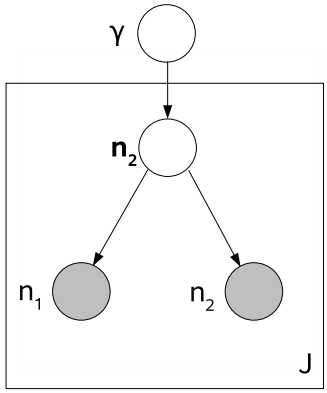
example, if we use the verb-noun pairs to cluster nouns, we will miss 1,357 nouns. If we use the noun-noun pairs, we will miss 549 nouns. By using both kinds of pairs we can find clusters with extended noun coverage.

Our algorithm is based on a combined hierarchical model consisting of two simple models, similar to belief nets (BNs). In order to make simultaneous use of the two tables of data that are available, the two models are fused, by forcing them to have the same parameters for certain conditional probabilities, according to the problem-specific representation. The difference from other multi-clustering algorithms is that our proposed method is based on likelihood maximization of a hierarchical model and is thus simple, principled and quite fast. In contrast, the method of [5] maximizes the mutual information between pairs of variables by using a user-defined clustering schedule to perform agglomerative and divisive clustering for different subsets of the variables. The multivariate information bottleneck [39] is more similar to our method, as it is based on Bayesian networks, but it relies on maximizing mutual information under constraints defined by a second network. Both networks have to be specified by the user.

The rest of the chapter is structured as follows: the next section presents the model used in the experiments and derives the EM algorithm with the introduction of the tied parameters. Section 4.3 describes the data we used and how the results were evaluated. Section 4.4 describes the experiments we ran and compares the results of clustering with different amounts of extra information, as well as with k-means.



(a) The V-N part of the full model. Also the standard V-N model



(b) The N-N part of the full model. Also the standard N-N model.

Figure 4.1: The full model.

4.2 Method

4.2.1 Data format

Our model is designed for data that are in the form of instances of co-occurrence pairs. For the experiments presented later we use two kinds of pairs: a set of verb-noun pairs and a set of noun-noun pairs. How these were extracted from text is described in section 4.3. The model we describe next is therefore tailored to them, but is easily generalized to other datasets.

The dataset can be viewed as two sparse tables, the first having verbs corresponding to rows and nouns to columns and the second having nouns as both rows and columns. An element (i, j) of either table is the number of times n_{ij} that the words corresponding to row i and column j co-occur. In section 4.3 we describe the methodology we used to extract our pairs. The data tables contain I instances of verb-noun pairs and J instances of noun-noun pairs.

4.2.2 Model and notation

The idea behind our method is to use standard generative models to encode the information derived from the dataset. The number and structure of these models depends on the number and type of data tables that are available. Each data table corresponds to a submodel, whose structure is defined by the actual variables that appear in the dataset. For example, in our case, the first data table contains co-occurrence counts of a verb and its direct object, so we created a hierarchical submodel with a dependence of the noun cluster on the verb cluster (figure 4.1(a)). Our second data table contains co-occurrences of nouns in appositions, conjunctions and disjunctions, so we put a single noun cluster generating both nouns in the pair (figure 4.1(b)). Once the submodels have been chosen, the parameters that correspond to the same variables in the data tables are tied, thus achieving the combining of the information contained in the data tables. The probabilities of generating a noun given a noun cluster are constrained to be the same in both submodels. The compound model we used for our experiments is shown in figure 4.1. This "full model" consists of the two separate submodels described previously, one for the verb-noun pairs, which we call the "standard V-N model", and another for the noun-noun pairs, which we call the "standard N-N model".

The random variables we use are defined as follows: $\mathbf{V}$ takes values from the set of verb clusters, $\mathbf{N}_1$ and $\mathbf{N}_2$ take values from the set of noun clusters, V from the set of verbs and N, N_1, N_2 from the set of nouns. We use bold font for variables associated with clusters and regular font for variables associated with observations.

The parameters of the model corresponding to the conditional probabilities are:

$$P(\mathbf{V} = \mathbf{v}) = \alpha_{\mathbf{v}} \tag{4.1}$$

$$P(\mathbf{N}_1 = \mathbf{n}|\mathbf{V} = \mathbf{v}) = \beta_{\mathbf{vn}} \tag{4.2}$$

$$P(V = v|\mathbf{V} = \mathbf{v}) = \theta_{\mathbf{v}v} \tag{4.3}$$

$$P(N = n|\mathbf{N}_1 = \mathbf{n}) = \psi_{\mathbf{n}n} \tag{4.4}$$

$$P(\mathbf{N}_2 = \mathbf{n}) = \gamma_{\mathbf{n}} \tag{4.5}$$

$$P(N_1 = n_1|\mathbf{N}_2 = \mathbf{n}) = \psi_{\mathbf{n}n_1} \tag{4.6}$$

$$P(N_2 = n_2|\mathbf{N}_2 = \mathbf{n}) = \psi_{\mathbf{n}n_2} \tag{4.7}$$

The parameter ψ, representing the conditional probability of a noun given a noun cluster, is the same in both parts of the model, thus achieving the desired coupling.

To generate a verb-noun pair from this model, we pick a verb cluster $\mathbf{v}$ according to probabilities α, a noun cluster $\mathbf{n}$ given $\mathbf{v}$ according to $\beta_{\mathbf{v}}$ and then generate a verb from $\mathbf{v}$ with probability $\theta_{\mathbf{v}}$ and a noun from $\mathbf{n}$ with probability $\psi_{\mathbf{n}}$. To generate a noun-noun pair, we pick a noun cluster $\mathbf{n}$ according to probabilities γ and then pick two nouns from $\mathbf{n}$ with probabilities $\psi_{\mathbf{n}}$. Thus, the nouns in both parts of the model are generated using the same probability distribution.

4.2.3 The EM algorithm

The model is estimated using the EM algorithm, which we now derive. The implementations for the separate V-N and N-N models are not shown, as they follow trivially. The variables V, N, N_1, N_2 are observed and $\mathbf{V}, \mathbf{N}_1, \mathbf{N}_2$ are treated as unobserved. The most interesting part is the update for the parameter ψ. It includes the counts of co-occurrences from both datasets (verb-noun and noun-noun) which are summed together. The result therefore uses an equivalent weighting proportional to the sizes of the two datasets.

Given observations of pairs $\{(v_i, n_i),\ i = 1 \ldots I\}$ and $\{(n_{1j}, n_{2j}),\ j = 1 \ldots J\}$ the

log-likelihood function for the complete data is

$$
\begin{aligned}
l &= \sum_{i=1}^{I} \log P(v_i, n_i, \mathbf{v}_i, \mathbf{n}_{1i}) + \sum_{j=1}^{J} \log P(n_{1j}, n_{2j}, \mathbf{n}_{2j}) \\
&= \sum_{i=1}^{I} (\log \alpha_{\mathbf{v}_i} + \log \beta_{\mathbf{v}_i \mathbf{n}_{1i}} + \log \theta_{\mathbf{v}_i v_i} + \log \psi_{\mathbf{n}_{1i} n_i}) \\
&\quad + \sum_{j=1}^{J} (\log \gamma_{\mathbf{n}_{2j}} + \log \psi_{\mathbf{n}_{2j} n_{1j}} + \log \psi_{\mathbf{n}_{2j} n_{2j}}).
\end{aligned} \tag{4.8}
$$

It is convenient to rewrite the log-likelihood using the number of times $\#_x$ that a random variable x takes a specific value. Let N_{VC} and N_{NC} denote the number of verb and noun clusters respectively and N_V and N_N denote the numbers of distinct verbs and nouns respectively that appear in the co-occurrence pairs. Computing the E-step is now straightforward; we use o to denote the set of observations:

$$
\begin{aligned}
E[l|o] = &\sum_{\mathbf{v}=1}^{N_{VC}} E[\#_{\mathbf{v}}|o] \log \alpha_{\mathbf{v}} + \\
&\sum_{\mathbf{v}=1}^{N_{VC}} \sum_{\mathbf{n}_1=1}^{N_{NC}} E[\#_{\mathbf{v}\mathbf{n}_1}|o] \log \beta_{\mathbf{v}\mathbf{n}_1} + \\
&\sum_{\mathbf{v}=1}^{N_{VC}} \sum_{v=1}^{N_V} E[\#_{\mathbf{v}v}|o] \log \theta_{\mathbf{v}v} + \\
&3 \sum_{\mathbf{n}_1=1}^{N_{NC}} \sum_{n=1}^{N_N} E[\#_{\mathbf{n}_1 n}|o] \log \psi_{\mathbf{n}_1 n} + \\
&\sum_{\mathbf{n}_2=1}^{N_{NC}} E[\#_{\mathbf{n}_2}|o] \log \gamma_{\mathbf{n}_2}.
\end{aligned} \tag{4.9}
$$

The expected counts are computed by summing (over all observations) the probabilities of the corresponding variables given the observations. The exact computations are omitted as trivial. The only noteworthy calculation is for the expected number of times $E[\#_{\mathbf{n}_1 n}|o]$ that the variables $\mathbf{N}_1$ and N take some specific values

$\mathbf{n}_{1k}$ and n_k, which involves summing over both datasets:

$$E[\#_{\mathbf{n}_1 n}|o] = \sum_{i=1}^{I} P(\mathbf{N}_1 = \mathbf{n}_{1k}, N = n_k|v_i, n_i) + \sum_{j=1}^{J} P(\mathbf{N}_2 = \mathbf{n}_{2k}, N = n_k|n_{1j}, n_{2j}). \quad (4.10)$$

Given the expected counts, the M-step updates are then straightforward.

$$\alpha_a = \frac{E[\#_a|o]}{I} \quad (4.11)$$

$$\beta_{ab} = \frac{E[\#_{ab}|o]}{\sum_b E[\#_{ab}|o]} \quad (4.12)$$

$$\theta_{av} = \frac{E[\#_{av}|o]}{\sum_v E[\#_{av}|o]} \quad (4.13)$$

$$\gamma_c = \frac{E[\#_c|o]}{J} \quad (4.14)$$

$$\psi_{cn} = \frac{E[\#_{cn}|o]}{\sum_n E[\#_{cn}|o]} \quad (4.15)$$

4.2.4 Properties

Using two separate submodels with tied parameters allows side information to be of arbitrary size and dimension and to reside in a different feature space. This makes it easy to incorporate extra information (in the form of extra tables), by adding separate submodels. The user only needs to decide on the model structure and what parameters should be tied and then use existing methods, such as the EM algorithm, to fit the parameters.

The second advantage of the combined submodels is that they trivially allow the data tables to contain different subsets of items (i.e. words) that are clustered. As long as there are common items between pairs of tables, the algorithm will be able to use them as connections between the data tables and combine the information contained in them. The end result will be a clustering of the union of items contained in the given tables, giving greater coverage.

4.3 Experimental setup

4.3.1 Dataset

We generated our dataset ("MEDLINE") by parsing 1,800,547 abstracts from the MEDLINE database, ranging from years 1995 to 2000, with the MINIPAR parser [78]. MINIPAR can be configured to output a sequence of "dependency triples" that represent shallow syntactic configurations between words. A dependency triple has the form (w_1, rel_r, w_2) where w_1 and w_2 are words in a sentence that engage in some syntactic relation rel_r. We extracted two types of relations. The first was verb-direct object, giving us the set of verb-noun pairs. The second was noun-TYPE-noun, where TYPE could be apposition, conjunction or disjunction, giving the set of noun-noun pairs. From the extracted verbs and nouns we selected the 1000 most common verbs and 1000 most common nouns as our vocabulary and randomly chose 500,000 pairs of verbs and nouns and an equal number of noun-noun pairs. To simulate the situation shown in table 4.1 in a systematic and easily quantifiable way, we also constructed a reduced noun-noun table by randomly removing 250 of the 1000 nouns as well as all the noun-noun co-occurrence instances where one or both nouns were in the removed subset of 250 nouns. We thus artificially created a reduced noun-noun dataset, which does not cover all the nouns and used it to test how extra information about the missing 250 nouns (in the form of the verb-noun table) can help.

4.3.2 Evaluation

The clusters were evaluated using class labels. We labeled the verbs and nouns by mapping them to WORDNET [32] and using the hypernyms of the synsets they mapped to as labels. The nouns were also mapped to MeSH [92] and labeled by their grandparent node. The mappings gave us multiple labels per word for most words. Each cluster was represented by a distribution of labels, found by creating a

histogram of label occurrences for the (labeled) words in the cluster.

These label distributions were in turn used by the evaluation measure, which was the weighted average entropy (WAE), defined as the average of the label distribution entropies for the clusters weighted by the cluster sizes [34]:

$$WAE = \sum_{i=1}^{C} \frac{n_i}{N} E_i \tag{4.16}$$

where n_i is the number of items in cluster i, N is the total number of items clustered and E_i is the entropy of the distribution of labels Y for cluster i. The lower the value of WAE, the better the clustering matches the given labels of the data.

We also used the label log likelihood (LLL) as a second measure of quality. For LLL, each labeled word is treated as an instance generated from a multinomial distribution, i.e. the label distribution of the cluster where the word was assigned to. We compute the LLL by summing the log likelihoods for every such instance. We also hand-labeled the words in the largest clusters (those containing more than 3% of the words) as either belonging or not in the clusters. To avoid any selection bias this labeling was done without knowledge of which algorithm generated which cluster. This gave us a Bernoulli distribution of "good" and "bad" words in each cluster and we computed the WAE of these distributions. Because the "bad" words were always less than half of the words in every cluster, a low entropy value corresponds to mostly "good" words in the cluster. Therefore clusterings with low WAE are preferred. The results for both the LLL and the manual labeling were in agreement to the WAE results and are thus not shown.

4.4 Results

We performed a number of different experiments and used k-means as a standard measure of comparison. In all experiments we used 25 verb clusters and 50 noun clusters, as we found these numbers to give a reasonable tradeoff between cluster

coherence and diversity. The EM algorithm was initialized with the output of k-means applied to the verb-noun data table only. Because EM gives soft clusters, we performed a hardening procedure at the end of the clustering, so that every word was assigned to one and only one cluster. According to this procedure, each word was assigned to the cluster with the highest posterior probability $P(cluster|word)$, computed from the model parameters using Bayes rule:

$$P(cluster|word) \propto P(word|cluster)P(cluster) \tag{4.17}$$

We examine the performance of our method in finding verb and noun clusters in three general cases. In sections 4.4.2 to 4.4.4 we find and evaluate clusters for all 1,000 nouns, for the reduced set of 750 nouns, and for all verbs.

4.4.1 Nature of clusters found

Before comparing the effects of combining noun-noun and verb-noun data, we briefly describe the nature of the clusters we found. The clusters we obtained included many words not in MeSH, or words that are only part of largely unrelated terms, for example *C., J., one, part, constant, intermediate*. We also found clusters whose terms were not in MeSH, for instance the cluster {I, ii, iii, iv, v}. Most of the clusters were broader than MeSH categories (e.g., one cluster contained terms such as *protein, receptor, peptide, antibody*). Thus, the entropies of the clusters are artificially high. Nevertheless, the quality of the clusters (as subjectively assessed) corresponded with their entropies, as measured using MeSH or Wordnet.

4.4.2 Clustering 1,000 nouns

For the first set of experiments we compared the noun clusters found by EM and using:

1. only the verb-noun data and the V-N model (results denoted as VN),

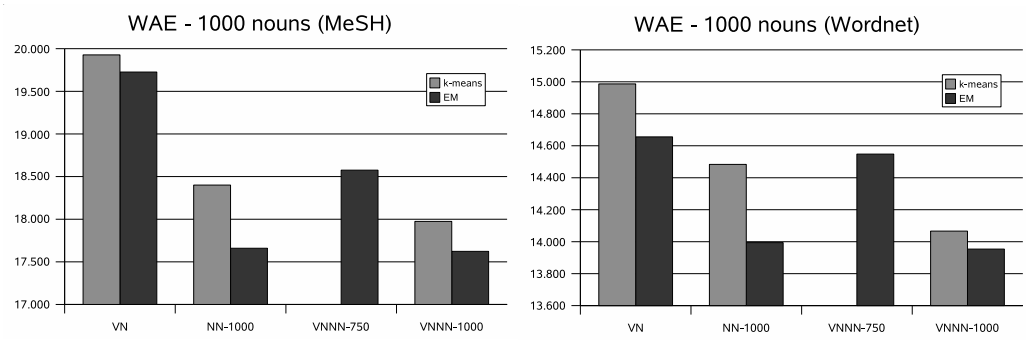


Figure 4.2: Weighted average entropy of the noun clusters for 1000 nouns.

2. only the complete noun-noun data and the N-N model (denoted as NN-1000),

3. the noun-noun data for 750 nouns and augmented with side information about all 1000 nouns in the form of the verb-noun table and using the full model (denoted as VNNN-750),

4. the complete noun-noun data for all 1000 nouns augmented with the verb-noun data and the full model (denoted as VNNN-1000).

For comparison with k-means we used information as similar as possible. For case 1 above, we used the verb-noun pairs, treating the verbs as features. For case 2, we used the noun-noun pairs and clustered the nouns using nouns as features. For case 4, the verb-noun and noun-noun tables were combined by concatenating them and k-means found noun clusters using both verb and noun co-occurrences as features for the nouns. We did not compare k-means with VNNN-750 of case 3, because combining the two data tables as before would give vectors with missing features and using k-means with missing features would require special treatment.

The results of these experiments are shown in figure 4.2. The results were similar when mapping the nouns in either MeSH or Wordnet. In every case, EM on the BN model gave better results than k-means. We now examine two scenarios. In the first one, the noun-noun pairs provide information for all 1000 nouns. In this case,

the VN results (using only verb-noun data) perform worst, while the NN and the combined VNNN-1000 results were better in terms of WAE.

For the second scenario, we assume we only have noun-noun information for 750 nouns. Without any other information, the only way to cluster more nouns (in this experiment the extra 250 ones) is to assign them randomly to clusters. The WAE in this case is much worse (statistically significant). The alternative is to use the verb-noun data. By combining the two tables we get the VNNN-750 results which are better than the VN ones, even though they do not give information for the 250 nouns. We can thus achieve coverage of 1,000 nouns with our clusters with better quality than if we had simply used the verb-noun data, which in this scenario are the only available data covering 1,000 nouns.

According to figure 4.2, NN-1000 is better than VN. The reason lies in the nature of the noun-noun data (pairs of nouns extracted from conjunctions and appositions), which makes them of high quality, because two nouns that appear together in a conjunction, disjunction or apposition are usually very similar. These pairs therefore give very good clusters. The verb-noun data on the other hand contain less information on nouns, because in verb-direct object pairs the same verb can be observed with a wider variety of nouns as direct objects.

Looking at the actual clusters it is obvious that the VN clusters are of lower quality than the other results. This is not the case for the NN-1000 and the VNNN-1000 clusters. In table 4.2 we give representative examples of clusters from these methods. The clusters were manually selected to correspond to the same high-level notions. Both the NN-1000 and VNNN-1000 clusters in that table look of good quality, better than VN-1000, which contains several spurious words. Comparing the NN-1000 and VNNN-1000 clusters in the same table, we note that they capture different types of concepts. The NN-1000 cluster contains words related to *knowledge* and *behavior/performance*, whereas the VNNN-1000 cluster is a combination of *knowledge* and *institutions*. Both of them can be considered correct in some setting;

VN	control, use, therapy, development, procedure, diagnosis, process, that, surgery, strategy, measure, essential, management, care, ability, education, detection, knowledge, practice, safety, chemotherapy, course, identification, prevention, support, service, modification, step, regimen, screening, operation, repair, resection, selection, transplantation, task, now, graft, technology, environment, transfer, medication, recognition, means, radiotherapy, maintenance, limitation, energy, availability, concept, understanding, trend, modality, introduction, implant, protection, resource, ventilation, flap, dissection
NN-1000	system, research, strategy, information, program, application, care, work, behavior, education, experience, health, knowledge, practice, hospital, performance, need, issue, support, preparation, service, unit, training, technology, standard, environment, effort, way, laboratory, attitude, guideline, setting, network, availability, behaviour, theory, center, understanding, skill, perception, policy, concern, decision, access, medicine, community, database, formulation, clinic, resource, planning, life, country, communication, project, barrier, goal
VNNN-1000	system, author, trial, research, strategy, information, program, application, care, work, education, source, experience, health, knowledge, practice, hospital, need, issue, survey, support, service, unit, training, literature, technology, standard, environment, effort, question, laboratory, attitude, challenge, guideline, network, availability, concept, theory, center, advance, understanding, skill, perception, policy, concern, decision, access, medicine, attempt, community, contact, interest, search, database, record, recommendation, basis, situation, clinic, resource, methodology, life, country, communication, project, barrier, discussion, consideration
VN	mice, type, tumor, tissue, line, material, strain, mutant, liver, carcinoma, culture, virus, human, specie, muscle, some, heart, clone, variant, bacteria, nucleus, derivative, tumour, cortex, spleen, fibroblast, fiber, organ, vessel, nuclei, mouse, artery, neutrophil, neck, platelet, nerve, embryo, particle, monocyte, chromosome, lymph node, organism, adenocarcinoma, adenoma, axon, intestine, astrocyte, leukocyte, CD4
NN-1000	cell, line, neuron, source, macrophage, majority, lymphocyte, clone, fibroblast, neutrophil, platelet, epithelial cell, monocyte, precursor, subset, astrocyte, leukocyte, CD4
NN-1000	model, rat, mice, animal, normal, human, dog, mouse, plant, embryo, rabbit, cat, pig
VNNN-1000	cell, rat, mice, line, that, strain, animal, mutant, neuron, normal, human, specie, macrophage, majority, lymphocyte, clone, bacteria, vector, dog, fibroblast, mouse, neutrophil, platelet, plant, epithelial cell, embryo, monocyte, precursor, organism, rabbit, cat, wild-type, subset, pathogen, pig, astrocyte, leukocyte, CD4

Table 4.2: Examples of corresponding noun clusters found with three of the methods.

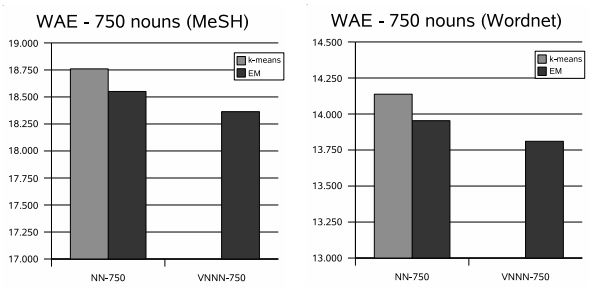


Figure 4.3: Evaluation of the noun clusters for the 750 randomly selected nouns.

the best choice would be dependent on the application. Similarly, in the second set of clusters, the VNNN-1000 cluster is about *biological substrates*, whereas NN-1000 found two separate clusters, one for *animals* and one for *micro-organisms*.

The VN-1000 clusters are broader, whereas the NN-1000 and VNNN-1000 ones correspond better to MeSH. The way the noun-noun pairs were extracted from the text produces highly-correlated co-occurrences that in turn yield highly specific clusters. On the other hand, the nouns that appear in verb-noun pairs are considered similar if similar actions are done to them (i.e. similar verbs apply to similar direct objects). The use of the verb-noun pairs in the full model introduces new relations between words that are not present in the noun-noun data. This causes the found clusters to be broader.

4.4.3 Clustering 750 nouns

For the second set of experiments we tested how well the 750 randomly chosen nouns were clustered. We therefore ignored the 250 missing nouns when computing the WAE. Specifically, we compared the clusters found by:

1. VNNN-750 (defined in the previous subsection),

2. the N-N model, operating on the reduced noun-noun data (NN-750).

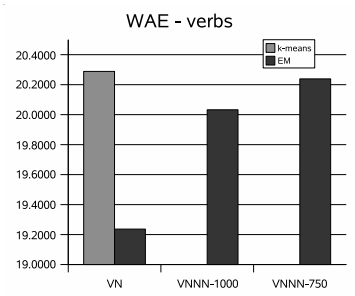


Figure 4.4: Evaluation of the verb clusters.

The results show that for both mappings, k-means is again worse than EM (figure 4.3). We see more pronounced what we noticed in the previous experiment, i.e. that the extra information helped improve the cluster quality. The full model for this case performs better than the N-N model in terms of WAE. The verb-noun data containing information for all 1000 nouns help improve the clusters for the 750 nouns, making a greater contribution than in the previous set of experiments.

4.4.4 Clustering verbs

Most ontologies cover nouns, but verbs are also of potential interest. For example, when extracting relationships between e.g. proteins, it is useful to have term-sets of verbs of protein interaction such as *bind, ubiquinate*, or *phosphorylate*. For our last set of experiments we looked at finding verb clusters. Since there is no straightforward way to find verb clusters with k-means using both the verb-noun and noun-noun information, we did not compute verb clusters for the VNNN cases. We found the best results when using the standard V-N model and not the full model (figure 4.4). It seems that the introduction of the noun-noun co-occurrence information pulls the verb clusters apart in order to achieve better noun clusters. However, even though the extra information hurts the precision of the verb clusters, it introduces new relations between verbs. Similar to the case of noun clusters, the outcome is a

different clustering result, which, depending on the application, may be preferable. To support this point, we give examples of verb clusters in table 4.3.

4.5 Discussion

Our algorithm can be used in any application that requires clustering, such as information or relation extraction. For example, it can be used to extend MeSH [92], or to create an alternate ontology to MeSH, for instance one where the relations are of different type. We have not examined issues like the best number of clusters, or what and how many co-occurrence types to use to generate the data. The choice of the co-occurrence types will depend on the application domain and the kind of clusters we want to find, whereas the number of clusters can be found through other methods (see for example [89, 70])

Our focus, instead, was to show that using EM on BNs with tied parameters is a useful method for incorporating information from several different sources to achieve greater coverage and cluster quality. One advantage we found was the potential to increase the coverage of clustered words. By using different syntactic patterns one can easily gather information about extra words from a relatively small corpus of text and combine it with the proposed method, acquiring clusters of a large number of words. So, nouns which do not show up in any noun-noun collocations can still be clustered if verb-noun pairs are available. The alternative would be to use a single syntactic pattern on a larger corpus, which may not always be available.

The second "advantage" is more subtle, and may sometimes be a disadvantage; Supplementing noun-noun co-occurrence data with verb-noun data changes the nature of the clusters that are found. Nouns that are the targets of the same action (and thus appear with the same verb) often constitute a different, and broader, set than nouns that are mentioned together only in noun-noun collocations. For example, different types of tissues and of animal were clustered together when the verb-noun

VN	increase, decrease, differ, change, correlate, vary, range, reach, tend, decline, exceed, rise, return to, average, approach, fall, peak, adjust, double, parallel, drop, fell, compensate for, index, amount to, approximate, fraction, equal, decay, fall to, fluctuate, profile, stratify, leak, plot
VNNN-750	increase, decrease, differ, change, correlate, vary, range, reach, tend, decline, exceed, rise, return to, total, average, approach, fall, peak, adjust, double, divide, drop, free, near, fell, bypass, curve, index, weight, amount to, pressure, approximate, fraction, equal, balance, fall to, fluctuate, overload, profile, stratify, leak, deviate, beat, lag, wound, plot
VNNN-1000	increase, decrease, differ, change, correlate, vary, range, reach, decline, exceed, rise, return to, gain, total, average, approach, fall, peak, adjust, rest, value, double, drop, free, moderate, fell, weight, amount to, pressure, approximate, water, fraction, equal, balance, fall to, fluctuate, spike, profile, leak, power, beat, lag
VN	express, exhibit, display, lack, carry, grow, differentiate, accumulate, bear, surface, stain, derive from, retain, originate, surround, count, secrete, acquire, synthesize, infect, proliferate, migrate, overexpress, harbor, spread, divide, behave, cluster, kill, project, cycle, supply, penetrate, infiltrate, line, invade, adhere to, fire, regenerate, size, escape, attach to, harbour, pulse, swell, mature, cut, mount, resist, pass through, attach, immobilize, enlarge, layer, metastasize, take up, preexist, degenerate, roll, sense, colonize, adhere, develop into, branch, internalize
VNNN-750	express, differentiate, surface, stain, surround, count, secrete, proliferate, overexpress, infiltrate, line, fire, ionize, swell, mature, layer, take up, roll, sense, aggregate, exit
VNNN-1000	express, grow, differentiate, bear, surface, feed, stain, count, secrete, transform, synthesize, proliferate, overexpress, spread, divide, kill, project, cycle, infiltrate, line, invade, adhere to, fire, incubate, regenerate, skin, ionize, swell, mature, mount, resist, subject to, layer, swim, take up, roll, sense, colonize, aggregate, adhere

Table 4.3: Examples of corresponding verb clusters from the three models.

information was used, since they are all similar experimental substrates. Finding clusters of different type can be helpful for building ontologies that organize information in alternative ways, for example for use in extracting relationships between entities, where broader classes would be useful.

Chapter 5

Discussion

We have presented Streemer [63, 64], a few-pass online algorithm that clusters observations into foreground clusters and a background. Unlike k-means, Streemer does not require a predetermined number of clusters and does not make strong assumptions about cluster sizes and variances. Streemer's performance was evaluated on a dataset of scientific papers citing other papers, where the conference or journal in which the paper appeared was treated as a class label. It was compared to k-means as well as a variation of k-means that we developed and which distinguishes foreground from background. We found that by filtering some points as background we can improve the quality of the foreground clusters.

We used Streemer on a dataset of scientific papers to discover scientific knowledge communities and study their properties and their evolution over time. For this we used a "rolling clustering" scheme, in which we divided and clustered the papers into overlapping time periods, and used the overlap to track the continuity of the clusters. Based on these findings, we built models predicting the growth of scientific communities and the citation impact of papers using features such as the citation structure, the vocabulary of the papers, and the affiliations and prestige of the authors. Analysis of the statistical significance of the model features gave insight into the characteristics of successful knowledge communities and high impact papers.

We have also given a mixture model that is based on log likelihood and uses a Dirichlet process prior and described how it can be used in an streaming clustering setting. The Dirichlet process provides the mechanism for generating new clusters as the observations are processed sequentially and the Gaussian components constitute the resulting clusters. Because a single pass makes many wrong decisions, especially in the beginning, we have proposed the use of two passes for clustering the data. In the first pass we find a much higher number of clusters than desired, so as to avoid undesirable local minima, and in the second pass we treat the first-pass clusters as weighted points to be clustered. This exposition corresponds in large degree to the way Streemer operates and therefore Streemer can be viewed as a DPMM with additional heuristics for avoiding local minima and an extra step for finding a background.

On multiway clustering, we have developed a probabilistic model for simultaneous clustering on multiple dimensions and used it to simultaneously cluster verbs and nouns using tables of verb-noun and noun-noun co-occurrence data [62]. The advantage of simultaneous clustering is that the clustering of each variable informs the clustering of the others. Additionally, the use of two different co-occurrence tables provides greater coverage of words than either set of co-occurrence pairs alone. It makes it thus possible to combine data from multiple sources and achieve clustering even if the amount of data from each source is very limited. We applied our model on data extracted from Medline abstracts and evaluated the results by mapping them to MeSH [92] and Wordnet [32].

5.1 Future work

The foreground/background aspect of Streemer has been used for the identification of knowledge communities. Documents that don't fall cleanly in an established

community, such as review papers, are assigned to the background making the foreground clusters "cleaner". This split between foreground groups that are unique and dissimilar from each other and a background group with observations that don't fit in any foreground cluster can be used for any dataset where we expect to have some data points that constitute noise. Furthermore, the foreground/background distinction can also be usefull when the *background* is the group of interest. For example, the points in the background could be considered as outliers and foreground/background clustering could potentially be used for anomaly detection. One must be careful however when applying the methodology, since the results may be different from the expected. We provided an illustrative example for finding stop words where this happened.

Chapter 4 addresses multiway clustering in a batch setting and with a predetermined number of clusters. A number of possible improvements could be made, when viewed in combination with the previous chapters. One improvement would be to introduce a DP to the model, so that the number of clusters does not have to be specified by the user. This would also allow the use of a streaming algorithm with the model instead of EM for the case of very large datasets. In that case, a two-pass clustering approach similar to chapter 3 is a possibility. Secondly, the existence of two clustered variables creates four possibilities for assignment to new or existing clusters. Further work would be to examine how to choose one of the four possibilities for each observation. Once the DP has been added to the model, one can then develop an online algorithm, preferably utilizing two passes for better performance. Such work, however, falls outside the scope of this thesis.

Bibliography

[1] P.D. Allison and R.P. Waterman. Fixed-Effects Negative Binomial Regression Models. *Sociological Methodology*, 32(1):247–265, 2002.

[2] A. Banerjee, I.S. Dhillon, J. Ghosh, and S. Sra. Clustering on the Unit Hypersphere using von Mises-Fisher Distributions. *Journal of Machine Learning Research*, 6(2):1345, 2006.

[3] A. Banerjee, S. Merugu, I.S. Dhillon, and J. Ghosh. Clustering with Bregman divergences. *The Journal of Machine Learning Research*, 6:1705–1749, 2005.

[4] Arindam Banerjee, Inderjit Dhillon, Joydeep Ghosh, Srujana Merugu, and Dharmendra S. Modha. A generalized maximum entropy approach to Bregman co-clustering and matrix approximation. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 509–514, New York, NY, USA, 2004. ACM.

[5] R. Bekkerman, R. El-Yaniv, and A. McCallum. Multi-way distributional clustering via pairwise interactions. *Proceedings of ICML*, 22:41–48, 2005.

[6] Pavel Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, CA, 2002.

[7] David M. Blei and John D. Lafferty. Dynamic topic models. In *ICML '06: Proceedings of the 23rd international conference on Machine Learning*, pages 113–120, New York, NY, USA, 2006. ACM.

[8] D.M. Blei, T.L. Griffiths, M.I. Jordan, and J.B. Tenenbaum. Hierarchical Topic Models and the Nested Chinese Restaurant Process. In *Advances in Neural Information Processing Systems 16: Proceedings of the 2003 Conference*. Bradford Book, 2004.

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

[10] P. Bonacich. Using Boolean algebra to analyze overlapping memberships. *Sociological Methodology*, pages 101–115, 1978.

[11] S.P. Borgatti and M.G. Everett. Network analysis of 2-mode data. *Social Networks*, 19(3):243–269, 1997.

[12] Robert R. Braam, Henk F. Moed, and Anthony F. J. van Raan. Mapping of science by combined co-citation and word analysis. ii: Dynamical aspects. *Journal of the American Society for Information Science*, 42(4):252–266, 1991.

[13] M. Bundschus, M. Dejori, M. Stetter, V. Tresp, and H.P. Kriegel. Extraction of semantic biomedical relations from text using conditional random fields. *BMC Bioinformatics*, 9:207, 2008.

[14] Gilles Celeux and Gerard Govaert. A classification EM algorithm for clustering and two stochastic versions. *Comput. Stat. Data Anal.*, 14(3):315–332, 1992.

[15] D. Crane. *Invisible Colleges: Diffusion of Knowledge in Scientific Communities*. University of Chicago Press, Chicago, 1972.

[16] P. Damien and S. Walker. A full Bayesian analysis of circular data using the von Mises distribution. *The Canadian Journal of Statistics*, 27(2):291–298, 1999.

[17] S. Dasgupta and L. Schulman. A two-round variant of EM for Gaussian mixtures. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 143–151, 2000.

[18] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of Royal Statistical Society*, 39:1–38, 1977.

[19] Inderjit S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 269–274, New York, NY, USA, 2001. ACM Press.

[20] Inderjit S. Dhillon and Yuqiang Guan. Information theoretic clustering of sparse co-occurrence data. In *ICDM '03: Proceedings of the Third IEEE International Conference on Data Mining*, page 517, Washington, DC, USA, 2003. IEEE Computer Society.

[21] Inderjit S. Dhillon, Yuqiang Guan, and Brian Kulis. Kernel k-means: spectral clustering and normalized cuts. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 551–556, New York, NY, USA, 2004. ACM.

[22] Inderjit S. Dhillon, Subramanyam Mallela, and Dharmendra S. Modha. Information-theoretic co-clustering. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 89–98, New York, NY, USA, 2003. ACM Press.

[23] I.S. Dhillon, J. Fan, and Y. Guan. Efficient clustering of very large document collections. *Data Mining for Scientific and Engineering Applications*, pages 357–381, 2001.

[24] I.S. Dhillon, J. Fan, and Y. Guan. Efficient clustering of very large document collections. *Data Mining for Scientific and Engineering Applications*, pages 357–381, 2001.

[25] I.S. Dhillon and D.S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1):143–175, Jan 2001.

[26] P. Doreian. On the delineation of small group structure. *Classifying Social Data. Jossey-Bass, San Francisco, CA*, 1979.

[27] D. Downey, S. Schoenmackers, and O. Etzioni. Sparse Information Extraction: Unsupervised Language Models to the Rescue. *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 696–703, 2007.

[28] M. Ester, H.P. Kriegel, J. Sander, and X. Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining*, pages 226–231, Portland, OR, 1996. AAAI Press.

[29] M.G. Everett and S.P. Borgatti. An extension of regular colouring of graphs to digraphs, networks and hypergraphs. *Social Networks*, 15(23):7–254, 1993.

[30] J.W. Fan and C. Friedman. Semantic Classification of Biomedical Concepts Using Distributional Similarity. *Journal of the American Medical Informatics Association*, 14(4):467–477, 2007.

[31] P. Fearnhead. Particle filters for mixture models with an unknown number of components. *Statistics and Computing*, 14(1):11–21, 2004.

[32] C. Fellbaum. *Wordnet: An Electronic Lexical Database*. MIT Press, 1998.

[33] T.S. Ferguson. A Bayesian analysis of some nonparametric problems. *Ann. Statist*, 1(2):209–230, 1973.

[34] Xiaoli Zhang Fern and Carla E. Brodley. Random projection for high dimensional data clustering: A cluster ensemble approach. In *Proceedings of the Twentieth International Conference of Machine Learning*, pages 186–193, Washington, DC, USA, 2003. AAAI Press.

[35] Daniel Fink. A compendium of conjugate priors. Technical report, Cornell University, 1995.

[36] Gary William Flake, Steve Lawrence, and C. Lee Giles. Efficient identification of Web communities. In *KDD '00: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 150–160, New York, NY, USA, 2000. ACM.

[37] L.C. Freeman. Finding social groups: A meta-analysis of the southern women data. In *Dynamic Social Network Modeling and Analysis: Workshop Summary and Papers*, page 39. National Academy Press, 2003.

[38] K. Frenken, W. Hölzl, and F. Vor. The citation impact of research collaborations: the case of European biotechnology and applied microbiology (1988–2002). *Journal of Engineering and Technology Management*, 22(1-2):9–30, 2005.

[39] N. Friedman, O. Mosenzon, N. Slonim, and N. Tishby. Multivariate information bottleneck. *Uncertainty in Artificial Intelligence: Proceedings of the Seventeenth Conference (UAI-2001)*, pages 152–161, 2001.

[40] Lise Getoor, Nir Friedman, Daphne Koller, and Benjamin Taskar. Learning probabilistic models of relational structure. In *ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning*, pages 170–177, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[41] David Gibson, Jon Kleinberg, and Prabhakar Raghavan. *Inferring Web communities from link topology.* ACM, New York, NY, USA, 1998.

[42] C. Lee Giles, Kurt Bollacker, and Steve Lawrence. CiteSeer: An automatic citation indexing system. In Ian Witten, Rob Akscyn, and Frank M. Shipman III, editors, *Digital Libraries 98 - The Third ACM Conference on Digital Libraries*, pages 89–98, Pittsburgh, PA, June 23–26 1998. ACM Press.

[43] R. Gomes, M. Welling, and P. Perona. Incremental learning of nonparametric Bayesian mixture models. In *IEEE Conference on Computer Vision and Pattern Recognition, 2008. CVPR 2008*, pages 1–8, 2008.

[44] K.C. Gowda and G. Krishna. Agglomerative Clustering Using the Concept of Mutual Nearest Neighbourhood. *Nearest Neighbor (Nn) Norms: Nn Pattern Classification Techniques*, 1991.

[45] William H. Greene. Accounting for excess zeros and sample selection in Poisson and negative binomial regression models. Working Papers 94-10, New York University, Leonard N. Stern School of Business, Department of Economics, 1994.

[46] B.C. Griffith, H.G. Small, J.A. Stonehill, and S. Dey. The Structure of Scientific Literatures II: Toward a Macro-and Microstructure for Science. *Science Studies*, 4(4):339–365, 1974.

[47] T.L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences of the United States of America*, 101:5228–5235, 2004.

[48] S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering Data Streams: Theory and Practice. *IEEE Transactions on Knowledge and Data Engineering*, 15(3):515–528, 2003.

[49] S. Guha, R. Rastogi, and K. Shim. Cure: an efficient clustering algorithm for large databases. *Information Systems*, 26(1):35–58, 2001.

[50] G. Gupta and J. Ghosh. Bregman Bubble Clustering: A Robust, Scalable Framework for Locating Multiple, Dense Regions in Data. In *Proc. ICDM: The 2006 IEEE International Conference on Data Mining*, 2006.

[51] J. Hage and M.T.H. Meeus. *Innovation, Science, and Institutional Change*. Oxford University Press, USA, 2006.

[52] JT Hage. Organizational Innovation And Organizational Change. *Annual Reviews in Sociology*, 25(1):597–622, 1999.

[53] M.T. Hannan and J. Freeman. The Population Ecology of Organizations. *American Journal of Sociology*, 82(5):929, 1977.

[54] J. Hausman, B.H. Hall, and Z. Griliches. Econometric Models for Count Data with an Application to the Patents-R & D Relationship. *Econometrica*, 52(4):909–938, 1984.

[55] Yulan He and Siu Cheung Hui. Mining a web citation database for author co-citation analysis. *Inf. Process. Manage.*, 38(4):491–508, 2002.

[56] John Hopcroft, Omar Khan, Brian Kulis, and Bart Selman. Natural communities in large linked networks. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 541–546, New York, NY, USA, 2003. ACM Press.

[57] Q. Huang, B. Dom, D. Steele, J. Ashley, and W. Niblack. Foreground/background segmentation of color images by integration of multiple cues. *IEEE Int. Conf. on Image Processing*, 1:246–249, 1995.

[58] H. Ishwaran and M. Zarepour. Markov chain Monte Carlo in approximate Dirichlet and beta two-parameter process hierarchical models. *Biometrika*, 87(2):371–390, 2000.

[59] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Comput. Surv.*, 31(3):264–323, 1999.

[60] Frizo Janssens, Wolfgang Glänzel, and Bart De Moor. Dynamic hybrid clustering of bioinformatics by incorporating text mining and citation analysis. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 360–369, New York, NY, USA, 2007. ACM Press.

[61] I.T. Jolliffe. *Principal Component Analysis*. Springer, 2002.

[62] V. Kandylas, L. Ungar, T. Sandler, and S. Jensen. Multiway Clustering for Creating Biomedical Term Sets. In *Bioinformatics and Biomedicine, 2008. BIBM'08. IEEE International Conference on*, pages 449–452, 2008.

[63] V. Kandylas, S.P. Upham, and L.H. Ungar. Finding cohesive clusters for analyzing knowledge communities. *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 203–212, Oct. 2007.

[64] Vasileios Kandylas, S. Phineas Upham, and Lyle H. Ungar. Finding cohesive clusters for analyzing knowledge communities. *Knowledge and Information Systems*, 17(3):335–354, 2008.

[65] V. Kashyap, C. Ramakrishnan, and T.C. Rindflesch. Towards (Semi-)automatic Generation of Bio-medical ontologies. *AMIA. Annual Symposium proceedings [electronic resource]*, 2003:886, 2003.

[66] Michael J. Kearns, Yishay Mansour, and Andrew Y. Ng. An information-theoretic analysis of hard and soft assignment methods for clustering. *Proceedings of UAI*, pages 282–293, 1997.

[67] Yuval Kluger, Ronen Basri, Joseph T. Chang, and Mark Gerstein. Spectral Biclustering of Microarray Data: Coclustering Genes and Conditions. *Genome Res.*, 13(4):703–716, 2003.

[68] A. Koloydenko, M. Käärik, and J. Lember. On adjusted Viterbi training. *Acta Applicandae Mathematicae: An International Survey Journal on Applying Mathematics and Mathematical Applications*, 96(1):309–326, 2007.

[69] R.N. Kostoff, J.A. del Rio, J.A. Humenik, E.O. Garcia, and A.M. Ramirez. Citation mining: Integrating text mining and bibliometrics for research user profiling. *Journal of the American Society for Information Science and Technology*, 52(13):1148–1156, 2001.

[70] R. Kothari and D. Pitts. On finding the number of clusters. *Pattern Recognition Letters*, 20(4):405–416, 1999.

[71] M. Krauthammer and G. Nenadic. Term identification in the biomedical literature. *Journal of Biomedical Informatics*, 37(6):512–526, 2004.

[72] B. Larsen and C. Aone. Fast and effective text mining using linear-time document clustering. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 16–22, San Diego, CA, 1999. ACM Press.

[73] J.D. Lee, K.J. Vicente, A. Cassano, and A. Shearer. Can scientific impact be judged prospectively? A bibliometric test of Simonton's model of creative productivity. *Scientometrics*, 56(2):223–232, 2003.

[74] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *KDD '05: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 177–187, New York, NY, USA, 2005. ACM.

[75] Michael Ley. The DBLP computer science bibliography: Evolution, research issues, perspectives. In *SPIRE 2002: Proceedings of the 9th International Symposium on String Processing and Information Retrieval*, pages 1–10, London, UK, 2002. Springer-Verlag.

[76] H. Li. Word clustering and disambiguation based on co-occurrence data. *Natural Language Engineering*, 8(01):25–42, 2002.

[77] D. Lin and P. Pantel. Induction of semantic classes from natural language text. *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 317–322, 2001.

[78] Dekang Lin. Dependency-based evaluation of MINIPAR. In *Proceedings of the Workshop on the Evaluation of Parsing Systems, at LREC-98*. Springer, 1998.

[79] S.N. MacEachern and P. Mueller. Estimating Mixture of Dirichlet Process Models. *Journal Of Computational And Graphical Statistics*, 7:223–238, 1998.

[80] M.H. MacRoberts and B.R. MacRoberts. Problems of citation analysis. *Scientometrics*, 36(3):435–444, 1996.

[81] Sara C. Madeira and Arlindo L. Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 01(1):24–45, 2004.

[82] J.G. March. Exploration and Exploitation in Organizational Learning. *Organization Science*, 2(1):71–87, 1991.

[83] KV Mardia and SAM El-Atoum. Bayesian inference for the von Mises-Fisher distribution. *Biometrika*, 63(1):203–206, 1976.

[84] Naohiro Matsumura, Yukio Ohsawa, and Mitsuru Ishizuka. Discovery of emerging topics between communities on WWW. *Lecture Notes in Computer Science*, 2198:473–482, 2001.

[85] J.D. McAuliffe, D.M. Blei, and M.I. Jordan. Nonparametric empirical Bayes for the Dirichlet process mixture model. *Statistics and Computing*, 16(1):5–14, 2006.

[86] P. McCullagh and J.A. Nelder. *Generalized Linear Models*. Chapman & Hall/CRC, Boca Raton, FL, USA, 1989.

[87] AJ McGann. The Advantages of Ideological Cohesion a Model of Constituency Representation and Electoral Competition in Multi-Party Democracies. *Journal of Theoretical Politics*, 14(1):37–70, 2002.

[88] Amy McGovern, Lisa Friedland, Michael Hay, Brian Gallagher, Andrew Fast, Jennifer Neville, and David Jensen. Exploiting relational structure to understand publication patterns in high-energy physics. *SIGKDD Explor. Newsl.*, 5(2):165–172, 2003.

[89] G.W. Milligan and M.C. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2):159–179, 1985.

[90] H.F. Moed and I. NetLibrary. *Citation Analysis in Research Evaluation*. Springer, 2005.

[91] J. Moody. Peer influence groups: identifying dense clusters in large networks. *Social Networks*, 23(4):261–283, 2001.

[92] National Library of Medicine. Medical subject headings – annotated alphabetic list, 2002. Bethesda, MD, The Library, 2001.

[93] R.M. Neal. Markov chain sampling methods for Dirichlet process mixture models. *Journal of computational and graphical statistics*, pages 249–265, 2000.

[94] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14: Proceedings of the 2002 Conference*. MIT Press, 2002.

[95] Kamal Nigam, Andrew McCallum, Sebastian Thrun, and Tom M. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134, 2000.

[96] Patrick Pantel and Dekang Lin. Document clustering with committees. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 199–206, New York, NY, USA, 2002. ACM Press.

[97] F. Pereira, N. Tishby, and L. Lee. Distributional clustering of English words. *Proceedings of the 31st conference on Association for Computational Linguistics*, pages 183–190, 1993.

[98] Claudia Perlich, Foster J. Provost, and Sofus A. Macskassy. Predicting citation rates for physics papers: constructing features for an ordered probit model. *SIGKDD Explorations*, 5:154, 2003.

[99] J. Pfeffer. Barriers to the Advance of Organizational Science: Paradigm Development as a Dependent Variable. *The Academy of Management Review*, 18(4):599–620, 1993.

[100] J. Pitman and M. Yor. The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. *Annals Of Probability*, 25:855–900, 1997.

[101] A. Popescul, GW Flake, S. Lawrence, LH Ungar, and CL Giles. Clustering and identifying temporal trends in document databases. In *Advances in Digital Libraries, 2000. ADL 2000. Proceedings. IEEE*, pages 173–182, Washington, DC, USA, 2000. IEEE Computer Society.

[102] G. Punj and D.W. Stewart. Cluster Analysis in Marketing Research: Review and Suggestions for Application. *Journal of Marketing Research*, 20(2):134–148, 1983.

[103] C.E. Rasmussen. The infinite Gaussian mixture model. *Advances in Neural Information Processing Systems*, 12:554–560, 2000.

[104] S.T. Roweis and L.K. Saul. Nonlinear Dimensionality Reduction by Locally Linear Embedding, 2000.

[105] T. Sandler, A.I. Schein, and L.H. Ungar. Automatic term list generation for entity tagging. *Bioinformatics*, 22(6):651–657, 2006.

[106] M. Sato. Online model selection based on the variational Bayes. *Neural Computation*, 13(7):1649–1681, 2001.

[107] A.E. Savakis. Adaptive document image thresholding using foreground and background clustering. In *Proceedings of International Conference on Image Processing ICIP98*, Chicago, IL, 1998. IEEE Computer Society.

[108] H. Schütze and J.O. Pedersen. A cooccurrence-based thesaurus and two applications to information retrieval. *Information Processing and Management*, 33(3):307–318, 1997.

[109] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. In *CVPR '97: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, page 731, Washington, DC, USA, 1997. IEEE Computer Society.

[110] Shun-Hong Sie and Jian-Hua Yeh. Automatic ontology generation using schema information. In *WI '06: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 526–531, Washington, DC, USA, 2006. IEEE Computer Society.

[111] Noam Slonim, Nir Friedman, and Naftali Tishby. Agglomerative multivariate information bottleneck. In *Advances in Neural Information Processing Systems (NIPS)*, Vancouver, British Columbia, Canada, 2001. MIT Press.

[112] H. Small. Paradigms, citations, and maps of science: A personal history. *Journal of the American Society for Information Science and Technology*, 54(5):394–399, 2003.

[113] Henry Small and Ernest Sweeney. Clustering the science citation index using co-citations i: a comparison of methods. *Scientometrics*, 7:391–409, 1985.

[114] Henry Small and Ernest Sweeney. Clustering the science citation index using co-citations ii: Mapping science. *Scientometrics*, 8:321–340, 1985.

[115] HG Small and D. Crane. Specialties and disciplines in science and social science: An examination of their structure using citation indexes. *Scientometrics*, 1(5):445–461, 1979.

[116] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. *KDD Workshop on Text Mining*, 34:35, 2000.

[117] A. Strehl, J. Ghosh, and R. Mooney. Impact of similarity measures on web-page clustering. In *Proc. AAAI Workshop on AI for Web Search (AAAI 2000), Austin*, pages 58–64, 2000.

[118] Alexander Strehl and Joydeep Ghosh. Cluster ensembles – a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3:583–617, 2002.

[119] Daniel Sullivan, D. Hywel White, and Edward J. Barboni. Co-citation analyses of science: An evaluation. *Social Studies of Science*, 7(2):223–240, 1977.

[120] Jimeng Sun, Christos Faloutsos, Spiros Papadimitriou, and Philip S. Yu. GraphScope: parameter-free mining of large time-evolving graphs. In *KDD*

'07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 687–696, New York, NY, USA, 2007. ACM Press.

[121] C. Tantipathananandh, T. Berger-Wolf, and D. Kempe. A framework for community identification in dynamic social networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 717–726, San Jose, CA, 2007. ACM Press New York, NY, USA.

[122] S. Phineas Upham. *Communities of innovation*. PhD thesis, University of Pennsylvania, 2006.

[123] P. Wang. A bivariate zero-inflated negative binomial regression model for count data with excess zeros. *Economics Letters*, 78(3):373–378, 2003.

[124] X. Wang and A. McCallum. Topics over time: a non-Markov continuous-time model of topical trends. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 424–433, Philadelphia, PA, 2006. ACM Press New York, NY, USA.

[125] J. Weeds and D. Weir. Co-occurrence Retrieval: A Flexible Framework for Lexical Distributional Similarity. *Computational Linguistics*, 31(4):439–475, 2005.

[126] Julie Weeds, James Dowdall, Gerold Schneider, Bill Keller, and David J. Weir. Using distributional similarity to organise biomedical terminology. *Terminology*, 11:107–141(35), 2005.

[127] J. Zhang, Z. Ghahramani, and Y. Yang. A probabilistic model for online document clustering with application to novelty detection. *Advances in Neural Information Processing Systems*, 17:1617–1624, 2005.

[128] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. In *Proceedings of the 1996 ACM SIGMOD international conference on Management of data*, pages 103–114, Montreal, Quebec, Canada, 1996. ACM Press New York, NY, USA.

[129] Shi Zhong. Efficient streaming text clustering. *Neural Netw.*, 18(5-6):790–798, 2005.