Improving Clustering Stability with Combinatorial MRFs

Bekkerman, Ron; Scholz, Martin; Viswanathan, Krishnamurthy

HP Laboratories
HPL-2009-46

Keyword(s):
Clustering stability, combinatorial MRF, Comraf

Abstract:

As clustering methods are often sensitive to parameter tuning, obtaining stability in clustering results is an important task. In this work, we aim at improving clustering stability by attempting to diminish the influence of algorithmic inconsistencies and enhance the signal that comes from the data. We propose a mechanism that takes m clusterings as input and outputs m clusterings of comparable quality, which are in higher agreement with each other. We call our method the Clustering Agreement Process (CAP). To preserve the clustering quality, CAP uses the same optimization procedure as used in clustering. In particular, we study the stability problem of randomized clustering methods (which usually produce different results at each run). We focus on methods that are based on inference in a combinatorial Markov Random Field (or Comraf, for short) of a simple topology. We instantiate CAP as inference within a more complex, bipartite Comraf. We test the resulting system on four datasets, three of which are medium-sized text collections, while the fourth is a large-scale user/movie dataset. First, in all the four cases, our system significantly improves the clustering stability measured in terms of the macro-averaged Jaccard index. Second, in all the four cases our system managed to significantly improve clustering quality as well, achieving the state-of-the-art results. Third, our system significantly improves stability of consensus clustering built on top of the randomized clustering solutions.

External Posting Date: August 24, 2009 [Fulltext] Approved for External Publication
Internal Posting Date: February 21, 2009 [Fulltext]

Published in KDD Conference, Paris, France, June 29, 2009.

Improving Clustering Stability with Combinatorial MRFs

Ron Bekkerman
HP Labs
1501 Page Mill Rd
Palo Alto, CA 94304
ron.bekkerman@hp.com

Martin Scholz
HP Labs
1501 Page Mill Rd
Palo Alto, CA 94304
scholz@hp.com

Krishnamurthy Viswanathan
HP Labs
1501 Page Mill Rd
Palo Alto, CA 94304
krishnamurthy.viswanathan@hp.com

ABSTRACT

As clustering methods are often sensitive to parameter tuning, obtaining stability in clustering results is an important task. In this work, we aim at improving clustering stability by attempting to diminish the influence of algorithmic inconsistencies and enhance the signal that comes from the data. We propose a mechanism that takes m clusterings as input and outputs m clusterings of comparable quality, which are in higher agreement with each other. We call our method the *Clustering Agreement Process (CAP)*. To preserve the clustering quality, CAP uses the same optimization procedure as used in clustering. In particular, we study the stability problem of randomized clustering methods (which usually produce different results at each run). We focus on methods that are based on inference in a *combinatorial Markov Random Field* (or *Comraf*, for short) of a simple topology. We instantiate CAP as inference within a more complex, bipartite Comraf. We test the resulting system on four datasets, three of which are medium-sized text collections, while the fourth is a large-scale user/movie dataset. First, in all the four cases, our system significantly improves the clustering stability measured in terms of the macro-averaged Jaccard index. Second, in all the four cases our system managed to significantly improve clustering quality as well, achieving the state-of-the-art results. Third, our system significantly improves stability of consensus clustering built on top of the randomized clustering solutions.

Categories and Subject Descriptors

I.5 [**Pattern Recognition**]: Clustering

General Terms

Algorithms

Keywords

Clustering stability, combinatorial MRF, Comraf

1. INTRODUCTION

Data mining practitioners are often challenged with vaguely stated requests such as: "We have a large data collection—analyze it for us". Many data analysis approaches have been developed, among which data clustering—partitioning the data collection to semantically coherent groups—is one of the most prominent approaches. A large variety of clustering methods are available, each of which has its own advantages and drawbacks. The major question that arises after a clustering method has been applied to a data collection is why the data was clustered this way. A data instance x_1 was placed together with a data instance x_2—does this mean that they are inherently similar? Or they are not so similar but all the other options appear worse? Or our clustering method just made a mistake by associating x_1 with x_2?

This question becomes even more acute when taking into account the fact that different clustering methods usually obtain different results on the same data. Even the same method can obtain different results depending on a particular setting of its parameters. Those clustering methods that base on non-convex optimization often employ randomization to escape local optima, which implies that, with high probability, their results are different from run to run. Having dozens or hundreds of different clusterings of the same data does not add much clarity to our data analysis task.

Nevertheless, having more than one clustering can be helpful. Some data instances tend to come together with some others; some instances tend to float around. If all the available clustering methods assign x_1 to the same cluster with x_2, then this can be a good indicator that x_1 and x_2 have something profound in common. Unfortunately, such signals are usually weak: it is a relatively rare case when one data instance *always* goes with another. Some clusterings may capture a few patterns like that, whereas others would capture different patterns. An agreement amongst all those clusterings would be much desired.

A somewhat standard approach to achieving an agreement of clusterings is called *Cluster Ensembles* [22] or, alternatively, *Consensus Clustering* (see, e.g., [13]), which embodies a wide variety of averaging schema for the original set of clusterings. The result of a consensus clustering procedure is a single clustering that is an optimal combination (in one sense or another) of the original clusterings. We argue that the consensus clustering approach provides only partial solution to the agreement problem: although it does achieve an agreement about a particular clustering, it is still not clear how *stable* this agreement is with respect to tuning the consensus method's parameters. Also, will the same consensus method lead to a similar consensus when applied to different clusterings of the same data? Will a different consensus method lead to a substantially different consensus? Moreover, since a consensus method aims at obtaining a *single* clustering, it cannot provide answers to questions such as: which data instances compose a *stable component* (i.e. they are most often—or always—clustered together)? Which stable components tend to be clustered with which? Which data instances do not have a strong pattern of association with the others?

In this work, we introduce the *Clustering Agreement Process (CAP)* that provides means for improving stability of a clustering ensemble, without shrinking the space down to a single clustering solution, and without compromising the clustering ensemble's quality. In a nutshell, CAP can be described as follows. It operates in the space of all possible partitions of a given data collection. Initialized with the original clusterings, it employs local search that makes the clusterings "get closer" (i.e. become more similar) to each other. Definitely enough, an arbitrarily chosen local search technique may heavily hurt the quality of the original clusterings. As a matter of fact, for any pair of clusterings x_1^c and x_2^c, there exists a sequence of local updates (i.e. moves of one data instance from one cluster to another), which transfers x_1^c into x_2^c. We overcome this problem by applying the type of local search similar to the one used in clustering for constructing semantically meaningful solutions.

Our underlying clustering method is essentially *bi-modal*—it allows to simultaneously construct two clusterings: one of data instances, another one of their features. We use the term *data modality* to address a set of data instances, as well as the set of their features.[1] Bi-modal (and, generally, multi-modal) clustering methods are commonly believed to exceed the ordinary, uni-modal clustering methods in the quality of clustering multidimensional data (for a discussion, see [2]). A *Combinatorial Markov Random Field* (or *Comraf*, see Section 3) has proven itself to be a powerful model for multi-modal clustering [4, 3]. In original Comraf models, clusterings of data modalities are organized in a Markov Random Field (MRF) with one node per modality, and edges representing interactions between the clusterings. A multi-modal clustering task is subsequently expressed in terms of the Most Probable Explanation (MPE) estimation in the constructed MRF. At a node level, the inference procedure reduces to a local search routine that starts with some initial clustering and stops at a clustering for which the MPE objective reaches its local maximum, representing an agreement between this clustering and its neighbors.

In previously proposed Comraf models, it was not useful to have more than one node per modality. In this paper, however, we show that the clustering agreement process introduced above can be easily embedded into the Comraf framework, with as many nodes as the number of the original clusterings. We assume that m bi-modal clustering algorithms were initially applied to a given data collection, which led to constructing m clusterings of modality $\mathcal{X}$ and m clusterings of modality $\mathcal{Y}$. We build a Comraf model with m nodes per one modality and m nodes per another. Each node that corresponds to $\mathcal{X}$ is connected to each node that corresponds to $\mathcal{Y}$ and vice versa, resulting in an MRF of bipartite topology. An MPE inference is applied to the constructed MRF, which at each node boils down to a local search procedure similar to the one previously proposed. It is initialized with one of the pre-constructed clusterings and aims at finding a clustering that maximizes the agreement with its neighbors in the MRF. Given the bipartite topology, all m new clusterings of $\mathcal{X}$ maximize agreement with all m clusterings of $\mathcal{Y}$, which results in improving an agreement between themselves (all details are provided in Section 4).

Note that the algorithms that originally clustered the data do not play any role in the agreement process proposed above, such that no pre-conditions are imposed on them, apart from being bi-modal. However, the problem of choosing a clustering ensemble is not in the focus of our current study, and therefore we decided to construct original clusterings using the same bi-modal Comraf method that underlies the agreement process. This clustering method employs randomization such that, with high probability, it produces different results from run to run, and thus is suitable for our task. To a large extent, the proposed agreement process allows the original clustering method to escape the local maximum it is stuck in, and continue the clustering optimization process while biasing towards other solutions. This observation brings us to a hypothesis that the proposed process may significantly improve not only the clustering stability, but also the clustering quality (we confirm this in Section 6).

Potentially, such an agreement process can lead to a single solution, i.e. all the local searches that started from the original clusterings will meet together. Practically, however, this would be quite impossible for the following reasons: (a) the search space is prohibitively large such that finding a common solution of high quality is computationally hard; (b) our non-convex optimization procedure is unable to break ties: if a data instance x_1 equally probably belongs to two clusters, it can be found in either of them. Building m similar, but not equal clustering solutions opens the door to extensive data analysis. Being similar, the clusterings define relatively large stable components, each of which has a pattern of association with the others. Such a pattern can be easily converted into a distribution over the clusters, which is likely to be peaky enough and therefore easy for comprehension and interpretation. A smoother distribution will then determine a floater. To a certain extent, the proposed mechanism can be used for *softening* a set of hard clustering solutions (i.e. partitions), with a variety of practical applications in which soft clusters are needed.

The proposed agreement process is likely to identify a dense region in the space of possible clusterings. If the user is interested in a single clustering instead of an ensemble, a consensus clustering mechanism can be applied to the constructed clusterings. Such a consensus method will most probably explore the identified dense region, which would lead to a potentially more stable and therefore more reliable solution compared to the consensus of the original clusterings. We justify this empirically in Section 6.1.

[1]From here on, we will intentionally refrain from using the term "feature", which is somewhat misleading in our context. For example, in a document collection, documents are commonly considered "data instances", while their words are considered "features". However, if the task is to build a meaningful *word clustering*, then words become "data instances", while the identities of documents in which the words appear become "features". Accepting the Bag-Of-Words assumption, a document collection can be represented as a contingency table with documents as rows and words as columns. In a contingency table, rows and columns are equivalent up to transposition.

2. RELATED WORK

An extensive body of work has been published on finding a consensus within a clustering ensemble. As discussed above, we are solving an essentially different problem of improving clustering stability, however consensus clustering is a related task that needs to be reviewed. Cluster ensembles were proposed and extensively studied by Strehl and Ghosh [22], who used an information-theoretic framework for discovering a consensus of clusterings. Our MPE inference in the Comraf model also has information-theoretic nature, with its roots in information-theoretic co-clustering [10] and multivariate information bottleneck [21]. Note that Strehl and Ghosh's method is uni-modal while we apply a multi-modal technique. A consensus of bi-modal clustering methods was first investigated by Badea [1]. Fern and Brodley [11] proposed to solve the cluster ensemble problem using bipartite graph partitioning. Our Comraf model also has a bipartite topology, however our method is not graph-theoretic. Recently, Punera and Ghosh [18] proposed a hard clustering consensus of soft clusterings. To some extent, we address the opposite problem of softening an ensemble of hard clusterings.

In the previously published literature, we were unable to find any work on *improving* clustering stability, however many works deal with assessing and measuring stability of clustering methods. First works in the area were published in the early 1980's [19] which were based on an even earlier work by Rand [20] who developed criteria for clustering evaluation. In the last decade, an effort was done on assessing stability of deterministic clustering algorithms using either Rand's criteria [7], resampling [16], or classification [15], for the tasks of cluster validation and model selection.[2] Later, Ben-David et al. [6] formalized the problem and proposed theoretic criteria for stability of clustering methods, under a few assumptions. Presumably the most relevant work to our current study was done by Kuncheva and Vetrov [14] who evaluated stability of cluster ensembles under random initialization. They proposed to measure an ensemble's stability with a pairwise *adjusted Rand index*, averaged over all the clustering pairs. We adopt their averaging idea, but instead of using the adjusted Rand index we use the *Jaccard index* [12] which is easier for interpretation (see Section 5.2).

3. PRELIMINARIES

In this section we will provide all the necessary definitions and notation. As discussed in Section 1, we are given m clusterings of a data modality $\mathcal{X}$ and m clusterings of a data modality $\mathcal{Y}$, and our goal is to search the solution space around those clusterings in order to come up with new clusterings that maximize an agreement objective function. To formalize the space of all possible clusterings in which the local search is performed, we define a discrete distribution P over the entire space, such as each possible clustering is assigned a certain probability mass. We define a *combinatorial* random variable that is distributed according to P:

Definition 3.1. *A* combinatorial random variable *(or* combinatorial r.v.*) X^c is a discrete random variable defined over a combinatorial set $\mathcal{X}^c$, which is a finite set whose size is exponential with respect to another finite set $\mathcal{X}$, i.e.* $\log|\mathcal{X}^c| = O(|\mathcal{X}|)$.

[2]One major model selection problem extensively covered in the literature is the choice of the number of clusters k. In this work, however, we assume that k is given to us.

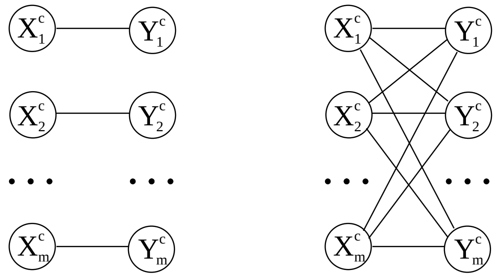

Figure 1: (left) Comraf model used for the original clustering; (right) bipartite Comraf for the clustering agreement process.

For our particular case, the combinatorial set $\mathcal{X}^c$ is a set of all possible clusterings of the data modality $\mathcal{X}$, and the event of picking one clustering x_1^c out of the set $\mathcal{X}^c$ has probability $P(X^c = x_1^c)$. From the theoretical point of view, $\mathcal{X}^c$ is just a discrete random variable with a finite support. In practice, however, the set $\mathcal{X}^c$ is extremely large, such that the distribution $P(X^c)$ cannot be explicitly specified. Nevertheless, given two values x_1^c and x_2^c, we can determine which one is more probable than the other. An interaction pattern of a combinatorial random variable with other random variables can be modeled through a combinatorial MRF:

Definition 3.2. *A* combinatorial Markov Random Field (Comraf) *is a Markov Random Field, at least one node of which is a combinatorial random variable.*

From here on, we will discuss only Comraf models *each* node of which is a combinatorial r.v. Generalizing P into a joint distribution over all combinatorial r.v.'s in the Comraf, let us write P down in the form of the Gibbs distribution [8]:

$$P(\mathbf{x}^c) = \frac{1}{Z_{\mathbf{f}}} \exp\left(\sum_{\mathcal{C}} f_{\mathcal{C}}(\mathbf{x}^c_{\mathcal{C}})\right),$$

where $\mathbf{x}^c$ is a vector of all combinatorial r.v.'s in the Comraf; $\mathcal{C}$ is a clique in the Comraf graph; $\mathbf{x}^c_{\mathcal{C}}$ are combinatorial r.v.'s participating in the clique $\mathcal{C}$; $f_{\mathcal{C}}$ is a real-valued function called a *log-potential*; and $Z_{\mathbf{f}}$ is a normalization factor called a *partition function*. If we fix the log-potentials $f_{\mathcal{C}}$ for each clique, the partition function $Z_{\mathbf{f}}$ becomes a constant. Thus, the Most Probable Explanation (MPE) inference in the Comraf is defined as

$$\begin{aligned}\mathbf{x}^c_* &= \arg\max_{\mathbf{x}^c} P(\mathbf{x}^c) = \arg\max_{\mathbf{x}^c} \exp\left(\sum_{\mathcal{C}} f_{\mathcal{C}}(\mathbf{x}^c_{\mathcal{C}})\right) \\ &= \arg\max_{\mathbf{x}^c} \sum_{\mathcal{C}} f_{\mathcal{C}}(\mathbf{x}^c_{\mathcal{C}}), \qquad (1)\end{aligned}$$

which now solely depends on the choice of log-potentials. Both for simplicity and for feasibility of our inference algorithms, let us consider only cliques of size 2, i.e. graph edges. Denoting by $\mathcal{E}$ the set of Comraf's edges, Equation (1) can then be rewritten as:

$$\mathbf{x}^c_* = \arg\max_{\mathbf{x}^c} \sum_{e_{ij}\in\mathcal{E}} f_{ij}(x_i^c, x_j^c). \qquad (2)$$

4. CLUSTERING AGREEMENT PROCESS

Let us first describe how the underlying clustering method works. For the task of bi-modal clustering, a very simple Comraf model can be built which contains two nodes

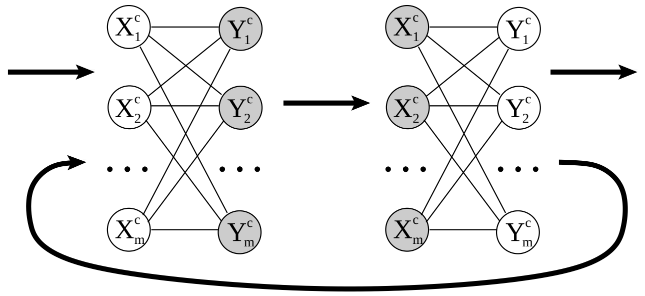

Figure 2: The clustering agreement process. First, we fix the values of all $Y_j^c|_{j=1}^m$ variables (shaded on the picture) and optimize all $X_i^c|_{i=1}^m$ variables in parallel. Then we fix new values of all X_i^c variables (shaded) and optimize all Y_j^c variables in parallel.

connected with an edge. The two nodes are combinatorial r.v.'s X^c and Y^c defined over all possible clusterings of data modalities $\mathcal{X}$ and $\mathcal{Y}$, respectively. Over the only edge, we can define a log-potential function that would characterize an agreement between clusterings x^c and y^c. Assuming that the number of clusters k is given to us and never changes, we follow Dhillon et al. [10] by choosing *Mutual Information* between the clusterings to be our log-potential function (for a discussion, see [4]). We define a random variable $\tilde{X}$ over all clusters in the clustering x^c. We define $\tilde{Y}$ analogously. The only log-potential in the model is then:

$$f(x^c, y^c) = I(\tilde{X}; \tilde{Y}) = \sum_{ij} p(\tilde{x}_i, \tilde{y}_j) \log \frac{p(\tilde{x}_i, \tilde{y}_j)}{p(\tilde{x}_i)p(\tilde{y}_j)}, \qquad (3)$$

where the joint distribution $p(\tilde{x}_i, \tilde{y}_j)$ is estimated from the data as $p(\tilde{x}_i, \tilde{y}_j) = \sum_{x \in \tilde{x}_i} \sum_{y \in \tilde{y}_j} p(x, y)$, while the marginals are obtained through standard marginalization: $p(\tilde{x}_i) = \sum_{\tilde{y}_j \in y^c} p(\tilde{x}_i, \tilde{y}_j)$ and, analogously, $p(\tilde{y}_j) = \sum_{\tilde{x}_i \in x^c} p(\tilde{x}_i, \tilde{y}_j)$.

Our optimization procedure in the resulting model starts with some initial values x_0^c and y_0^c of X^c and Y^c, respectively. First, we fix y_0^c and perform a local search around x_0^c that aims at maximizing the objective (3). The specific local search procedure we use is hill climbing, performed by moving a data instance x from its current cluster to another for which the objective is maximal (or keeping x in its current cluster in case of having no better option). After some while (e.g. when the objective reaches its plateau, or when all the data instances have been tested, etc.), we fix the new clustering x_1^c and switch to optimizing Y^c, and then go on over the variables in a round robin fashion.[3] After T such iterations, clustering x_T^c and y_T^c come to an agreement with each other. For the sake of our current work on clustering agreement, we build m models like that (for an illustration, see Figure 1, left).

Bekkerman et al. [4, 3] showed that such a bi-modal clustering algorithm has a great potential in clustering multidimensional data, but adding more modalities in most cases improves the clustering results even more, as it improves the model's regularization and reduces its tendency to overfitting. We claim that a similar effect can be achieved even with only two data modalities—when more than one clustering result is available. Since the original bi-modal clustering algorithm is randomized,[4] each of its run is likely to produce a different result that captures different signals coming from the data. We may take the best from each of the constructed models by letting them agree with each other. To initiate such an agreement process, we connect all the $X_i^c|_{i=1}^m$ nodes of our original models with all the $Y_j^c|_{j=1}^m$ nodes, which results in a bipartite graph (see Figure 1, right).

Over each edge e_{ij} of the resulting Comraf model, we define a Mutual Information log-potential $f_{ij}(x_i^c, y_j^c) = I(\tilde{X}_i; \tilde{Y}_j)$. We then derive the new MPE inference process from Equation (2) as follows:

$$(\mathbf{x}_*^c, \mathbf{y}_*^c) = \arg\max_{\mathbf{x}^c, \mathbf{x}^c} \sum_{e_{ij} \in \mathcal{E}} f_{ij}(x_i^c, y_j^c) = \arg\max_{\mathbf{x}^c, \mathbf{x}^c} \sum_{i=1}^m \sum_{j=1}^m I(\tilde{X}_i; \tilde{Y}_j). \qquad (4)$$

To perform this optimization, we apply the Iterative Conditional Mode (ICM) inference algorithm [9]. At each iteration of the ICM algorithm, one random variable from the MRF gets chosen, and the values of all the rest of the model get fixed. Next, the chosen variable is optimized with respect to the fixed values of its neighboring random variables. After this optimization step is over, we fix the final value of the chosen variable and move to optimizing another variable—and so on in a round robin. When only one variable gets optimized at a time, our objective from Equation (4) becomes linear in m. Suppose that we optimize variable x_i^c:

$$(x_i^c)_* = \arg\max_{x_i^c} \sum_{i=1}^m \sum_{j=1}^m I(\tilde{X}_i; \tilde{Y}_j) = \arg\max_{x_i^c} \sum_{j=1}^m I(\tilde{X}_i; \tilde{Y}_j). \qquad (5)$$

We notice however that rather than optimizing one variable at a time, we can employ parallelization. If we fix all the values of the $Y_j^c|_{j=1}^m$ variables in our bipartite Comraf, then all the $X_i^c|_{i=1}^m$ variables become conditionally independent such that we can optimize them in parallel. Once new values of the X_i^c variables are obtained, we fix them and move to optimizing the Y_j^c variables—in parallel again. We can perform this process (illustrated in Figure 2) until its convergence.

PROPOSITION 4.1. *The agreement process converges to a local maximum of the objective function from Equation (4).*

PROOF. The proof of the proposition is fairly straightforward. It is based on two observations. First, at a node level, the local search optimization does not decrease its objective in Equation (5), because any data instance is moved from its cluster to another only if the objective is increased. Since the global objective from Equation (4) is a sum of these local objectives, then it cannot be decreased either. The second observation is that the global objective remains unchanged at the transition time (when one set of variable values gets fixed and we switch to optimizing the second set of variables). Since our global objective is a sum of Mutual Information terms that are bounded from above, and the presented process never decreases its objective, it must reach its local maximum—which will stop the process. □

The importance of parallelization should not be underestimated in the case of our clustering agreement process—it allows up to apply our method to large data collections.

[3]Note that the requirement on the number of clusters k to be fixed during the optimization process does not hold at the transition stage (when we switch between X^c and Y^c). At that stage we are free to split or merge clusters, which allows exploring clustering hierarchy.

[4]The randomization can come from the clustering initialization, as well as from ordering the data instances to test for a potential relocation.

5. EXPERIMENTAL SETUP

We evaluate our methods on two unsupervised learning tasks: (a) document/word clustering, where the goal is to construct topically related groups of documents, while simultaneously constructing groups of words—which presumably represent topics; (b) user/movie clustering for the collaborative filtering purposes, where the goal is to recommend a movie to a user based on collaborative behavior of the users' community. On both tasks, we measure the clustering quality as well as the clustering stability (using criteria defined in Section 5.2), as a relative improvement that CAP provides with respect to the original clusterings. We also measure an improvement in stability and quality of a consensus clustering method, when applied to the clusterings *after* CAP was applied on them. For this purpose, we use the Stable Component Consensus Clustering (SCCC) method discussed in Section 5.3. We come up with three experimental setups (CAP, stand-alone SCCC, and SCCC applied after CAP) and test them on four data collections, three of which are document collections, and the fourth is the user/movie dataset (see Section 5.1). We provide technical details on the underlying clustering methods in Section 5.4.

5.1 Datasets

For the task of document clustering, we use three publicly available datasets. Two of them (called *sanders-r* and *kitchen-l*) are relatively large email directories of two former Enron employees, derived from the Enron Email Dataset.[5] The *sanders-r* collection contains 1188 email messages stored in 30 folders. The *kitchen-l* collection contains 4015 messages stored in 47 folders.[6] The document clustering task on those collections aims at reconstructing the email folders. The third document collection is the benchmark *20 Newsgroups* dataset, which consists of 19997 newsgroup postings, classified into 20 categories. It is known that about 4.5% of the postings are duplications of other postings—however, for better replicability, we do not exclude the duplicated postings. A particular preprocessing scheme of these datasets is described in Bekkerman et al. [2].

For the collaborative filtering task, we use the Netflix data as it was used for the Netflix KDD'07 Cup.[7] For this competition, the task was set up as predicting which movies would be rated by which users, without predicting the actual ratings. The training data consists of about 100M user/movie pairs (positive instances) for 480,189 users and 17770 movies. The hold-out data consists of 100,000 pairs, 7.8% of which are positive instances. We simultaneously construct both a clustering of users and a clustering of movies, and then use the resulting clusterings to rank the test instances. Our ranking function $r(u, m)$ for a user u and a movie m is composed of the popularity score $p(u)p(m)$, lifted with the information gained at the clustering process:

$$r(u, m) = p(u)p(m)\frac{p(\tilde{u}, \tilde{m})}{p(\tilde{u})p(\tilde{m})},$$

when the user u was assigned into a cluster $\tilde{u}$, and the movie m was assigned into a cluster $\tilde{m}$ (for a discussion, see [5]).

[5]`http://www.cs.cmu.edu/~enron`

[6]Preprocessed versions of the *sanders-r* and *kitchen-l* datasets can be downloaded from `http://www.cs.umass.edu/~ronb/enron_dataset.html`.

[7]`http://cs.uic.edu/~liub/Netflix-KDD-Cup-2007.html`

5.2 Evaluation criteria

For assessing the quality of document clustering we use the *clustering accuracy* measure, as used in previous work [10, 2]. Let x^c be a clustering of a document collection $\mathcal{X}$. Let $\mathcal{T}$ be the set of ground truth categories. For each cluster $\tilde{x}$ of x^c, let $n_{\mathcal{T}}(\tilde{x})$ be the maximal number of $\tilde{x}$'s elements that belong to one category. Then, precision $Prec(\tilde{x}, \mathcal{T})$ of the cluster $\tilde{x}$ with respect to $\mathcal{T}$ is defined as $Prec(\tilde{x}, \mathcal{T}) = n_{\mathcal{T}}(\tilde{x})/|\tilde{x}|$. The micro-averaged precision of the entire clustering x^c is:

$$Prec(x^c, \mathcal{T}) = \frac{\sum_{\tilde{x}} n_{\mathcal{T}}(\tilde{x})}{\sum_{\tilde{x}} |\tilde{x}|} = \frac{\sum_{\tilde{x}} n_{\mathcal{T}}(\tilde{x})}{n}, \qquad (6)$$

where n is the size of the dataset. In words, $Prec(x^c, \mathcal{T})$ is the portion of documents classified into the dominant categories. Note that this measure is meaningless when the number of clusters k is large. In fact, if k equals the number of data points, the micro-averaged precision is 1. In all our experiments, we choose the number of clusters to be equal to the number of ground truth categories: $k = |\mathcal{T}|$. And when $k = |\mathcal{T}|$, the micro-average precision $Prec(x^c, \mathcal{T})$ equals *micro-averaged clustering accuracy*, or just *clustering accuracy*, for short.

For evaluating our collaborative filtering results based on the constructed clusterings of users and movies, we follow Bekkerman and Scholz [5] who compute the *Area Under the ROC Curve* (or *AUC*, in short) for a constructed ranking of the user/movie pairs (see Section 5.1 for details on our ranking scheme).

To evaluate the stability of a clustering method, we use the Jaccard index [12] averaged over all $\frac{m(m-1)}{2}$ pairs of m outcomes of this method. Given a pair of clusterings, x_1^c and x_2^c, we define a as the number of data instance pairs that belong to the same cluster in x_1^c as well as in x_2^c. We define b as the number of data instance pairs that belong to the same cluster in x_1^c but not in x_2^c. We define c as the number of data instance pairs that belong to the same cluster in x_2^c but not in x_1^c. The Jaccard index between x_1^c and x_2^c is then defined as:

$$J(x_1^c, x_2^c) = \frac{a}{a + b + c}.$$

Note that the Jaccard index ranges between 0 and 1. Our underlying clustering methods (Section 5.4) are randomized, such that, when applied m times, they produce different results. We compute the averaged Jaccard index over those results, assessing by this the stability of the method. The stability of the SCCC method (see 5.3 below) is assessed by applying it a number of times with different values of its parameter, and computing the averaged Jaccard index over the outcomes. Evaluating the stability of our CAP method is straightforward: it is designed to produce m outcomes, over which we compute the averaged Jaccard index.

5.3 Stable Component Consensus Clustering

One of the applications of our clustering agreement process is in improving stability and accuracy of consensus clustering. A possible scenario for this task would be to come up with the set of original clusterings, let them interact via our clustering agreement mechanism, and then apply a consensus clustering method that would presumably be more accurate and more stable than a consensus of the original clusterings. To evaluate this hypothesis, we need to have a consensus clustering algorithm available for experimentation. There are many off-the-shelf consensus clustering algorithms (see, e.g., [17]), however, we did not find any existing algorithm that would have the following property: it is desired that each cluster in the resulting consensus clustering would be "labeled" by a large portion of data instances that compose a *stable component*, i.e. are often, if not always, clustered together.

Let us first provide some intuition. In Section 5.2 we presented the clustering accuracy measure as the micro-averaged accuracy of each cluster, which is computed as a portion of the cluster having the most common label. For example, in document clustering, if a large portion of a cluster is classified as, say, *sports news*, then we assume that the entire cluster is about sports while the cluster members that were not classified as sports news are considered noise. All the above refers to the evaluation stage, where the ground truth labels are available. At the consensus clustering stage, however, the task remains completely unsupervised, i.e. there is no ground truth. Nevertheless, the notion of "large data portions" still exists. If a large group of data instances is always clustered together, while not having a strong correlation pattern with other groups, then this group can be a good candidate for seeding a cluster. If all clusters can be seeded with such groups, then these groups will impose "implicit labels" on their corresponding clusters, which will be very helpful in further data analysis.

Lacking an existing consensus method that would have such a property, we came up with a method of our own, which is very simple, very efficient, and still effective. We call it *Stable Component Consensus Clustering (SCCC)* and note that it is somewhat similar to the Iterative Voting Consensus (IVC) method proposed by Nguyen and Caruana [17]. The SCCC method operates in two stages: first, it seeds clusters with large stable components that are distant from each other, and second it associates all the other stable components with one of the seeds. Let us provide more details.

DEFINITION 5.1. *Given an ordered list of m clusterings $(x_1^c, x_2^c, \ldots, x_m^c)$ and a data instance x, we define a* signature *of x as an ordered list of cluster identities to which x belongs in each of the m clusterings: $s(x) = (\tilde{x}_{i_1}, \tilde{x}_{i_2}, \ldots, \tilde{x}_{i_m})$.*

DEFINITION 5.2. *A* stable component *is the largest subset of data instances each of which have the same signature.*

A distance between two stable components is then defined as a Hamming distance between their signatures. Our SCCC algorithm works as follows:

1. Construct all stable components of the data. Construct their ranked list R in which they are sorted in decreasing order of their sizes.

2. Traverse the ranked list R and test each component for being able to serve as a cluster seed. Choose a component to seed a cluster if it is at least at distance g from each previously assigned cluster seed. If the component is chosen as a cluster seed, exclude it from R. If the number of cluster seeds reaches the number of clusters k, stop.

3. Traverse the ranked list R again, this time assigning each remaining component to a cluster whose seed is the closest one to the component being assigned.[8]

The main advantage of the SCCC algorithm is in its efficiency—its time complexity is linear in the number of data instances. Its main drawback is that it has a free parameter g, which we call the *seed gap* parameter. The values of the seed gap parameter are integers in the $[0..m]$ range. We do not attempt to set it up in the optimal way, instead, we apply the SCCC algorithm with all possible seed gap values[9] and then average the results. This provides a notion of stability of the SCCC algorithm with respect to tuning its parameter, and as well as a notion of statistical significance of its results.

Other possible limitations of the SCCC algorithm include its instability for very large values of m: if an ensemble consists of very many clusterings, it is highly probable that stable components will be small, and then the seeding process will be almost random. In our experiments, however, we choose $m = 10$, which is small enough to allow many large stable components. Also, the seeding procedure can be almost random if the original clusterings are very different from each other—but it is exactly the problem that we solve with our clustering agreement process.

It is important to note that in this paper we do not focus on consensus clustering, and therefore do not intend to propose the most effective consensus clustering method. Our goal is to show that CAP can improve consensus clustering, applied *on top* of CAP's results, as compared to the *same* consensus clustering method applied *without* CAP.

5.4 Underlying clustering methods

In our document clustering setup, we use the *Multi-way Distributional Clustering (MDC)* tool [2, 4] to produce original clusterings. MDC is an open source tool,[10] which is able to perform simultaneous clustering of multiple modalities, organized in a Comraf model of arbitrary topology. First, we apply it to construct initial clusterings, using the Comraf topology from Figure 1 (left). MDC allows hierarchical clustering, which is proved to be very effective [2]. One of its advantages is that it is not heavily dependent on the clustering initialization. We perform top-down clustering of words and, simultaneously, bottom-up clustering of documents. The top-down scheme is initialized by assigning all the words in one cluster. The bottom-up scheme is initialized by having one cluster per document. At each MDC's iteration, we first split each word cluster to two, and then perform the local search (as described in Section 4) to maximize the objective (3). Then we merge each two clusters of documents and perform the same optimization over the resulting document clustering. We continue this process until we reach the required number of document clusters. At the first iteration of the algorithm, we perform three split-and-optimize procedures over word clusters (ending up with eight clusters), before we start with the merge-and-optimize procedure over the document clusters.

For our clustering agreement process, we apply MDC on the bipartite Comraf from Figure 1 (right). We initialize each clustering node with a clustering obtained at the bi-modal stage, and perform *flat* (i.e. non-hierarchical) clus-

[8]We also tested a centroid-based version of the SCCC algorithm, in which each cluster's centroid is updated after a stable component is added. The resulting version did not show significantly better results on our test datasets.

[9]In some cases, the seed gap g can be too large such that not all k clusters can be seeded. In those cases, we disregard such a value of g.

[10]`http://comraf.sourceforge.net`

Table 1: Results on document datasets. Accuracies of the initial clustering and of CAP are averaged over $m = 10$ runs. Accuracies of SCCC are averaged over the number of valid values of the seed gap parameter g (which is between 8 and 10). Jaccard index values are averaged over all the clustering pairs. The standard error of the mean is given after the ± sign. Relative improvements with respect to the original clustering results are shown in brackets.

Measure	Before	After CAP	After SCCC	After CAP and SCCC
***sanders-r* dataset:**				
Averaged accuracy of document clusterings	0.6594 ± 0.0051	0.6745 ± 0.0036 (+2.2%)	0.6763 ± 0.0028 (+2.5%)	0.6826 ± 0.0018 (+3.5%)
Averaged Jaccard index of document clusterings	0.4186 ± 0.0062	0.6435 ± 0.0099 (+53.7%)	0.6831 ± 0.0266 (+63.1%)	0.7242 ± 0.0306 (+73.0%)
Averaged Jaccard index of word clusterings	0.1743 ± 0.0013	0.3453 ± 0.0034 (+98.1%)	0.3459 ± 0.0264 (+98.4%)	0.5024 ± 0.0315 (+188.2%)
***kitchen-l* dataset:**				
Averaged accuracy of document clusterings	0.4053 ± 0.0035	0.4350 ± 0.0017 (+7.3%)	0.4214 ± 0.0060 (+3.9%)	0.4481 ± 0.0016 (+10.5%)
Averaged Jaccard index of document clusterings	0.1757 ± 0.0020	0.3784 ± 0.0071 (+115.3%)	0.5576 ± 0.0262 (+217.3%)	0.7067 ± 0.0214 (+302.2%)
Averaged Jaccard index of word clusterings	0.2038 ± 0.0014	0.3696 ± 0.0027 (+81.3%)	0.3202 ± 0.0308 (+57.1%)	0.4476 ± 0.0405 (+119.6%)
***20 Newsgroups* dataset:**				
Averaged accuracy of document clusterings	0.7203 ± 0.0044	0.7562 ± 0.0047 (+4.9%)	0.7388 ± 0.0083 (+2.5%)	0.7523 ± 0.0067 (+4.4%)
Averaged Jaccard index of document clusterings	0.4924 ± 0.0061	0.7404 ± 0.0133 (+50.3%)	0.7732 ± 0.0292 (+57.0%)	0.8556 ± 0.0165 (+73.7%)
Averaged Jaccard index of word clusterings	0.1006 ± 0.0005	0.6430 ± 0.0010 (+539.1%)	0.2639 ± 0.0363 (+162.3%)	0.4607 ± 0.0395 (+357.9%)

tering until the changes in the value of the objective (4) are small enough. In practice, it turned out that only a few optimization iterations are required to reach convergence. For consistency, we fixed the number of iterations to four.

The main disadvantage of the MDC method is that it is inherently sequential and time-consuming. It cannot be applied to the large Netflix dataset. Instead, we apply its parallelized version, called DataLoom [5]. The DataLoom algorithm optimizes the same objective, however, it is not hierarchical, therefore, at the bi-modal stage, we randomly initialize user and movie clusterings. We fix the number of clusters at 800—for both cases. At the CAP stage, however, there is no principal difference between DataLoom and MDC. We performed only two CAP iterations.

6. RESULTS

Our results on the document clustering task are summarized in Table 1. As can be seen from the table, on all the three datasets CAP significantly improves both accuracy and stability of the original clustering algorithm. The stability improvements are quite impressive. An interesting case is the stability improvement of word clustering on the *20 Newsgroups* dataset. CAP managed to increase the averaged Jaccard index from about 0.1 to over 0.6. We decided to study this case more closely, and constructed all stable components before and after the CAP application. Even a quick scan through the results showed dramatic changes. For example, one stable component of the post-CAP result consists of the following 15 words:

```
altar       materialistic   revelation
apprentice  metaphorical    teachings
bible       narratives      theologians
gospel      philistines     tierra
inerrant    recite          unbeliever
```

Checking on these words in the pre-CAP data, we figured out that each of them composed a stable component of size one, i.e. was disconnected from any other. The stability and accuracy improvement per CAP's iteration are shown in Figure 4. As we can see, the first two iterations are most crucial, after which the curves become flatter.

From the right two columns of Table 1 we can learn that CAP also improves both stability and accuracy of consensus clustering. The improvement on words is always more substantial than the improvement on documents, while the absolute values on words are always lower than on documents—possibly because the word modality is larger than the document modality, and the number of word clusters is larger than the number of document clusters.

We have to note that, technically, stability of our consensus method (with or without CAP) cannot be directly compared with the stability of the original clustering (again, with or without CAP), because SCCC is a deterministic method, such that its stability is measured over various values of its parameter g, while the original clustering is randomized. Nevertheless, we added the relative improvement values, for consistency with the rest of the table.

In contrast, the accuracy values are comparable between all the listed methods. Although relative improvements do not look as dramatic as for the stability measure, we should point out that in two of the three cases we managed to obtain the state-of-the-art results. Specifically, the 44.8% accuracy obtained by the consensus method with the help of CAP on the *kitchen-l* dataset, as well as over 75% accuracy shown on the *20 Newsgroups* are, to our knowledge, the best results ever obtained on these data collections.

It is interesting to see how the CAP system responds to changes in m—the number of participating clusterings. Figure 3 shows such a result on the *20 Newsgroups.* First, it turns out that as few as just three participants can already

Table 2: Results of the Clustering Agreement Process on the Netflix dataset.

Measure	Before Agreement	After one iteration	After two iterations	Relative improvement
Averaged AUC	0.73128 ± 0.00023	0.73172 ± 0.00020	0.73201 ± 0.00023	+0.1%
Averaged Jaccard index of user clusterings	0.1488 ± 0.0014	0.1658 ± 0.0015	0.1716 ± 0.0014	+15.4%
Averaged Jaccard index of movie clusterings	0.2745 ± 0.0024	0.3158 ± 0.0024	0.3321 ± 0.0022	+21.0%

Table 3: Improving stability of clustering consensuses on the *sanders-r* dataset: we show that CAP consistently leads to similar agreements, when applied to different ensembles of clusterings.

Measure	Before CAP	After CAP	Relative improv.
Averaged accuracy of doc consensuses	$.6771 \pm .0033$	$.6870 \pm .0027$	+1.5%
Averaged Jaccard of doc consensuses	$.5365 \pm .0090$	$.6186 \pm .0057$	+15.3%
Averaged Jaccard of word consensuses	$.2163 \pm .0018$	$.3310 \pm .0028$	+53.0%

boost the accuracy by 2%. Second, as m goes up, the standard error of clustering accuracy decreases, and the after-CAP curve consistently rises while the before-CAP curve fluctuates.

Our stability and quality results on the Netflix dataset are shown in Table 2. As one can notice, CAP obtains a very modest improvement in the AUC measure. Although being statistically significant, it cannot be considered a success. However, we should note that improving AUC on this dataset is a very challenging task, and half-percent improvements are considered a valuable achievement (see [5]). Nonetheless, the stability improvement is substantial, especially taking into account the size of the data.

6.1 Consistency of clustering agreement

As we have shown, our clustering agreement process is able to significantly improve stability of clustering methods. An important question still to ask is whether CAP is *consistent* in its behavior, i.e. does it tend to lead to the same (or similar) solutions, independently of the initial clusterings? To answer this question, we set up the following experiment. Over the same data, we construct M clustering ensembles, and find their M consensuses (using our SCCC method with $g = 5$). Afterwards, we apply CAP to each of the ensembles, and find M consensuses of the after-CAP versions. Then we compute the stability of the two ensembles of consensuses—before CAP and after CAP. The results for the *sanders-r* dataset (with $M = 10$) are shown in Table 3. To our satisfaction, the after-CAP ensemble of consensuses is much more stable (and, as a matter of fact, is also slightly more accurate). This supports our hypothesis that CAP's behavior is consistent.

7. CONCLUSION

In this paper, we investigated a novel machine learning problem, the Improvement of Clustering Stability, which can play a key role in data analysis and other areas of data mining. We proposed an effective method for this problem's solution, the Clustering Agreement Process (CAP), which is based on inference in a new type of Combinatorial MRFs—of bipartite topology, with multiple nodes per data modality. We proposed an efficient, parallelized method for performing this inference, and showed its convergence. As a byproduct of this research, we proposed a Stable Component Consensus Clustering method, which has a property of "implicit labeling" of the constructed clusters. We applied our CAP method to four real-world datasets and obtained significant improvements in clustering stability, as well as in clustering quality, of both initial clusterings and their consensuses.

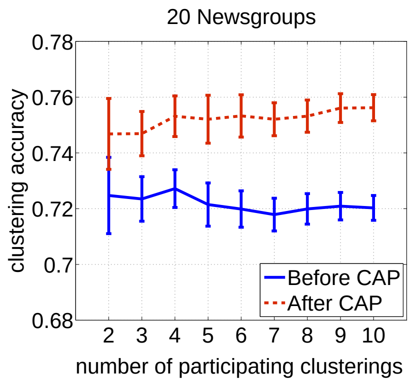

Figure 3: Clustering accuracy on the *20 Newsgroups* dataset, as a function of m—the number of clusterings which participate in the agreement process.

8. ACKNOWLEDGEMENTS

The authors thank Rajan Lukose for fruitful discussions at the early stages of this project. Ron thanks his wife Anna for her constant support.

9. REFERENCES

[1] L. Badea. Clustering and metaclustering with nonnegative matrix decompositions. In *Proceedings of ECML-16*, pages 10–22, 2005.

[2] R. Bekkerman, R. El-Yaniv, and A. McCallum. Multi-way distributional clustering via pairwise interactions. In *Proceedings of ICML-22*, 2005.

[3] R. Bekkerman and J. Jeon. Multi-modal clustering for multimedia collections. In *Proceedings of CVPR'07, the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007.

[4] R. Bekkerman, M. Sahami, and E. Learned-Miller. Combinatorial Markov Random Fields. In *Proceedings of ECML-17*, 2006.

[5] R. Bekkerman and M. Scholz. Data weaving: Scaling up the state-of-the-art in data clustering. In *Proceedings of CIKM-17*, pages 1083–1092, 2008.

[6] S. Ben-David, U. von Luxburg, and D. Pál. A sober look at clustering stability. In *Proceedings of COLT-19*, pages 5–19, 2006.

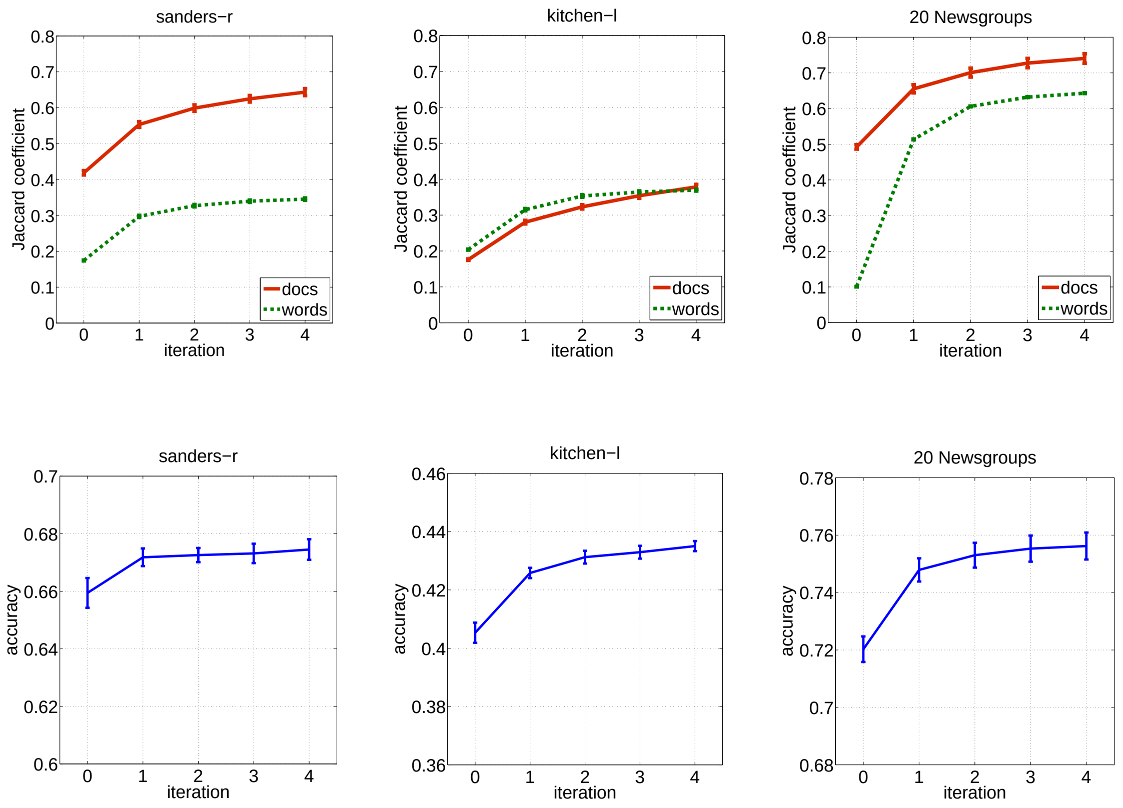

Figure 4: Stability in terms of the averaged Jaccard index (top) and document clustering accuracy (bottom) results on *sanders-r* (left), *kitchen-l* (middle), and the *20 Newsgroups* (right), as a function of the number of iterations in the clustering agreement process. The standard error of the mean is shown as errorbars of every plot, however, in some cases the error is very small and thus barely seen. The results of iteration 0 are the original clustering results, before CAP was applied.

[7] A. Ben-Hur, A. Elisseeff, and I. Guyon. A stability based method for discovering structure in clustered data. *Pacific Symposium on Biocomputing*, pages 6–17, 2002.

[8] J. Besag. Spatial interaction and statistical analysis of lattice systems. *Journal of the Royal Statistical Society*, 36(2):192–236, 1974.

[9] J. Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society*, 48(3), 1986.

[10] I. S. Dhillon, S. Mallela, and D. S. Modha. Information-theoretic co-clustering. In *Proceedings of SIGKDD-9*, pages 89–98, 2003.

[11] X. Z. Fern and C. Brodley. Solving cluster ensemble problems by bipartite graph partitioning. In *Proceedings of ICML-21*, 2004.

[12] P. Jaccard. The distribution of flora in the alpine zone. *New Phytologist*, 11:37–50, 1912.

[13] A. Kreiger and P. Green. A generalized rand-index method for consensus clustering of separate partitions of the same data base. *Journal of Classification*, 16:63–89, 1999.

[14] L. Kuncheva and D. Vetrov. Evaluation of stability of k-means cluster ensembles with respect to random initialization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11):1798–1808, 2006.

[15] T. Lange, V. Roth, M. Braun, and J. Buhmann. Stability-based validation of clustering solutions. *Neural Computation*, 16(6):1299–1323, 2004.

[16] E. Levine and E. Domany. Resampling method for unsupervised estimation of cluster validity. *Neural Computation*, 13(11):2573–2593, 2001.

[17] N. Nguyen and R. Caruana. Consensus clusterings. In *Proceedings of ICDM-7*, pages 607–612, 2007.

[18] K. Punera and J. Ghosh. Consensus-based ensembles of soft clusterings. *Applied Artificial Intelligence*, 22(7-8):780–810, 2008.

[19] V. Raghavan and M. Y. L. Ip. Techniques for measuring the stability of clustering: a comparative study. In *Proceedings of SIGIR-5*, pages 209–237, 1982.

[20] W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

[21] N. Slonim, N. Friedman, and N. Tishby. Multivariate information bottleneck. *Neural Computation*, 18(8):1739–1789, 2006.

[22] A. Strehl and J. Ghosh. Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3:583–617, 2003.