Decentralized Probabilistic Text Clustering

Odysseas Papapetrou, Wolf Siberski, and Norbert Fuhr

Abstract—Text clustering is an established technique for improving quality in information retrieval, for both centralized and distributed environments. However, traditional text clustering algorithms fail to scale on highly distributed environments, such as peer-to-peer networks. Our algorithm for peer-to-peer clustering achieves high scalability by using a probabilistic approach for assigning documents to clusters. It enables a peer to compare each of its documents only with very few selected clusters, without significant loss of clustering quality. The algorithm offers probabilistic guarantees for the correctness of each document assignment to a cluster. Extensive experimental evaluation with up to 1 million peers and 1 million documents demonstrates the scalability and effectiveness of the algorithm.

Index Terms—Distributed clustering, text clustering

1 Introduction

Text clustering is widely employed for automatically structuring large document collections and enabling cluster-based information browsing, which alleviates the problem of information overflow. It is especially useful in highly distributed environments such as distributed digital libraries [1] and peer-to-peer (P2P) information management systems [2], since these environments operate on large-scale document collections, scattered over the network. Existing P2P systems also employ text clustering to enhance information retrieval efficiency and effectiveness [3], [4], [5]. Hence, a distributed clustering algorithm that scales to large networks and large text collections is required.

Most existing text clustering algorithms are designed for central execution. They require that clustering is performed on a dedicated node, and are not suitable for deployment over large scale distributed networks. Therefore, specialized algorithms for distributed and P2P clustering have been developed, such as [6], [7], [8], [9]. However, these approaches are either limited to a small number of nodes, or they focus on low dimensional data only.

In this work, we are particularly interested in systems where the content distribution is imposed by their nature, such as P2P desktop sharing systems [2], distributed digital libraries [1], and mainstream file sharing P2P networks. Clustering in such networks is challenging, firstly because the data is inherently distributed and no participant has the capacity, or willingness, to collect and process all data, and secondly, because of the high churn rate, affecting availability of content and of computational nodes. For these systems, we require a P2P algorithm that can cluster distributed and highly dynamic text collections, without overloading any of the participating peers, and without requiring central coordination.

A key factor to reduce network traffic in these systems is to reduce the number of required comparisons between documents and clusters. Our approach achieves this by applying probabilistic pruning: Instead of considering all clusters for comparison with each document, only a few most relevant ones are taken into account. We apply this core idea to K-Means, one of the frequently used text clustering algorithms. The proposed algorithm, called Probabilistic Clustering for P2P (PCP2P), reduces the number of required comparisons by an order of magnitude, with negligible influence on clustering quality.

In the following section, we review the basic algorithms employed by PCP2P. Section 3 gives an overview of existing distributed clustering approaches. In Sections 4 and 5 we introduce PCP2P and the corresponding cost analysis. We present the probabilistic analysis in Section 6, and show how PCP2P is parameterized to achieve a desired correctness probability. In Section 7 we experimentally verify the scalability and quality of PCP2P, with setups of up to 1 million peers and 1 million documents, using real and synthetic data. We conclude in Section 8.

2 Background

In this section we briefly describe the infrastructure and algorithms used by PCP2P.

Distributed Hash Tables. PCP2P relies on a Distributed Hash Table (DHT) infrastructure. DHTs provide efficient hash table capabilities in a P2P environment by arranging peers in the network according to a specific graph structure, usually a hypercube. DHTs offer the same functionality as their standard hash table counterparts, a `put(key,value)` method, which associates `value` with `key`, and a `get(key)` method, which returns the value associated with `key`.

- *O. Papapetrou and W. Siberski are with L3S Research Center, Hannover, Germany.*
  E-mail: {papapetrou, siberski}@L3S.de
- *N. Fuhr is with the Faculty of Engineering Sciences, University of Duisburg-Essen, Germany.*
  E-mail: norbert.fuhr@uni-due.de

- *Accepted at 24th April, 2011. Copyright held by IEEE, Transactions of Knowledge and Data Engineering. The printed version will appear at http://www.computer.org/portal/web/tkde/*

Algorithm 1 DKMeans

```
1: allClusters ← findAllClusters()
2: for Document d in my documents do
3:     for Cluster c in allClusters do
4:         Send termVector(d) to ClusterHolder(c)
5:         Sim[c] ← Retrieve similarity(d,c)
6:     end for
7:     Assign(d, $\text{argmax}_c$ Sim[c])
8: end for
```

Both methods induce a cost of $O(\log(n))$ messages, where n is the number of peers. Existing DHT infrastructures, like Chord [10], P-Grid [11], Pastry [12], have methods to perform load balancing and to address node failures, such that the index does not suffer from bottlenecks and is robust to churn. Our implementation uses Chord [10], but any other DHT could be employed as well.

Document model. Similar to most clustering algorithms, PCP2P represents the documents and clusters using the Vector Space Model [13]. For document d and term t, we denote the frequency of t in d with $TF(t, d)$. As usual, we apply term stemming and stopword filtering to reduce the document model sizes and allow for more meaningful clustering. Since the Vector Space Model represents clusters and documents as high-dimensional vectors, all standard norms can be used for computing the vector length. The ones most interesting for data mining, which are frequently used for normalizing the term frequencies, are the L1-Norm and the L2-Norm. The L1-Norm of a cluster or document x is denoted as $|x|_1$ and is defined as the sum of all term frequencies in the cluster/document, $|x|_1 = \sum_{t \in x} TF(t, x)$. The L2-Norm is denoted as $|x|$ and defined as $|x| = \sqrt{\sum_{t \in x} TF^2(t, x)}$.

K-Means Clustering. K-Means, which we approximate with PCP2P, is one of the most frequently used clustering algorithms because of its low complexity (linear in the number of documents) and high clustering quality, particularly for text clustering [14]. The basic K-Means algorithm can be summarized as follows: (1) Select k random starting points as initial centroids for the k clusters. (2) Assign each document to the cluster with the nearest centroid. (3) Recompute the centroid of each cluster as the mean of all cluster documents. (4) Repeat steps 2-3 until a stopping criterion is met, e.g., no documents change clusters anymore.

DKMeans Clustering. A direct distribution of centralized clustering algorithms does not scale to large networks. We still sketch such an approach here, to point out where our optimizations take place. Like its centralized counterpart, distributed K-Means (*DKMeans*) requires maintaining the cluster centroids, and comparing all documents to all clusters to determine the best cluster. In DKMeans, the responsibility of maintaining each cluster centroid is assigned to a randomly selected peer, called *cluster holder*. As in K-Means, each cluster holder selects a random document as initial centroid. Clustering is performed as follows (cf. Algorithm 1): A peer first identifies all cluster holders, e.g., using a DHT index (Line 1). Then, for each of its documents, it sends its term vector to each cluster holder for comparison with the centroid and receives a similarity score (Lines 4, 5). It then assigns the document to the most similar cluster, by notifying the respective cluster holder (Line 7).

DKMeans requires that at each clustering iteration, all documents are sent over the network to all cluster holders, for comparison. This clearly prohibits the algorithm to scale. In Section 4 we show how PCP2P addresses this issue by drastically reducing the number of required comparisons.

3 Related Work

Several algorithms for parallelizing K-Means have been proposed, e.g., [15], to harness the power of multiple nodes for the clustering of large datasets. These however assume a controlled network or a shared memory architecture, and are not applicable for P2P, where these assumptions do not apply.

Several works focus on P2P text clustering [16], [17], [6], [7], [8], [9], [18]. Eisenhardt et al. [7] proposed one of the first P2P clustering algorithms. The algorithm distributes K-Means computation by broadcasting the centroid information to all peers. However, due to this centroid broadcasting, it does not scale to large networks. Hsiao and King [9] avoid broadcasting by employing a DHT to index all clusters using manually selected terms. This approach requires extensive human interaction for selecting the terms, and the algorithm cannot adapt to new topics.

Hammouda and Kamel [8] propose a hierarchical topology for distributing K-Means. Clustering starts at the lowest level of the hierarchy, and the local solutions are aggregated until the root peer is reached. This algorithm has the disadvantage that clustering quality decreases noticeably for each aggregation level, because of the random grouping of peers at each level. Therefore, quality decreases significantly for large networks. Already for a network of 65 nodes organized in three levels, the authors report a drastic drop in quality; the entropy measure for clustering the RCV1 dataset rises to 0.7, compared to 0.35 for a centralized K-Means. For more levels, the entropy almost reaches 1, denoting a random grouping. In [19], another distributed clustering algorithm is proposed, where peers exchange cluster summaries containing extracted key phrases from the cluster. Even though these summaries are compact, each peer needs to send them to all other peers, requiring $O(n^2)$ messages for a network of size n. This approach is therefore only suitable for very small networks (the reported experiments are with up to 7 nodes, and with less than 20 thousand documents).

The state of the art in P2P clustering is the proposal of Datta et al. [17], [6]. In particular, the authors proposed LSP2P and USP2P, two P2P approximations of K-Means. LSP2P uses gossiping to distribute the centroids. In an evaluation with 10-dimensional data, LSP2P achieved an average misclassification error of less than 3%. However, as we show in Section 7, when it comes to clustering text, LSP2P fails because it is based on the assumption that data is uniformly distributed among the peers, i.e., each peer has a representative set of documents from each cluster. This assumption clearly does not hold for text collections in P2P networks. The second algorithm, USP2P, uses sampling to provide probabilistic guarantees, which however are also based on the same assumption. Furthermore, USP2P requires a coordinating peer which gets easily overloaded, since it is responsible for exchanging centroids with a significant number of peers, for sampling, e.g., 500 peers out of 5500 peers.

A separate research stream is the usage of clustering to improve query routing efficiency. Peers are clustered by topic, and queries are primarily routed to members of the right cluster. The first P2P IR systems employing this approach were [3], and [4]; however, they do not use a distributed clustering algorithm, but perform clustering on a dedicated node. Koloniari and Pitoura [18] use game-theoretic insights to let the peers form clusters for improved retrieval performance. Papapetrou et al. [5] speed up the maintenance of distributed inverted indices with peer clustering. Note that these works – in contrast to general-purpose P2P clustering algorithms – are focused on efficiency gains and perform clustering on peer granularity instead of document level. Therefore they are not suitable for most other clustering applications.

4 PCP2P: Probabilistic Clustering for P2P

We start with a high-level description of the algorithm, followed by a detailed explanation of each step.

In PCP2P, a peer undertakes up to three different roles (Fig. 1). First, it serves as document holder, i.e., it takes care of clustering its documents. Second, it participates in the underlying DHT by holding part of the distributed index, and routing DHT lookup messages. Third, a peer may become a *cluster holder*, i.e., maintain the centroid and document assignments for one cluster.

PCP2P consists of two parallel activities, *cluster indexing* and *document assignment*. *Cluster indexing* is performed by the cluster holders. In regular intervals, these peers create compact cluster summaries and index them in the underlying DHT, using the most frequent cluster terms as keys. The second activity, *document assignment*, consists of two steps, preselection and full comparison. In the *preselection step*, the peer holding d retrieves selected cluster summaries

Algorithm 2 PCP2P: Clustering the documents

```
1: for Document d in my documents do
   PRESELECTION STEP:
2:     CandClusters ← CandidateClustersFromDHT()
   FULL COMPARISON STEP:
3:     RemainingClusters ← FilterOut(CandClusters)
4:     for Cluster c in RemainingClusters do
5:         Send termVector(d) to ClusterHolder(c)
6:         Sim[c] ← Retrieve similarity(d,c)
7:     end for
8:     Assign(d, cluster with maximum similarity)
9: end for
```

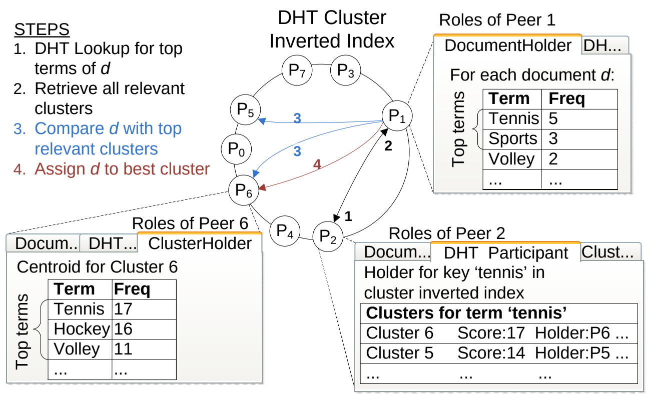
from the DHT index, to identify the most relevant clusters (Alg. 2, Line 2). Preselection already filters out most of the clusters. In the *full comparison step*, the peer computes similarity score estimates for d using the retrieved cluster summaries. Clusters with low similarity estimates are filtered out (Line 3), and the document is sent to the few remaining cluster holders for full similarity computation (Lines 4-7). Finally, d is assigned to the cluster with the highest similarity (Line 8). This two-stage filtering algorithm reduces drastically the number of full comparisons (usually less than five comparisons per document, independent of the number of clusters). Both cluster indexing and document assignment are repeated periodically to compensate churn, and to maintain an up-to-date clustering solution.

The algorithm enables controlling the tradeoff between the network cost and the clustering quality. In particular, the cluster indexing activity, as well as the preselection and full comparison steps, are configured using the results of a probabilistic analysis, thereby providing probabilistic guarantees that the resulting clustering solution exhibits nearly the same quality as centralized clustering.

In the next section, we further describe the process of indexing the cluster summaries. In Section 4.2 we look into the document assignment process, whereas in Section 4.3 we present three filtering strategies. Section 4.4 discusses further aspects of the algorithm.

4.1 Indexing of Cluster Summaries

Cluster holders are responsible for indexing the summaries of the clusters in the DHT. Particularly, each cluster holder periodically recomputes the cluster centroid, using the documents assigned to the cluster at the time. It also recomputes a *cluster summary* and publishes it to the DHT index, using selected cluster terms as keys. As we explain later, this enables peers to efficiently identify the relevant clusters for their documents. For this identification, it is sufficient to consider the most frequent terms of a cluster c as keys, i.e., all terms t with $TF(t,c) \geq CluTF_{min}(c)$, where $CluTF_{min}(c)$ denotes a frequency threshold for c. We use $TopTerms(c)$ to denote the set of these


STEPS
1. DHT Lookup for top terms of *d*
2. Retrieve all relevant clusters
3. Compare *d* with top relevant clusters
4. Assign *d* to best cluster
DHT Cluster Inverted Index
P7
P3
P5
P1
P0
P6
P4
P2
1
2
3
3
4
Roles of Peer 1
DocumentHolder
DH...
For each document *d*:
Top terms
Roles of Peer 6
Docum..
DHT...
ClusterHolder
Centroid for Cluster 6
Top terms
Roles of Peer 2
Docum...
DHT Participant
Clust...
Holder for key 'tennis' in cluster inverted index

Term	Freq
Tennis	5
Sports	3
Volley	2
...	...

Term	Freq
Tennis	17
Hockey	16
Volley	11
...	...

Clusters for term 'tennis'		
Cluster 6	Score:17	Holder:P6 ...
Cluster 5	Score:14	Holder:P5 ...
...	...	...

Fig. 1. PCP2P: System overview

terms. Note that $TopTerms(c)$ does not include stop-words; these are already removed when building the document vectors. For illustration purposes, for the remainder of this section we assume that $CluTF_{min}(c)$ is known. In Section 6 we explain how the optimal value for this threshold is derived, such that the algorithm satisfies the desired probabilistic guarantees.

The cluster summary includes: (1) all cluster terms in $TopTerms(c)$ and their corresponding TF values, (2) $CluTF_{min}(c)$, and, (3) the sum of all term frequencies (the L1-norm), cluster length (the L2-norm), and number of distinct cluster terms. We choose not to normalize the term frequencies with inverse document frequencies (IDF), because of the high cost associated for maintaining an IDF index over a P2P network, compared to the low influence in clustering quality [20]. Nevertheless, existing techniques for estimating IDF, such as [21], [22], or techniques to circumvent the lack of IDF, such as using the Inverse Peer Frequency [23] can also be combined with PCP2P.

To avoid overloading, each cluster holder selects random peers to serve as *helper cluster holders*, and replicates the cluster centroid to them. Their IP addresses are also included in the cluster summaries, so that peers can randomly choose a helper for comparing their documents with the cluster centroid without going through the cluster holder. Communication between the master and helper cluster holders only takes place for updating the centroids; therefore, load balancing does not impede the system scalability.

4.2 Document Assignment to Clusters

Each peer is responsible for clustering its documents periodically. Clustering of a document consists of two steps: (a) the preselection step, where the most promising clusters for the document are detected, and, (b) the full comparison step, where further clusters are filtered out, and the document is fully compared with the remaining clusters and assigned to the best one.

Preselection step. Consider a peer p, clustering a document d. Let $TopTerms(d)$ denote all terms in d with $TF(t,d) \geq DocTF_{min}(d)$, where $DocTF_{min}(d)$ denotes a frequency threshold for d (we explain how $DocTF_{min}$ is derived in Section 6). For each term t in $TopTerms(d)$, p performs a DHT lookup to find the peer that holds the cluster summaries for t (Fig. 1, Step 1). It then contacts that peer directly to retrieve all summaries published using t as a key (Step 2). To avoid duplicate retrieval of summaries, p executes these requests sequentially, and includes in each request the cluster ids of all summaries already retrieved. All responses are then merged, and a list with the retrieved cluster summaries is constructed. We refer to this list as the *preselection list*, and denote it with $\mathcal{C}_{pre}$.

As we will show later, the summary of the optimal cluster for d is included in $\mathcal{C}_{pre}$ with high probability. This probability can be determined by choosing the value of $DocTF_{min}$ for each document, and the value of $CluTF_{min}$ for each cluster. In Section 6 we explain how peers decide on these values automatically, such that the desired probabilistic guarantees are satisfied.

Full comparison step. Peer p then sends the term vector of d to the candidate cluster holders for performing full document-cluster comparison, and retrieves the comparison results (Fig. 1, Step 3). To avoid sending the document to all cluster holders in $\mathcal{C}_{pre}$ for comparison, p uses the cluster summaries contained in $\mathcal{C}_{pre}$ to filter out the clusters not appropriate for the document at hand. In the following section we describe three different strategies for this task. Finally, p assigns d to the most similar cluster, and notifies the respective cluster holder (Fig. 1, Step 4).

4.3 Filtering Strategies

We propose three different strategies to filter out clusters from $\mathcal{C}_{pre}$: (a) Conservative, (b) Zipf-based, and (c) Poisson-based filtering. All strategies employ the information contained in the retrieved cluster summaries to estimate the cosine similarity between a document and each of the clusters, and to filter out the unpromising clusters based on this estimation. The conservative strategy makes a worst-case similarity estimation, which guarantees that the optimal cluster from the ones in $\mathcal{C}_{pre}$ will be detected. The Zipf-based and Poisson-based strategies filter out the clusters more aggressively, reducing the network cost substantially, at the expense of a small error probability.

Conservative Filtering. This strategy works similar to Fagin's No Random Access (NRA) algorithm for top-k selection [24]: clusters whose maximum possible score is less than the best already known score are removed from the candidate clusters.

In our case the actual score is the cosine similarity between document and cluster centroid, defined as

$$Cos(d,c) = \sum_{t \in d} \frac{TF(t,d) \times TF(t,c)}{|d| \times |c|} \qquad (1)$$

where $|d|$ and $|c|$ are the corresponding L2-Norms, and TF denotes the term frequency of the term in the document/cluster.

Since the actual value of $Cos(d, c)$ is not available, conservative filtering employs the information contained in the cluster summaries to estimate the cosine similarity between the document and each candidate cluster, denoted as $ECos(d, c)$. To ensure that the optimal cluster will not be filtered out, conservative filtering sets the value of $ECos(d, c)$ to the maximum possible value for the actual similarity $Cos(d, c)$.

The similarity estimate relies on term frequency estimates for all document terms not included in the cluster summary. It is computed according to the following constraints: (a) each term not included in the summary of a cluster c has a term frequency of at most $CluTF_{min}(c) - 1$, otherwise it would have been included in the summary, and, (b) the sum of all estimated term frequencies in the cluster cannot exceed the actual cluster length. Precisely, let $t_1, \ldots, t_n$ denote the terms of d, sorted descending on their frequency. For cluster c and term t, we denote the estimated term frequency as $ETF(t, c)$.

Then, cosine similarity is estimated as

$$ECos(d, c) = \sum_{x=1}^{n} \frac{TF(t_x, d) \times ETF(t_x, c)}{|d| \times |c|} \tag{2}$$

Particularly for the conservative strategy, $ETF(t_x, c)$ is computed as follows:

$$ETF(t_x, c) = \begin{cases} TF(t_x, c) & \text{if } t_x \in TopTerms(c) \\ min(CluTF_{min}(c) - 1, & \\ \quad |c|_1 - ST(c) - SE(t_x, c)) & \text{otherwise} \end{cases}$$

where $|c|_1$ denotes the L1-Norm of the cluster terms. $ST(c) = \sum_{t \in TopTerms(c)} TF(t, c)$ is the sum of cluster frequencies for all terms included in $TopTerms(c)$, and $SE(t_x, c) = \sum_{y=1}^{x-1} ETF(t_y, c)$ is the sum of all term frequencies for c estimated up to now.

The filtering process works as follows. Peer p sends the compressed document vector to the first cluster holder in $\mathcal{C}_{pre}$, denoted as $c_{selected}$, and retrieves the actual cosine similarity $Cos(d, c_{selected})$. It then removes from $\mathcal{C}_{pre}$ the summary of $c_{selected}$, as well as the summaries of all clusters with estimated similarity $ECos(d, c) < Cos(d, c_{selected})$. The process is repeated until $\mathcal{C}_{pre}$ is empty. The document is finally assigned to the cluster with the highest cosine similarity. Since the expected cosine similarity is never underestimated, conservative strategy always assigns the document to the optimal cluster.

For the process to complete faster, the list of clusters is sorted based on their lower bound for the cosine similarity. This bound per cluster is computed from the information included in the corresponding cluster summary. We call this the partial cosine similarity:

$$PCos(d, c) = \sum_{\forall t \in TopTerms(c)} \frac{TF(t, d) \times TF(t, c)}{|d| \times |c|} \tag{3}$$

Zipf-based filtering. Although conservative filtering substantially reduces the number of comparisons, it is based on a worst-case estimate for the cosine similarity. A more optimistic estimate, which allows for further reductions, can be derived based on the assumption that the term frequencies in the cluster follow a Zipf distribution. A term frequency estimate for the cosine similarity based on the Zipf distribution is computed as follows:

$$ETF(t_x, c) = \begin{cases} TF(t_x, c) & \text{if } t_x \in TopTerms(c) \\ min(|c|_1/(r(t, c)^s \times H_{DT(c),s}), & \\ \quad |c|_1 - ST(c) - SE(t_x, c)) & \text{otherwise} \end{cases}$$

$DT(c)$ denotes the number of distinct terms in c, and $H_{DT(c),s}$ the generalized harmonic number of order $DT(c)$ of s, i.e., $\sum_{i=1}^{DT(c)} i^{-s}$. Assuming that term frequencies follow a Zipf distribution with exponent s, the expected term frequency of t in c is given by $|c|_1/(r^s \times H_{DT,s})$, where $r(t, c)$ represents the estimated rank of t in c. The ranks of missing terms are estimated as follows: the i-th document term that is not included in $TopTerms(c)$ is assumed to exist in the cluster centroid, with rank $r = |TopTerms(c)| + i$.

Apart from the definition of $ETF(t, c)$, Zipf-based filtering is identical to conservative filtering.

Poisson-based filtering. This strategy follows a different approach for pruning the candidate clusters, which allows for probabilistic guarantees, and for a tradeoff between clustering quality and network cost. The strategy is based on the assumption that the score of each term t across all documents follows a Poisson distribution, where the score of a term represents the term frequency normalized on the document's length, i.e., $TF(t)/|d|$. Poisson distribution is often used, e.g., [25], as it provides a reasonable fit for the term scores, and it has some convenient properties, which will be discussed in Section 6.5.

Note that the Poisson distribution model does not contradict the well accepted Zipf model for term frequencies within a single cluster, since the two distributions model two discrete concepts.

Consider peer p, which has already retrieved $\mathcal{C}_{pre}$ for document d. First, p computes the partial cosine similarity $PCos(d, c)$ for each cluster $c \in \mathcal{C}_{pre}$. For the cluster c_{maxp} with the best partial cosine similarity, it sends d to the cluster holder and computes the full cosine similarity $Cos(d, c_{\text{maxp}})$. Then, p estimates the remaining cosine similarity $RCos(d, c)$ for all other clusters $c \in \mathcal{C}_{pre}$, i.e., the difference between the actual cosine similarity $Cos(d, c)$ and the partial cosine similarity $PCos(d, c)$. Instead of estimating $RCos(d, c)$ directly, which would be inefficient, the peer computes the Probability Density Function (PDF) describing $RCos(d, c)$, which allows computing the probability that $RCos(d, c)$ is above a threshold. In particular, as we show in Section 6.5, $RCos(d, c)$ follows a Poisson distribution with parameter $\lambda = \sum_{t \in d \setminus TopTerms(c)} TF(t, d) \times AvgSc(t, \mathcal{C}_{pre})/|d|$, where

$AvgSc(t, \mathcal{C}_{pre})$ is defined as:

$$AvgSc(t, \mathcal{C}_{pre}) = \sum_{\forall c \in \mathcal{C}_{pre}} \begin{cases} TF(t,c)/|c| & \text{if } t \in TopTerms(c) \\ \frac{CluTF_{min}(c)-1}{|c|} & \text{otherwise} \end{cases}$$

Using this distribution, peer p can compute the probability that $RCos(d,c)$ exceeds any value $Z \in [0,1]$. This probability is (Section 6.5):

$$Pr[RCos(d,c) \geq Z] \leq \sum_{i=\left\lfloor \frac{(1-Z)m}{MaxSc} \right\rfloor}^{m} \exp(-\lambda) \times \frac{\lambda^i}{i!}$$

with $MaxSc = \frac{CluTF_{min}(c)-1}{|c|}$.

Following, p discards all clusters $c \in \mathcal{C}_{pre}$ for which the remaining cosine similarity is less than the difference $diff = Cos(d, c_{\text{maxp}}) - PCos(d,c)$ with high probability. In particular, for a required correctness probability Pr_{fcs}, clusters are discarded if $Pr[RCos(d,c) < diff\ ] \geq Pr_{fcs}$. The optimal cluster for the document is therefore kept with a probability higher than Pr_{fcs}. Peer p sends d to the remaining candidate clusters for cosine similarity computation, and assigns the document to the cluster with the highest cosine similarity.

Compared to conservative filtering, Poisson-based filtering discards clusters more aggressively, and thus reduces the required cosine similarities substantially. Its main benefits compared to Zipf is that it allows for controlling the tradeoff between cost and clustering quality, and provides probabilistic guarantees. Therefore, it can reduce network usage significantly, while still exhibiting a low and predictable error probability.

4.4 Further aspects

In this section we describe how PCP2P is initialized, and how it addresses churn.

Initialization. There are different possible ways of initialization. The easiest one assumes that a peer from the network starts the algorithm by selecting k peers randomly to act as cluster holders. These cluster holders choose one of their documents as initial centroid, and publish the cluster summary to the DHT. Then, the initiating peer uses broadcasting over DHT to notify all participating peers to start clustering. In a continuously clustering network, e.g., [3], initialization incurs only at the begining, and the clusters are maintained for the lifetime of the network.

Churn. Cluster indexing and document assignment are repeated periodically to compensate churn. No synchronization is required between the peers. Cluster holders rebuild the cluster centroids in regular intervals, i.e., every m minutes. They include in the centroids only documents that were assigned to them during the last interval. Therefore, if a peer gets disconnected, its documents are automatically removed from the clusters within the next interval. Peers re-cluster their documents also every m minutes. Hence, each document belongs to exactly one cluster at any time, and cluster centroids are at most m minutes stale. In case a cluster holder is disconnected unexpectedly, one of its helper cluster holders replaces it, without information loss. If there is no helper cluster holder, the next peer detecting the absense of the cluster holder, i.e., a peer that has already clustered a document to it, becomes the cluster holder, re-seeds the cluster with one of its documents, and updates the cluster inverted index accordingly.

5 Cost Analysis

The network cost is composed of: (a) indexing the cluster summaries, (b) the preselection step for each document, and, (c) the full comparison step.

Indexing of the summaries requires performing a DHT lookup for each term in $TopTerms(c)$, and publishing the summaries. The corresponding indexing cost per cluster is $Cost_{ind} = O(|TopTerms(c)| \times \log(n))$, both with respect to transfer volume and number of messages. This cost is easily manageable by the cluster holders, since $|TopTerms(c)|$ is usually small, e.g., an average of 66 in the experiments with 100 clusters presented in Section 7.

The preselection step is executed for each document d, and causes $|TopTerms(d)|$ DHT lookups, which translate to a cost of $Cost_{pre} = O(|TopTerms(d)| \times \log(n))$ messages. With respect to transfer volume, preselection requires $O(|TopTerms(d)| \times \log(n))$ for looking up the document $TopTerms$, and $O(\sum_{c_i \in \mathcal{C}_{pre}} |TopTerms(c_i)|)$ for retrieving the summaries, where $\mathcal{C}_{pre}$ denotes the clusters retrieved at the preselection step. The cardinality of $TopTerms(d)$ is also small. For example, for correctness probability $Pr_{pre} = 0.8$, the average cardinality was 6.6, whereas for $Pr_{pre} = 0.95$, which yields practically the same quality as the exact counterpart, it was only 8.9.

Finally, the full comparison step requires sending the document to all remaining candidate clusters. Let $\mathcal{C}_{fcs}$ denote the set of these clusters for d. The full comparison step requires $Cost_{fcs} = O(|\mathcal{C}_{fcs}|)$ messages. With respect to transfer volume, this step incurs a cost $TVCost_{fcs} = O(|\mathcal{C}_{fcs}| \times |d|)$.

The cost of indexing the cluster summaries is negligible, because the number of clusters is small, and their summaries are very compact. The dominating cost, both with respect to number of messages and transfer volume, is the one incurred for assigning documents to clusters, namely $Cost_{pre} + Cost_{fcs}$. Per document, this cost has the following properties: (a) it scales logarithmically with the number of peers, because DHT access cost grows logarithmically, and (b) it is independent of the size of the document collection. It also depends on $|\mathcal{C}_{fcs}|$, which is by definition at most equal to the number of clusters. Therefore, the clustering cost per document scales, in the worst case, linearly with the number of clusters. However, our experimental evaluation shows that the number of clusters in $\mathcal{C}_{fcs}$ is very small, and independent of the total number of clusters k. Therefore, in practice, this cost grows at a much slower rate than linear. This means that PCP2P scales to large networks, and with large numbers of documents and clusters.

6 Probabilistic Analysis

For Section 4, we have assumed that the optimal values for $CluTF_{min}(c)$ and $DocTF_{min}(d)$ are known. We now describe how each peer computes these values dynamically to satisfy the desired system-wide clustering quality requirements, expressed using probabilistic guarantees.

The analysis uses a probabilistic document generation model [26], [27]. Briefly, the model assumes that each document belongs to a topic $\mathcal{T}$, and each topic $\mathcal{T}_i$ is described by a term probability distribution ϕ_i (a language model). Composite topics are also possible. A document of length l belonging to $\mathcal{T}_i$ is created by randomly selecting l terms with replacement from ϕ_i. The probability of selecting a term t is given by the topic distribution ϕ_i.

6.1 Notations and Overview

The user initiating the algorithm chooses the desired system quality, in terms of the required probability $Pr_{correct}$ that each document is assigned to the optimal cluster. The other system parameters are determined automatically from PCP2P. Specifically, $Pr_{correct}$ determines the following three system parameters: (a) Pr_{ind}, which represents the probability that the summary of each cluster c_i is indexed in all $Top(\alpha, \phi_i)$, (b) Pr_{pre}, expressing the probability that the preselection step for a document retrieves the optimal cluster for this document, and, (c) Pr_{fcs}, which is the probability that the full comparison step retrieves the optimal cluster. The desired values for these probabilities are common to all peers, and they are selected automatically such that $Pr_{pre} \times Pr_{fcs} \geq Pr_{correct}$.

The purpose of the probabilistic analysis is to enable PCP2P to automatically set Pr_{ind}, Pr_{pre}, and Pr_{fcs} for the given $Pr_{correct}$, and to configure each step accordingly such that these probabilities are satisfied. First, during the cluster indexing step, the cluster holder of each cluster c_i computes the value of $CluTF_{min}(c_i)$ so that the cluster summary is indexed in all terms in $Top(\alpha, \phi_i)$ with the predefined probability Pr_{ind}. In Section 6.3 we show how $CluTF_{min}$ is computed per cluster such that Pr_{ind} is satisfied. A document is clustered correctly when both the preselection and full comparison step return the best cluster. The two steps succeed with probabilities Pr_{pre} and Pr_{fcs}, respectively. At the preselection step, the peer holding document d selects $DocTF_{min}(d)$ so that the optimal cluster for d is retrieved with probability Pr_{pre}. We explain how $DocTF_{min}$ is computed in Section 6.4. Finally, the full comparison step takes place. In this step, p decides which candidate clusters to filter out as non-promising such that the best cluster is found with probability Pr_{fcs}. We present the corresponding probabilistic analysis for this step in Section 6.5.

k	Number of clusters
ϕ_i	Term distribution of cluster c_i
$Top(\alpha, \phi_i)$	The set of α terms with highest probability in ϕ_i
$t_x[\phi_i]$	The x term of distribution ϕ_i, where terms are sorted by descending probability
$Pr_{correct}$	Desired correctness probability
Pr_{ind}	Probability that the cluster summary is indexed in all terms in $Top(\alpha, \phi_i)$
Pr_{pre}	Probability that the preselection step succeeds
Pr_{fcs}	Probability that the full comparison step succeeds

TABLE 1
Notations

The following notations are used throughout the analysis. With $\mathcal{C}_{sol} := \{c_1, \ldots, c_k\}$ we denote a snapshot of clusters on an ongoing clustering. Each cluster $c_i \in \mathcal{C}_{sol}$ follows the language model ϕ_i. We use $t_1[\phi_i], \ldots, t_n[\phi_i]$ to denote the terms of ϕ_i sorted by descending probabilities. With $Top(\alpha, \phi_i)$ we denote the set of α terms with highest probability in ϕ_i, i.e., $t_1[\phi_i]$, ..., $t_\alpha[\phi_i]$. Table 1 summarizes the notations.

6.2 Preliminaries

We first derive probabilistic bounds for the estimation of the term frequency of any term t in a document or a cluster, given the term frequency distribution of the document or the cluster. These bounds are required for the subsequent analysis.

Theorem 1 computes the probability that a term with a given expected frequency has an actual frequency in the document higher than a given value.

Theorem 1: Given a document d which follows language model ϕ_i. The expected frequency of term t in d according to ϕ_i is denoted with $\hat{TF}(t,d)$. For term t with $\hat{TF}(t,d) > DocTF_{min}$, the probability of the actual term frequency $TF(t,d)$ exceeding $DocTF_{min}$ is at least $1 - \exp(-\hat{TF}(t,d) \times (1 - DocTF_{min}/\hat{TF}(t,d))^2/2)$. Furthermore, for a probability Pr_{min} the frequency of term t in d is at least $\lfloor \hat{TF}(t,d) - \sqrt{2 \times \hat{TF}(t,d) \times \log(\frac{1}{1-Pr_{min}})} \rfloor$.

Proof: The proof uses the simplified Chernoff bound for the lower tail ([28], p. 72), which states that for any $\delta > 0$, the probability of the sum of N independent Poisson trials to be less than $(1-\delta) \times \mu$ is less than $\exp(-\mu \times \delta^2/2)$. Symbol μ denotes the expected value for the sum.

We apply the bound to find $Pr[TF(t, c_i) < DocTF_{min}]$ as follows. For a document d of length l and for a term t, we define binary random variables $Z_1, \ldots, Z_l$, where Z_i denotes the event that the i^{th} term of d is t. As in most language models, we assume that terms are independent. The expected number of occurrences of t is denoted by $\hat{TF}(t,d) = \phi_i[t] \times l$. Then, by Chernoff bounds (lower tail):

$$Pr[TF(t,d) \geq DocTF_{min}] \geq$$
$$1 - \exp(-\hat{TF}(t,d) \times (1 - DocTF_{min}/\hat{TF}(t,d))^2/2)$$

For any probability Pr_{min}, the minimum value of $DocTF_{min}$ satisfying $Pr[TF(t,d) \geq DocTF_{min}] \geq Pr_{min}$ is:

$$DocTF_{min} = \hat{TF}(t,d) - \sqrt{2 \times \hat{TF}(t,d) \times \log(\frac{1}{1 - Pr_{min}})} \quad (4)$$

Since $DocTF_{min}$ is a natural number, we take the floor of the RHS of expression 4 as its value. □

Similarly, Theorem 2 computes the probability of a term with a given expected term frequency to have a frequency in the cluster exceeding a given value.

Theorem 2: Given a cluster c_i which follows language model ϕ_i. The expected term frequency of term t in c_i according to ϕ_i is denoted with $\hat{TF}(t,c_i)$. For term t with $\hat{TF}(t,c_i) > CluTF_{min}$, the probability of the actual term frequency $TF(t,c_i)$ to exceed $CluTF_{min}$ is at least $1 - \exp(-\hat{TF}(t,c_i) \times (1 - CluTF_{min}/\hat{TF}(t,c_i))^2/2)$. Furthermore, for a probability Pr_{min} the term frequency of term t in c_i is at least $\lfloor \hat{TF}(t,c_i) - \sqrt{2 \times \hat{TF}(t,c_i) \times \log(\frac{1}{1-Pr_{min}})} \rfloor$.

Proof: We represent clusters as concatenations of their documents. Since we assume that all documents in cluster c_i follow the same language model ϕ_i, the cluster also follows ϕ_i. The equations follow directly from Chernoff inequality, similar to Theorem 1. □

6.3 Indexing of Cluster Summaries

We now show how cluster holders derive the value of $CluTF_{min}$ so that cluster indexing satisfies the required probabilistic guarantees. Assuming that term frequencies follow a Zipf distribution (validated, for example, in [29]) the expected frequency of the α'th most probable term of ϕ_i in a given cluster c_i is $\hat{TF}(t_\alpha[\phi_i],c_i) \approx |c_i|_1/(\alpha^s \times H_{m,s})$ where $|c_i|_1$ is the L1-norm of c_i, m is the number of distinct terms in c_i, and s is the Zipf exponent of ϕ_i (typically around 1).

Cluster holders are required to publish the cluster summary using each of the terms in $Top(\alpha,\phi_i)$ as a key, with a probability at least equal to a system-wide value Pr_{ind}. They find the respective lower bound for the term frequency of $t_\alpha[\phi_i]$ using Theorem 2:

$$CluTF_{min}(t_\alpha[\phi_i],c_i) = \Big\lfloor \hat{TF}(t_\alpha[\phi_i],c_i) - \sqrt{2 \times \hat{TF}(t_\alpha[\phi_i],c_i) \times \log(\frac{1}{1 - Pr_{ind}})} \Big\rfloor \quad (5)$$

All terms $t_j[\phi_i]$ for $j \leq \alpha$ will have a term frequency in the cluster at least equal to $CluTF_{min}$ with a minimum probability Pr_{ind}. Thus, cluster holders will detect each term in $Top(\alpha,\phi_i)$ with a minimum probability Pr_{ind}.

In Section 6.6, we describe how the system-wide values for Pr_{ind} and α are set, and how the Zipf exponent is estimated.

6.4 Preselection Step

A peer p needs to set the value of $DocTF_{min}$ per document, which will guarantee that the optimal cluster for the document is detected in the preselection step with a probability Pr_{pre}. We now describe how this value is determined.

According to the cluster indexing activity, the cluster holder of c_i includes in the cluster summary each term in $Top(\alpha,\phi_i)$ with a probability of at least Pr_{ind}. Consider a document d which follows ϕ_i. For the preselection step to succeed, peer p needs to correctly identify from d at least one of the terms from $Top(\alpha,\phi_i)$ that were also included in the cluster summary. In this case, the optimal cluster will be included in the list of clusters collected in the preselection step.

For a given value of $DocTF_{min}$ and for all $j \leq \alpha$, term $t_j[\phi_i]$ is correctly identified by peer p when the term frequency of $t_j[\phi_i]$ in d is at least equal to $DocTF_{min}$. The probability that this happens is denoted by $Pr[TF(t,d) \geq DocTF_{min}]$, and we can compute it with Theorem 1. Therefore, the probability that the preselection step succeeds is

$$Pr_{pre}(DocTF_{min}) \geq 1 - \prod_{j=1}^{\alpha}(1 - Pr[TF(t_j,d) \geq DocTF_{min}] \times Pr_{ind}) \quad (6)$$

Using Eqn. 6, p derives the maximum value of $DocTF_{min}$ satisfying $Pr_{pre}(DocTF_{min}) \geq Pr_{pre}$. Then, it determines the document's terms with frequency at least equal to $DocTF_{min}$, and executes preselection.

6.5 Full comparison step

The preselection step for d returns a set of candidate clusters $\mathcal{C}_{pre}$. From this list, peer p selects the clusters that will be fully compared with d, using one of the filtering strategies introduced in Section 4.3. Conservative filtering always returns the best cluster c_{opt} from $\mathcal{C}_{pre}$, and thus it always assigns d to the best candidate cluster. For the Poisson-based strategy, we now obtain probabilistic guarantees that d will be assigned to c_{opt}. The Zipf-based strategy does not provide probabilistic guarantees for the full comparison step.

The analysis assumes that for each term $t \in d$, the Probability Density Function (PDF) of the term scores of t in all clusters follows the Poisson distribution, where term scores are defined as the term frequencies normalized on the cluster length. In other words, $\{TF(t,c_1)/|c_1|,\ TF(t,c_2)/|c_2|,\ \ldots,\ TF(t,c_k)/|c_k|\}$ follow a Poisson distribution. Poisson distribution, besides being a good fit for the term scores, has two useful properties: (i) Poisson fitting is efficient, as it only requires finding the average value, and, (ii) the convolution of k Poisson distributions with $\lambda_1,\ldots,\lambda_k$ is also a Poisson with $\lambda = \sum_{i=1}^{k}\lambda_i$. This second property is of particular importance, because it allows us to efficiently compute the PDF of $RCos(d,c)$ by convoluting the Poisson distributions corresponding to the normalized term frequencies of all document terms not found in $TopTerms(c)$.

To estimate the PDF of a term t, peer p finds the $AvgSc(t, \mathcal{C}_{pre})$ value:

$$AvgSc(t, \mathcal{C}_{pre}) = \sum_{c \in \mathcal{C}_{pre}} \begin{cases} TF(t,c)/|c| & \text{if } t \in \mathit{TopTerms}(c) \\ \frac{CluTF_{min}(c)-1}{|c|} & \text{otherwise} \end{cases}$$

Since Poisson is a discrete distribution, p discretizes the score range to m equidistant values, $v_0, v_1, \ldots, v_{m-1}$, with $v_i = 1 - i \times MaxSc(t)/m$. $MaxSc(t)$ refers to the upper bound score of t, which is $\frac{CluTF_{min}(c)-1}{|c|}$ when $t \notin TopTerms(c)$. The value of m represents the resolution of the discretization.

The PDF of $RCos(d, c)$ for $c \in \mathcal{C}_{pre}$ is estimated in a similar fashion. First, p finds all document terms not included in $TopTerms(c)$. These terms are the terms that may contribute to $RCos(d, c)$. The contribution of any of these terms to $RCos(d, c)$ is $\frac{TF(t,d)}{|d|} \times \frac{TF(t,c)}{|c|}$. Since the PDF of $\frac{TF(t,c)}{|c|}$ is a Poisson, the PDF of the term's contribution is also Poisson, with $\lambda_t = \frac{TF(t,d)}{|d|} \times AvgSc(t)$. The PDF of $RCos(d, c)$ is the convolution of the PDFs for all terms $t \in d \backslash TopTerms(c)$. We need to discretize the scores to m equidistant values, using $1 - PCos(d, c)$ as a maximum. The resulting PDF is:

$$Pr[RCos(d, c) \geq v_j] \leq \sum_{i=j}^{m} e^{-\lambda} \times \frac{\lambda^i}{i!} \qquad (7)$$

where $\lambda = \sum_{t \in d \backslash TopTerms(c)} \lambda_t$.

6.6 Algorithm Configuration

In the previous sections we have described the algorithm without focusing on how the user sets the parameters, and what implications these parameters have. As in standard K-Means, a user can freely select the number of clusters k. In addition, she can configure the acceptable correctness probability $Pr_{correct}$ for her application. All other parameters are derived automatically, as follows.

First, a few sampled documents are collected from the network and are used to estimate the Zipf distribution skew. The algorithm computes the remaining values as follows. By default, α is set to 10 and Pr_{ind} to 0.9, since the algorithm adapts to these values to satisfy the probabilistic guarantees. The values of Pr_{pre} and Pr_{fcs} are set as follows. In conservative filtering, the full comparison step is always correct, therefore $Pr_{pre} = Pr_{correct}$ satisfies the probabilistic guarantees. For Zipf-based filtering we only provide probabilistic guarantees for the preselection step, and these are achieved by setting $Pr_{pre} = Pr_{correct}$. For Poisson-based filtering, $Pr_{pre} = Pr_{fcs} = \sqrt{Pr_{correct}}$ clearly satisfies the probabilistic guarantees. The algorithm then derives $DocTF_{min}$ and $CluTF_{min}$, as explained earlier, to satisfy the desired probabilistic guarantees.

The previous combination of α and the probability values is not the only satisfying combination. All combinations satisfying the constraint $Pr_{fcs} \times Pr_{pre} \geq Pr_{correct}$ satisfy the probabilistic guarantees; the optimal combination of α, Pr_{ind}, Pr_{pre}, and Pr_{fcs} is the one that minimizes the cost. Preliminary experiments show that such an optimization could reduce the cost further by around 10%. Part of our future work is to efficiently identify the configuration that minimizes the cost.

7 Experimental Evaluation

PCP2P was experimentally evaluated with up to 1 million peers and 1 million documents. The experiments had the following objectives: (a) to evaluate the quality and efficiency of PCP2P with different network configurations and multiple datasets, (b) to compare the algorithm with the state-of-the-art algorithm for P2P clustering, as well as with the standard centralized counterpart, (c) to examine the scalability of PCP2P, with respect to network size, number of documents and number of clusters, and, (d) to assess the effects of load balancing. We now describe the evaluation methodology, datasets, and measures.

Evaluation measures. Efficiency was evaluated using the following criteria: (a) number of messages, (b) transfer volume, and, (c) number of document-cluster comparisons per clustering iteration.

Clustering quality was evaluated using the standard quality measures of *entropy*, *purity*, and *normalized mutual information* (*NMI*) [30], [13], which compare the clustering results against human-generated document classifications. Furthermore, since PCP2P approximates standard K-Means, as an approximation quality metric we measured the number of document assignments differing from the standard K-Means clustering solution. Precisely, if $\mathcal{C}_{sol'}$ denotes the standard K-Means clustering solution, then the approximation quality is:

$$Ap(\mathcal{C}_{sol}, \mathcal{C}_{sol'}) = \frac{1}{N} \sum_i \max \arg_j |c_i \cap c'_j|$$

Datasets and Methodology. We simulated networks of up to one million documents and peers. The experiments were conducted on three datasets, a synthetic dataset generated according to the well-accepted Probabilistic Topic Model, and two real-world datasets, the REUTERS Corpus Volume 1 (RCV1), and MEDLINE, the largest standard text collections which include a human-generated classification. RCV1 includes more than 800,000 categorized newswire articles, pre-processed with stopword filtering and stemming. MEDLINE on the other hand contains information for more than 11 million citations with abstracts, categorized according to the MeSH vocabulary. To be able to apply the standard quality measures, we have used a subset of the two collections, taking all articles and abstracts that belonged to exactly one class. This resulted in approximately 140 thousands articles for RCV1 categorized in 53 classes and 130 thousands abstracts for MEDLINE, belonging to 40 classes. To systematically examine the effect of the collection's characteristics on the algorithm, and to

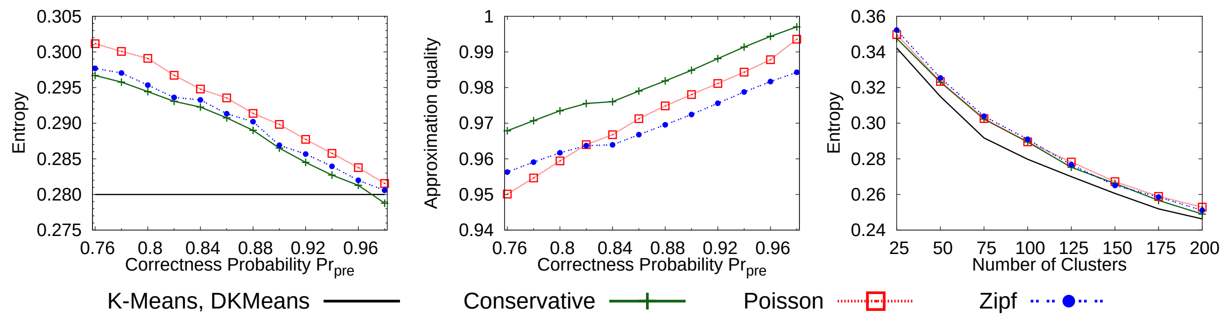


Fig. 2. Quality: a. Entropy, b. Approximation quality, c. Entropy for different number of clusters

evaluate it with a significantly larger dataset, we have also used synthetic document collections (SYNTH) with a size of 1.4 million documents each. These collections were created according to the Probabilistic Topic Model proposed in [27], from 200 composite language models, with different term distribution skews.

Unless otherwise noted, peer collections were created by partitioning the datasets uniformly to all peers. Churn was simulated by selecting a percentage of peers at each iteration, and replacing them with new peers, carrying new documents. The number of documents at each iteration, after churn, was 100,000 for the real-world datasets, and 1 million for the SYNTH dataset. Results are presented for 20% churn; the outcome was similar for other churn factors.

PCP2P was compared with two other distributed clustering algorithms: (a) LSP2P [6], the state-of-the-art in P2P clustering, and, (b) DKMeans, a distribution of K-Means over P2P (see Section 2). Note that DKMeans produces exactly the same results as K-Means, and therefore constitutes a good comparison baseline. We did not include other P2P clustering algorithms in the comparison, e.g., [7], [8], since these do not scale to large networks, and to high-dimensional data like text, as discussed in Section 3.

We present average results after 40 repetitions of each experiment. Owing to the continuous churn, the algorithms never finish, i.e., the cluster centroids do not converge. This is also the case for real-world setups, where network contents change continuously. However, the quality results of all algorithms stabilize after about 15 iterations. Therefore, the algorithms are let to run for 20 iterations, before we measure their network cost and quality per iteration.

7.1 Quality

The quality of PCP2P is influenced by the correctness probability, the number of clusters, and the dataset characteristics (number of documents and term distribution skew). The network size and the distribution of documents to peers do not have any effect on the quality, since each document is clustered individually. Therefore, the results reported in this section are also representative of networks of different sizes, as well as of different document distribution models.

Correctness probability. To examine the effect of the correctness probability Pr_{pre}, we executed PCP2P on a network of 10,000 peers, configured with Pr_{pre} in the range of 0.76 to 0.98. Fig. 2a. corresponds to the entropy measure for the RCV1 collection, whereas Fig. 2b. plots the approximation quality. Table 2 (Part I.), summarizes the results for purity and NMI. As expected, the quality of all PCP2P variants increases with the correctness probability. Conservative PCP2P yields the best quality, since it never filters out potential optimal clusters. The Poisson and Zipf-based variants yield comparable quality. For probabilities higher than 0.9, all variants closely approximate DKMeans. Also note that the resulting approximation quality is always higher than the one expected by the probabilistic guarantees, since the guarantees define upper bounds for the number of errors. The experimental results with MEDLINE and SYNTH had similar outcomes, and are summarized in Table 2 (Part IV.).

Number of clusters. As shown in Fig. 2c., the number of clusters also affects the quality of all algorithms. This is expected, considering that the computation of the quality measures depends on the number of clusters. However, the quality of PCP2P is always very close to the maximum quality delivered by DKMeans, even for the configuration with 200 clusters. Summarizing, the quality of PCP2P does not decrease with an increase of the number of clusters.

Dataset characteristics. To verify the applicability of PCP2P for different text corpora, we have used the SYNTH collection, which enabled us to manipulate its characteristics. All algorithms were executed on a network with a fixed number of peers, progressively increasing the number of documents up to 1 million. Table 2 (Part II.) presents key results of this experiment. Our first observation is that collection size does not have a significant effect on clustering quality; the minor fluctuations (between 0.164 and 0.168 for entropy) are due to the different document collections clustered at each configuration, and not due to an algorithmic factor. As such, the PCP2P quality always remains very close to the quality of DKMeans, even for the largest collection with one million documents.

The term frequency distribution skew of the dataset is also an important factor for the quality of PCP2P, since the algorithm relies on the fact that term frequencies follow the Zipf distribution. Although it is widely accepted that document collections follow the Zipf distribution, different document collections exhibit different distribution skews [29]. For example, the RCV1 collection has a skew of 0.55 and MEDLINE has a skew of 0.59. Other values, typically around 1.0, are also frequently reported in the literature, e.g., in [29], [31]. To evaluate the influence of the skew on PCP2P, we have used the SYNTH collections (subset of 100,000 documents) generated with different Zipf skew factors, between 0.5 and 1.4. The results, summarized in Table 2 (Part IV.), confirm that PCP2P adapts to the collection skew and delivers high-quality clustering in all cases.

Summary. A large set of experiments with different collections of up to 1 million documents and peers confirmed that all PCP2P variants closely approximate the solution of K-Means, as predicted by the theoretical analysis. The proposed algorithms are not influenced by the network size, the number of documents, and the distribution of documents to peers, and successfully adapt to the collection characteristics to deliver high-quality solutions.

7.2 Efficiency and scalability

We now investigate the effect on efficiency of the correctness probability, the network and collection characteristics, as well as the number of clusters. Furthermore, we examine the additional cost imposed by load balancing.

Correctness probability. Figures 3a. and b. show the number of messages and transfer volume per clustering iteration, in correlation to Pr_{pre}. The results correspond to the *total cost* of a network of 100,000 peers, with the RCV1 collection. All PCP2P variants generate an order of magnitude less messages than DKMeans, with Poisson and Zipf-based PCP2P being the most efficient. Concerning transfer volume, conservative filtering requires between 17% and 27% of the transfer volume of DKMeans, whereas Zipf-based and Poisson-based filtering require between 15% and 22%. Note that the Poisson variant already provides 90% correctness probability with 6% messages and 17% of the transfer volume of DKMeans. The number of document-cluster comparisons, which translates to computational cost for the cluster holders, is also substantially reduced compared to the baseline: the conservative strategy requires at most 12% of the DKMeans comparisons, whereas the Poisson and Zipf-based variants require below 1%, even for the highest investigated probabilistic guarantees, which deliver practically the same quality as DKMeans.

Interestingly, the influence of increasing the correctness probability on network cost is more noticeable for high Pr_{pre} values. For example, we see only minor cost fluctuations for Pr_{pre} in the range of 0.76 to 0.88,

Setup		Quality			Cost	
	Alg.	Entr.	NMI	Pur.	Msgs $\times 10^6$	Tr. Vol. (Gb)
I. Vary probability guarantees. RCV1, 100 clusters, 10,000 peers Pr_{pre}						
N/A	KMeans	0.279	0.524	0.689	186	13.2
0.8	Cons.	0.294	0.506	0.682	12.2	2.4
	Poisson	0.299	0.503	0.679	10.8	1.98
	Zipf	0.295	0.506	0.682	10.8	1.98
0.9	Cons.	0.286	0.516	0.689	13.7	2.7
	Poisson	0.289	0.513	0.687	12.1	2.24
	Zipf	0.286	0.516	0.689	12.1	2.23
0.98	Cons.	0.278	0.525	0.694	23.9	3.66
	Poisson	0.281	0.523	0.692	21.8	3.03
	Zipf	0.280	0.524	0.693	21.7	3
II. Vary #documents. SYNTH, exp.=1.0, Pr_{pre}=0.9, 100,000 peers Documents						
100000	KMeans	0.165	0.906	0.490	186.2	10.17
	Cons.	0.165	0.906	0.490	3.68	0.22
	Poisson	0.165	0.905	0.489	3.67	0.22
	Zipf	0.165	0.906	0.490	3.67	0.22
500000	KMeans	0.165	0.905	0.484	266.6	21.17
	Cons.	0.166	0.905	0.483	17.91	1.08
	Poisson	0.167	0.904	0.483	17.82	1.08
	Zipf	0.165	0.905	0.485	17.85	1.08
1000000	KMeans	0.164	0.907	0.484	367.1	34.91
	Cons.	0.163	0.907	0.485	34.78	2.08
	Poisson	0.164	0.906	0.485	34.61	2.07
	Zipf	0.165	0.905	0.482	34.67	2.07
III. Vary #peers. SYNTH, Pr_{pre}=0.9, 100 clusters Peers						
100000	KMeans	0.164	0.907	0.481	367.1	34.91
	Cons.	0.164	0.907	0.485	34.78	2.08
	Poisson	0.164	0.906	0.485	34.61	2.07
	Zipf	0.165	0.905	0.482	34.67	2.07
500000	KMeans	0.164	0.907	0.481	1147.58	69.80
	Cons.	0.164	0.907	0.485	41.38	2.42
	Poisson	0.164	0.906	0.485	41.27	2.44
	Zipf	0.165	0.905	0.482	41.29	2.44
1000000	KMeans	0.164	0.907	0.481	2194.16	116.59
	Cons.	0.164	0.907	0.485	43.78	2.57
	Poisson	0.164	0.906	0.485	43.67	2.56
	Zipf	0.165	0.905	0.482	43.72	2.56
IV. Vary collection characteristics. Pr_{pre}=0.9, 100,000 peers Collection (100,000 documents)						
MED LINE	KMeans	0.557	0.222	0.373	186.2	12.58
	Cons.	0.566	0.212	0.367	15.98	2.64
	Poisson	0.570	0.208	0.362	13.11	1.87
	Zipf	0.568	0.210	0.365	13.09	1.89
SYNTH exp.=0.5	KMeans	0.170	0.899	0.491	186.2	11.81
	Cons.	0.170	0.899	0.493	15.93	1.09
	Poisson	0.169	0.899	0.497	15.42	0.99
	Zipf	0.169	0.899	0.494	15.44	1
SYNTH exp.=1.0	KMeans	0.165	0.906	0.490	186.2	10.17
	Cons.	0.165	0.906	0.490	3.68	0.22
	Poisson	0.165	0.905	0.489	3.67	0.22
	Zipf	0.164	0.906	0.490	3.67	0.22
SYNTH exp.=1.4	KMeans	0.164	0.907	0.487	186.2	9.47
	Cons.	0.163	0.907	0.489	3.55	0.21
	Poisson	0.163	0.907	0.489	3.55	0.21
	Zipf	0.163	0.908	0.489	3.55	0.21

TABLE 2
Detailed experimental results

whereas the cost increase in the range of 0.9 to 0.98 becomes more significant. This behavior is also observed on MEDLINE and SYNTH, and is consistent with all cost measures. This is a frequently observed behavior of algorithms offering probabilistic guarantees. However, even for very high probabilistic guarantees, PCP2P yields significant network savings compared to DKMeans, inducing manageable network cost. Indicatively, the cost for clustering RCV1 with $Pr_{pre} = 0.98$ in 10,000 peers is below 385 Kbytes per peer, per reclustering period (Table 2, Part I.). According to recent Google statistics [32], this is comparable to the average transfer volume required to retrieve a single web-page, with its associated resources (320 Kbytes).

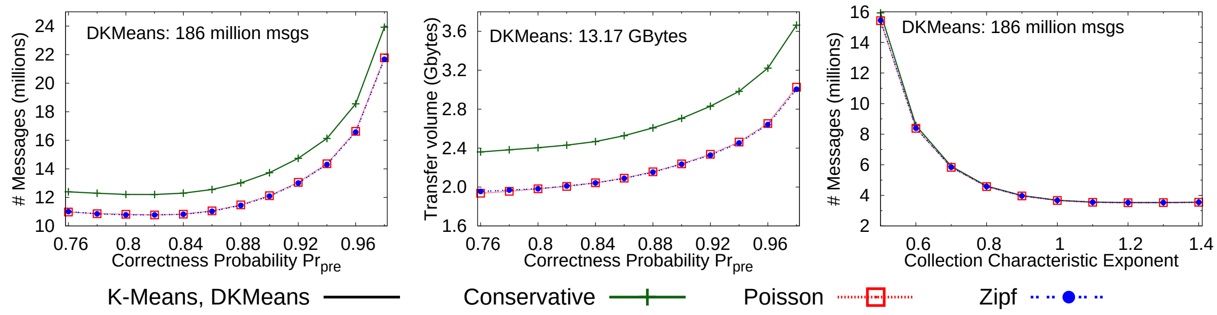


Fig. 3. Efficiency: a. # messages, b. Transfer volume, c. varying skew

Network size. For evaluating the effect of the number of peers on PCP2P, we distributed the datasets to networks of different sizes, and measured the cost for conducting the clustering. In Figure 4a. and Table 2 (Part III.) we present the results for the SYNTH collection, since this was the largest and allowed us to simulate networks of up to one million peers. We see that the cost for PCP2P increases only logarithmically with network size, while cost for DKMeans increases linearly. This behavior is expected, and in accordance with the cost analysis; the only factor changing with network size for PCP2P is the DHT access cost, which grows only logarithmically. On the other hand, DKMeans cost increases linearly since each peer needs to communicate with all cluster holders, independent of the number of documents it carries. Similar results were observed on the other two collections.

Number of clusters. To examine the effect of the number of clusters to the efficiency of PCP2P, we repeated the clustering of the three collections on a network of 100,000 peers with up to 200 clusters. Figure 4b. shows the number of messages in correlation to the number of clusters for the RCV1 collection. Clearly, all PCP2P variants scale favorably with the number of clusters. For example, for 25 clusters, the conservative variant induces 12 million messages, whereas for 200 clusters it causes only 3 million additional messages. The Poisson-based variant causes between 11.7 to 12.2 million messages in the same cluster range, whereas Zipf-based PCP2P causes 11.6 to 12.2 million messages. The same scale-up properties are observed with respect to the transfer volume and number of comparisons, not presented due to space limitations. This essentially means that the network cost of PCP2P is only slightly affected by the number of clusters, making the algorithm scalable for a wide range of setups and requirements.

Dataset characteristics. Similar to the quality experiments, the SYNTH collection was used to evaluate the effects of the collection characteristics to the efficiency of PCP2P. Table 2 (Part II.) includes the results corresponding to different dataset sizes, for a fixed network size of 100,000 peers. As predicted from the cost analysis, PCP2P scales linearly with the collection size with respect to all cost measures.

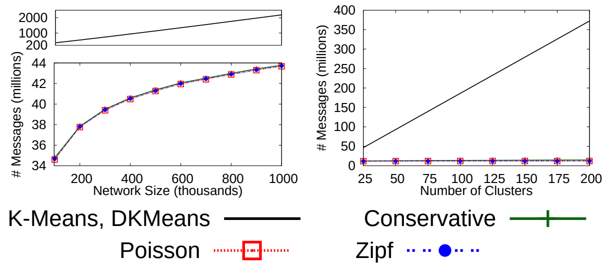


Fig. 4. Number of messages: a. varying network sizes, b. varying number of clusters

DKMeans also scales linearly, but has a significantly larger cost overall due to a larger constant factor.

Figure 3c. and Table 2 (Part IV.) displays the network cost, for varying distribution skew, and for $Pr_{pre} = 0.9$. We see that for the same quality level, PCP2P cost is substantially reduced for higher skews. This behavior is expected: with higher skews, the first few most frequent terms of documents and clusters are sufficient for finding the candidate clusters, and PCP2P requires fewer comparisons for satisfying the probabilistic guarantees. For commonly reported skew values (around 1.0), the number of messages is reduced by two orders of magnitude compared to DKMeans. Nonetheless, even for a skew as low as 0.5, the cost of both PCP2P variants is already an order of magnitude lower than the DKMeans cost. Recall that the quality remains unaffected by the skew factor (Section 7.1), since the algorithm adapts to the skew factor to satisfy the probabilistic guarantees.

Cost/quality tradeoff. Figures 5a.-b. plot the cost induced by the proposed algorithms in correlation to the achieved quality. The results correspond to the RCV1 collection, clustered over a network of 100,000 peers to 100 clusters, and the data points are determined by varying the probability Pr_{pre} in the range of 0.76 to 0.98. As expected, increasing the probabilistic guarantees yields an increase of the network cost, but also results to better clustering solutions for all examined PCP2P variants. Importantly, all PCP2P variants already approximate the exact algorithm with an approximation quality of 0.98, requiring less than 15 million messages, i.e., one order of magnitude less cost than DKMeans. Note that this approximation quality is already satisfactory for all practical

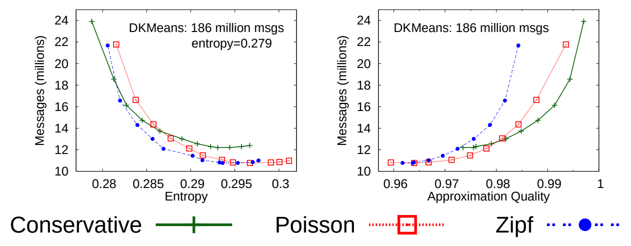


Fig. 5. Cost/quality tradeoff for PCP2P

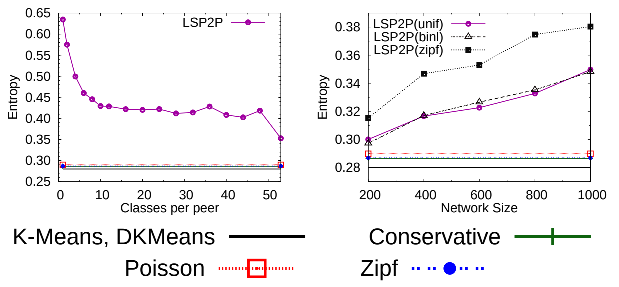


Fig. 6. Quality of PCP2P and LSP2P: a. varying the number of classes per peer, and b. varying the document distribution to peers

concerns of P2P applications. A further increase of the correctness probability induces a small quality increase, albeit with additional network cost. We also see that for high network budgets, conservative PCP2P offers a better cost/quality tradeoff compared to the Zipf-based and Poisson-based variants. Similar to the previous results (Fig 2a.-b.), Zipf-based and Poisson-based PCP2P have comparable performance, with Poisson-based performing better with respect to accuracy. With respect to entropy, the two variants do not show significant difference.

Load balancing. As explained in Section 4.1, cluster holders employ load balancing to avoid overloading. Load balancing incurs a small network overhead for synchronizing the centroids between the cluster holders and their helpers. To examine this additional overhead, we have configured the load balancing threshold such that the load of each cluster holder never exceeds twice the average expected load, and repeated all experiments. The overhead in all examined configurations was less than 100 additional messages, and less than 10 Mbytes total. Therefore, load balancing does not impede the efficiency of PCP2P.

Summary. All PCP2P variants scale well with the number of clusters and peers, and enable network savings which often exceed one order of magnitude compared to DKMeans for achieving practically the same quality. A configuration mechanism based on probabilistic guarantees enables fine tuning of the involved cost/quality tradeoffs, and load balancing at the cluster holders ensures that all peers have manageable load with negligible network overhead.

Classes per peer	Peers	Messages (millions)	Entropy		
			Uniform	Binomial	Zipf
	200	0.2	0.455	0.457	0.468
4	1000	1.2	0.494	0.496	0.535
	10000	18.0	0.622	0.624	0.655
	200	0.2	0.297	0.296	0.317
ALL	1000	1.2	0.347	0.349	0.386
	10000	18.0	0.505	0.506	0.515

TABLE 3
Performance of LSP2P

7.3 Comparison with other algorithms

PCP2P was also compared with LSP2P, the state of the art in P2P clustering. A preliminary testing of LSP2P with low-dimensional synthetic data verified the good results presented in [6], but also revealed a strong correlation of the algorithm's quality to the way the documents were distributed to the peers. Therefore, our further experiments focus on the effect of the document distribution to the compared algorithms.

Real-world peer collections are often multi-thematic, similar to real persons' interests. Some users may be well-focused, having very specific documents of only one topic. Other users may focus on a couple of non-related topics, and yet others may collect lots of diverse documents. We simulated all such users by using the document classification. Peers were creating their collections by: (a) randomly selecting i random categories/classes from the reference classification, and, (b) randomly selecting j documents for each class, ending up with $i \times j$ different documents.

We first examine the influence of the number of classes i assigned to each peer during the creation of the peer collections. Figure 6a. plots the quality of the compared algorithms, when used on a network of 1000 peers for clustering the RCV1 collection. The number of clusters was set to 100, and PCP2P was configured with $Pr_{pre} = 0.9$. The quality of DKMeans and PCP2P is independent of the number of classes per peer, since these algorithms handle each document individually. On the other hand, LSP2P requires that each participating peer carries documents from almost all classes to perform well. This is clearly a limiting factor for the applicability of LSP2P for text corpora, since it cannot be expected that real-world users have such a high variation of personal documents. In fact, in real-world networks, the number of possible classes and clusters might be even higher than the ones investigated here, aggravating the problem. The experiments with MEDLINE and SYNTH also confirmed this limitation of LSP2P.

We also examine the scalability of the algorithms, by varying the network size. To alleviate the pre-mentioned limitation of LSP2P that each peer requires a very diverse document collection, the documents were assigned to the peers randomly, ignoring the document classes. With respect to the number of documents per peer, we tested three different distributions: (a) a Zipf distribution, with skew equal to 1, (b) a binomial distribution, with average equal to $100,000/n$, where n denotes the number of peers, and, (c) a uniform distribution with the same average. These distributions are frequently considered in the literature for modeling the size of the peer collections.

Figure 6b. plots the quality of the compared algorithms. We see that uneven document distributions, such as the Zipf distribution, create additional problems to LSP2P, which become more apparent for larger networks. The quality of PCP2P on the other hand remains unaffected from the documents distribution. Similar results were also observed with the MEDLINE and the SYNTH collection.

Concerning network cost, LSP2P has the advantage that its peers exchange cluster-granularity data instead of document-granularity data. Therefore, LSP2P is more efficient than, or comparable to PCP2P, with respect to network cost (Table 3). Unfortunately, the limitation of LSP2P on large networks, and its requirement that all peers have documents from all classes, makes the algorithm unsuitable for most real-world scenarios.

Summary. LSP2P, the state-of-the-art in P2P clustering, is sensitive to the distribution of documents to peers, requiring that each peer has documents from all classes. Furthermore, it does not yield satisfactory results in large networks. PCP2P does not suffer from these limitations, thereby substantially outperforming LSP2P in real-world setups.

8 Conclusions

We presented PCP2P, the first scalable P2P text clustering algorithm. PCP2P achieves a clustering quality comparable to standard K-Means, while reducing communication costs by an order of magnitude. We provided a probabilistic analysis for the correctness of the algorithm, and showed how PCP2P adapts to satisfy the required probabilistic guarantees. Extensive experimental evaluation with real and synthetic data confirm the efficiency, effectiveness and scalability of the algorithm, and its appropriateness for text collections with a wide range of characteristics.

Our future work focuses on adapting the reclustering period to the network characteristics, i.e., the peer churn and the expected cluster shift, for further reducing the periodic reclusterings. Furthermore, we work towards a P2P IR method based on clustering, similar to [3], [4], but based on a fully distributed clustering infrastructure.

References

[1] Y. Ioannidis, D. Maier, S. Abiteboul, P. Buneman, S. Davidson, E. Fox, A. Halevy, C. Knoblock, F. Rabitti, H. Schek, and G. Weikum, "Digital library information-technology infrastructures," *Int J Digit Libr*, vol. 5, no. 4, pp. 266 – 274, 2005.

[2] P. Cudré-Mauroux, S. Agarwal, and K. Aberer, "Gridvine: An infrastructure for peer information management," *IEEE Internet Computing*, vol. 11, no. 5, 2007.

[3] J. Lu and J. Callan, "Content-based retrieval in hybrid peer-to-peer networks," in *CIKM*, 2003.

[4] J. Xu and W. B. Croft, "Cluster-based language models for distributed retrieval," in *SIGIR*, 1999.

[5] O. Papapetrou, W. Siberski, and W. Nejdl, "PCIR: Combining DHTs and peer clusters for efficient full-text P2P indexing," *Computer Networks*, vol. 54, no. 12, pp. 2019–2040, 2010.

[6] S. Datta, C. R. Giannella, and H. Kargupta, "Approximate distributed K-Means clustering over a peer-to-peer network," *IEEE TKDE*, vol. 21, no. 10, pp. 1372–1388, 2009.

[7] M. Eisenhardt, W. Müller, and A. Henrich, "Classifying documents by distributed P2P clustering." in *INFORMATIK*, 2003.

[8] K. M. Hammouda and M. S. Kamel, "Hierarchically distributed peer-to-peer document clustering and cluster summarization," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 5, pp. 681–698, 2009.

[9] H.-C. Hsiao and C.-T. King, "Similarity discovery in structured P2P overlays," in *ICPP*, 2003.

[10] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *SIGCOMM*, 2001.

[11] K. Aberer, P. Cudré-Mauroux, A. Datta, Z. Despotovic, M. Hauswirth, M. Punceva, and R. Schmidt, "P-Grid: a self-organizing structured P2P system," *SIGMOD Record*, vol. 32, no. 3, pp. 29–33, 2003.

[12] A. I. T. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *IFIP/ACM Middleware*, Germany, 2001.

[13] C. D. Manning, P. Raghavan, and H. Schtze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[14] M. Steinbach, G. Karypis, and V. Kumar, "A comparison of document clustering techniques," in *KDD Workshop on Text Mining*, 2000.

[15] G. Forman and B. Zhang, "Distributed data clustering can be efficient and exact," *SIGKDD Explor. Newsl.*, vol. 2, no. 2, pp. 34–38, 2000.

[16] S. Datta, K. Bhaduri, C. Giannella, R. Wolff, and H. Kargupta, "Distributed data mining in peer-to-peer networks," *IEEE Internet Computing*, vol. 10, no. 4, pp. 18–26, 2006.

[17] S. Datta, C. Giannella, and H. Kargupta, "K-Means clustering over a large, dynamic network," in *SDM*, 2006.

[18] G. Koloniari and E. Pitoura, "A recall-based cluster formation game in P2P systems," *PVLDB*, vol. 2, no. 1, pp. 455–466, 2009.

[19] K. M. Hammouda and M. S. Kamel, "Distributed collaborative web document clustering using cluster keyphrase summaries," *Information Fusion*, vol. 9, no. 4, pp. 465–480, 2008.

[20] O. Papapetrou, W. Siberski, F. Leitritz, and W. Nejdl, "Exploiting distribution skew for efficient P2P text clustering," in *DBISP2P*, 2008.

[21] H. F. Witschel, "Global term weights in distributed environments," *Inf. Process. Manage.*, vol. 44, no. 3, pp. 1049–1061, 2008.

[22] R. Neumayer, C. Doulkeridis, and K. Nørvåg, "Aggregation of document frequencies in unstructured P2P networks," in *WISE*, 2009.

[23] F. M. Cuenca-Acuna, C. Peery, R. P. Martin, and T. D. Nguyen, "PlanetP: Using Gossiping to Build Content Addressable Peer-to-Peer Information Sharing Communities," in *HPDC*, 2003.

[24] R. Fagin, A. Lotem, and M. Naor, "Optimal aggregation algorithms for middleware," in *PODS*, 2001.

[25] M. Theobald, G. Weikum, and R. Schenkel, "Top-k query evaluation with probabilistic guarantees," in *VLDB*, 2004.

[26] C. H. Papadimitriou, H. Tamaki, P. Raghavan, and S. Vempala, "Latent semantic indexing: a probabilistic analysis," in *PODS*, 1998.

[27] M. Steyvers and T. Griffiths, *Handbook of Latent Semantic Analysis*. Lawrence Erlbaum, 2007, ch. Probabilistic Topic Models.

[28] R. Motwani and P. Raghavan, *Randomized Algorithms*. Cambridge University Press, 1995.

[29] C. Blake, "A comparison of document, sentence, and term event spaces," in *ACL*, 2006.

[30] Y. Zhao and G. Karypis, "Empirical and theoretical comparisons of selected criterion functions for document clustering," *Machine Learning*, vol. 55, no. 3, pp. 311–331, 2004.

[31] G. K. Zipf, *Human Behavior and the Principle of Least-Effort*. Cambridge, MA: Addison-Wesley, 1949.

[32] S. Ramachandran, "Web metrics: Size and number of resources," available at http://code.google.com/speed/articles/web-metrics.html, May, 2010.