Exploiting Distribution Skew for Scalable P2P Text Clustering

Odysseas Papapetrou, Wolf Siberski, Fabian Leitritz, and Wolfgang Nejdl

Research Center L3S, Leibniz Universität Hannover
{papapetrou,siberski,leitritz,nejdl}@l3s.de

Abstract. K-Means clustering is widely used in information retrieval and data mining. Distributed K-Means variants have already been proposed, but none of the past algorithms scales to large numbers of nodes. In this work we describe a new P2P algorithm which significantly reduces the communication costs involved by exploiting distribution skew, naturally found in text and other datasets. The algorithm achieves high clustering quality and requires no synchronization between peers. An extensive evaluation with up to 100.000 peers shows the algorithm's effectiveness and scalability as well as its ability to cope with churn.

1 Introduction

Document clustering is one of the few effective methods for organizing and navigating huge information spaces. Its importance was recognized in information retrieval, for efficiently improving precision and recall [1]. It also found wide use for helping users to navigate in large document collections [2] and result sets [3], and is employed by several modern web search engines as an alternative way of presenting results.

For a centralized setting, one of the most popular clustering algorithms is K-Means [4]. K-Means is especially suitable for document clustering [5]. Although it is inherently parallelizable in a shared-memory architecture [6], distributed computation is difficult. As the number of documents and clusters grows, communications costs become too high (see Section 2).

To overcome this problem, we propose a new distributed clustering approach which approximates K-Means and reduces the network load significantly. We achieve high quality clustering as well as efficient distributed computation by exploiting the observation that word frequencies in text collections exhibit a skewed distribution [7, 8]. This allows us to focus on the most important (top-α) data dimensions when looking for cluster candidates for a document; we will consequently name our new algorithm *top-α K-Means clustering*. With respect to network topology, we rely on a Distributed Hash Table (DHT) data structure [9].

To validate the efficiency and effectiveness of top-α K-Means clustering, we perform extensive experiments with up to 100.000 peers. Our evaluation shows that the new algorithm creates high quality clustering solutions and easily scales to large networks. It achieves convergence with only a small number of iterations, and copes well with churn.

Although in this paper we focus on efficient text clustering, the proposed algorithm is suitable for any kind of high-dimensional data with Zipf [10] distribution. Especially on the Web, data frequently show this characteristic [11, 12]. For such data, our approach allows scaling up K-Means computation by just adding more computing nodes. The approach does not only apply to loosely coupled P2P networks, but also to computing clusters.

In the next section, we give an overview of related work. We present our top-α K-Means algorithm in Section 3. In Section 4, we describe the experimental setup and present the results of our extensive evaluation. We close with conclusions and future work.

2 Related work

K-Means [4], the basis of our work, is one of the most used clustering algorithms because of its low complexity (linear in the number of objects) and comparably high clustering quality. Especially for document clustering, Steinbach et al. confirmed that K-Means has comparable or better performance than Agglomerative Hierarchical Clustering algorithms while incurring significantly less cost [5].

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 centroids of each cluster as mean of all assigned documents (4) Repeat steps 2-3 until a stopping criterion is met, e.g., no documents change clusters anymore.

Parallelized K-Means: Dhillon et al. [13] describe an adaptation of K-Means for distributed memory multiprocessors. The centroid information is distributed to all processor nodes. The nodes then asynchronously assign the documents and compute new local centroids. Finally, each node broadcasts its new local centroids to all other nodes, and the nodes merge them to new global centroids. A very similar approach is presented in [14]. Although these algorithms efficiently parallelize the K-Means computation, they are restricted to a small number of nodes. In [13], when using more than 8 processors any added computing power is nearly completely used up by increased messaging costs. [14] reported only an experiment with two nodes. These algorithms fail to scale to large numbers of nodes because they require broadcasting intermediate results to all network nodes with a communication complexity of $O(n^2)$ for n nodes.

P2P K-Means: Based on Dhillon's work [13], Eisenhardt et al. proposed one of the first P2P clustering algorithms [15]. Their algorithm computes K-Means by using the PROBE/ECHO algorithmto broadcast centroid information to all connected peers. Similar to the parallel shared-nothing algorithms, this approach suffers from limited scalability due to the need of broadcasting. Hsiao and King [16] are the first employing a DHT for clustering. They recognize that indexing all terms in the DHT is too expensive. Therefore, only a small number of manually selected terms is indexed. This requires extensive human interaction, and the network cannot dynamically adapt to new documents and topics.

Hammouda and Kamel [17] use a hierarchical topology for the coordination of K-Means computation. Peers on the lowest level form groups and compute local clustering solutions. For each group, a representative peer joins the next level, using the group's centroids as data points. This process is repeated until the top of the hierarchy, where a single root peer computes the final solution. The main disadvantage of this algorithm is that the quality decreases significantly for each aggregation level, as a result of the random grouping of peers. In the reported experiments, top-level F-measure drops to less than 50% of the F-measure achieved by centralized K-Means. With growing network sizes the quality drops significantly, reaching less then 20% of the central K-Means quality for 65 nodes.

To avoid the cost of broadcasting, Datta et al. [18] employ a gossiping approach for distribution of the centroids. Each peer first conducts local K-Means clustering, and then asks its neighbors for their local centroids. The averages over these local centroids are used as the new centroids for the next iteration. With each iteration, the computed centroids improve their approximation for the global clustering solution. The algorithm terminates when the change between centroids in two subsequent iterations is lower than a threshold value. Compared to earlier attempts, the algorithm reduces communication requirements considerably. Evaluations with 10-dimensional synthetic data show an average misclassification error less than 3% (3.5% in dynamic environments). However, as we show in our experiments, the algorithm has difficulties scaling to large P2P networks for high-dimensional data such as documents (see Section 4).

3 Top-α K-Means Distributed Clustering

Top-α K-Means reduces clustering cost and maintains high clustering quality by focusing on the most frequent terms for each cluster and document. The most frequent terms contribute more to the similarity measure, the cosine similarity in our case. So top-α K-Means uses these terms for an estimation of the similarity measure between clusters and documents, and then it fully compares the documents with only the most promising clusters. Since term occurrences in documents and clusters follow Zipf distribution, top-α K-Means can make a good approximation of the final similarity measures by using only very few of the frequent terms.

3.1 Algorithm Overview

Given a P2P network where peers are structured over a DHT overlay. Each peer in the network carries a set of documents. We require a clustering algorithm for clustering all documents from the network to k clusters.

In centralized K-Means, it is affordable to compare each document with each cluster centroid, to decide on the most suitable cluster. However, as pointed out in Section 2, in a distributed setting the network costs for these comparisons become prohibitively high if peers exchange complete centroids and/or documents. The document assignment process needs to be adapted to reduce these

network costs. Our algorithm exploits the fact that term distribution is highly skewed. This allows us to pre-select the most relevant centroids based on the most frequent terms (after stopword removal). Instead of comparing a document with all centroids, only these few selected centroids are used for full comparison.

As first step we compute an approximation of the centroid similarity by considering only the most frequent (top-α_2) document terms. To limit the error introduced because of this approximation, we select a few (top-α_3) most similar clusters as candidates. A full cosine similarity computation between document and centroid is performed only for these candidates, and the document is assigned to the cluster with the highest cosine similarity.

To determine the cluster candidates, we need efficient access to centroid statistics by term, i.e., an inverted cluster index. This index is maintained within a Chord-based DHT, formed by all peers in the network. For each term, an index entry consists of the list of clusters that have published this term and their details. In particular, the list holds the cluster id, cluster length, frequency of the respective term, and the peer that holds the cluster centroid (see the sample entries on the right-hand side of Fig. 1). Again, we exploit the skewed term distribution and do *not* add this information for all the terms in a cluster. Instead, a cluster inserts this information into the index only for its most frequent (top-α_1) terms. See Table 1 for an overview of the approximation parameters.

Algorithm 1 Document Assignment

```
1: //Each peer p repeats the following periodically:
2: for all Document d in p do
3:    compute partial cosine similarities with centroids considering only top-α2 terms
      of d and top-α1 terms of centroids
4:    for top-α3 most similar clusters c do
5:       compute full cosine similarity between c and d
6:    end for
7:    Assign d to cluster c with highest cosine similarity
8: end for
```

As usual in information retrieval, documents are preprocessed by stemming and removal of stop-words. The latter is particularly important as we do not want the top-α term lists to become polluted with stopwords.

Periodic execution: Similar to the centralized K-Means algorithm, clustering is performed in iterations. For each iteration, top-α K-Means consists of two independent activities, *cluster index maintenance* and *document assignment*. We describe these activities in detail in the next subsections.

3.2 Cluster Index Maintenance

A core component of our algorithm is the distributed cluster index maintained in a DHT (see Figure 1). The peer initiates the clustering algorithm by assigning the

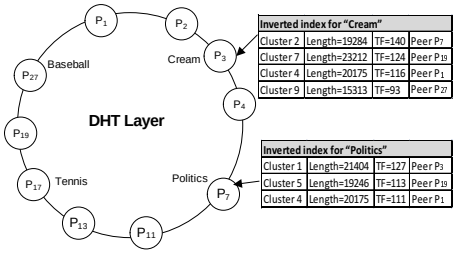

Fig. 1: DHT-based Cluster Inverted Index

α_1	Top-most frequent terms per cluster to publish at DHT
α_2	Top-most frequent terms per document to lookup at DHT
α_3	Top-most relevant clusters according to partial similarity, to fully compare with the document

Table 1: Algorithm parameters

maintenance of each cluster centroid to randomly selected peers (see Section 3.4). We call these peers *cluster holders*. All cluster holders *periodically* update the DHT index with information for their centroid. Because of this periodic nature, the cluster holders never need to unpublish old data; data which are not refreshed for a given time period expire and are removed from the DHT.

As already mentioned, a cluster holder does not publish index entries for all terms occurring in the cluster; only the top-α_1 most frequent terms are considered when submitting entries to the DHT. For these terms, the indexing message consists of term frequency and IP address of the cluster holder. Complete cluster centroids never have to be passed to other peers. This saves significant network resources for high-dimensional data. The total number of messages issued by the cluster holders to maintain the cluster indices is $Msgs \leq k{*}\alpha_1{*}(\log(n){+}1)$. Transfer volume is upper bounded by $TransferVol \leq k{*}\alpha_1{*}(\log(n){+}1){*}avgTermLen$, where $avgTermLen$ denotes the average term length.

Algorithm 2 Cluster Indexing

1: //Each cluster holder c repeats the following periodically:
2: **for** top-α_1 most frequent terms t in cluster **do**
3: DHT.insert(t, <ipaddr(c),length(c),TF(t, c)>)
4: **end for**

Cluster centroids are periodically updated to reflect the current cluster contents. In a centralized environment, centroids would be updated at the end of every iteration; here, they are recomputed at regular intervals, and the DHT index is updated accordingly. All cluster holders use the same update interval, but no synchronization between them is required.

3.3 Document Assignment

As with index maintenance, the assignment of documents to clusters is repeated periodically. Document assignment is split in two steps: (a) looking up candidate clusters in the DHT, and (b) choosing the best of them for assigning the document.

To cluster one of its documents, the peer first identifies the document's *top-*α_2 most frequent terms. For these terms, the peer performs a DHT lookup and collects the potentially relevant clusters, i.e., clusters that have published at least one of these terms. A partial cosine similarity is computed between the document and the potentially relevant clusters, considering only the terms of the clusters that are retrieved from the DHT and assuming a TF of 0 for all other terms. The clusters are ordered by this similarity score, and the top-α_3 most similar clusters are selected as candidates. A full cosine similarity comparison of the candidate cluster centroids with the document is then performed. For this purpose, the peer sends the document vector to the respective candidate cluster holders which return the corresponding similarity score. Although this incurs more computation cost for the peers responsible for holding the clusters, it saves significant communication costs because document vectors are orders of magnitude smaller than complete centroid vectors. Finally, the peer submits the document to the most similar cluster.

For k clusters and n peers, the network cost for clustering a document is as follows. Performing a DHT lookup on the top-α_2 document terms requires on average $\alpha_2 * \log(n)$ messages. Comparing the document with the α_3 most promising candidate clusters requires $2 * \alpha_3$ messages. An additional message is required for assigning the document to the selected cluster. The total number of messages for clustering a document is upper bounded by $Msgs \leq \alpha_2 * \log(n) + 2 * \alpha_3 + 1$. The respective transfer volume in bytes is approximately $TransferVol \approx \alpha_3 * docLength + \alpha_2 * \log(n) * avgTermLen$, where $docLength$ denotes the length of the document and $avgTermLen$ denotes the average term length. Note that in contrast to all prior algorithms, number of messages and transfer volume are not linear to k, but only depend on the approximation parameters α_1, α_2 and α_3, which are much smaller than k. Our evaluation indicates that for increasing k, α_3 can be kept constant and α_1 and α_2 need to increase only logarithmically to maintain the same clustering quality.

3.4 Further Aspects

Bootstrapping and terminating the algorithm: Any peer can initialize the clustering algorithm by selecting k random peers from the DHT, and making them cluster holders. Each cluster holder selects a random document from its collection as initial centroid. No further synchronization is required for newly-joined peers: they retrieve the value of k from the DHT entry point and start clustering their documents immediately. With respect to algorithm termination, typically in P2P systems the document collection held by the peers continuously changes, because of network churn and individual peer collection updates. Therefore, we need to run the clustering algorithm continuously if the goal is to always have a high-quality document clustering available.

Load Balancing: For load-balancing purposes, the responsibility of keeping a cluster may need to be split between two or more peers. Peers randomly select one of these peers as cluster holder when assigning a document. Coordination between several cluster holders only occurs for submission of centroid updates to

the DHT. Similar to [13], all cluster holders for a certain cluster exchange their local centroids and merge this information to compute the global centroid. As only one centroid needs to be transferred per iteration per cluster, and only a small constant number of peers is communicating, this does not affect scalability.

Periodic re-clustering and churn: Our approach can adapt to the frequency of updates and churn percentage by adjusting the length of the document assignment and publishing periods. After each document assignment cycle, peers check how many of their document assignments have changed. The smaller the fraction of changes, the longer the interval between updates becomes. Similarly, each cluster holder compares the current cluster contents with the published ones and adjusts its republishing interval accordingly. The republishing intervals need not be equal among all cluster holders; each peer decides on its own interval, and the expiring time for its terms is adjusted accordingly.

Inverse Document Frequency: Top-α K-Means does not include IDF scores in similarity computation. Existing proposals for maintenance of IDF values over P2P networks (e.g. [19]) require a complete term inverted index built over the DHT. In contrast, top-α K-Means only maintains information for a small fraction of the terms in the DHT. Indexing all the terms for the purpose of the IDF computation would introduce a high communication overhead. This is not justified by the increase in quality. Our experiments showed that the usage of IDF only increases quality scores by less than 10%. In particular, clustering 100.000 documents from the Reuters RCV1 collection with IDF gave 8% improvement on F-Measure compared to the results without IDF (0.439 compared to 0.406), and 1% improvement on entropy (1.53 compared to 1.54). The same quality improvement can be achieved with lower costs by adapting the algorithm approximation parameters.

4 Experimental Evaluation

The objective of our evaluation was twofold. First, we wanted to evaluate the effectiveness of the generated clustering solution, in both static and dynamic large P2P networks. Second, we wanted to determine the efficiency of the algorithm regarding network costs.

We evaluated top-α K-Means by comparing it to the following algorithms:

Basic distributed K-Means: This distributed K-Means implementation accurately simulates the centralized K-Means, thus it achieves exactly the same quality as its centralized counterpart. First, each cluster centroid is assigned to a peer (cluster holders). The addresses of all the cluster holders are maintained in the DHT using the cluster id as a key: $\{1, 2, \ldots k\}$. Then, a document is clustered by: (a) locating all the peers that hold the k cluster centroids, (b) forwarding the document to each of these peers for comparison, and (c) assigning the document to the cluster with the most similar centroid. To save messages, the peers cache the cluster holders IP addresses locally, and only access the DHT when a cluster holder becomes unreachable because of churn.

Gossiping-based K-means: The gossiping-based K-Means algorithm, proposed by Datta et al. [18], is the state-of-the art in distributed clustering (cf. Section 2). It is also the only distributed clustering algorithm applicable to high-dimensional data.

We evaluated the three algorithms by simulating them in large P2P networks with real documents, and by comparing F-measure and entropy values (see [5] for a description of the evaluation measures). We also measured the effectiveness of the two algorithms which had shown acceptable clustering quality: the basic distributed K-Means and the top-α K-Means algorithm. Effectiveness was measured with the number of messages and total transfer volume that each algorithm used per clustering iteration. All the network activity was measured, except of the messages required for connecting peers with the DHT overlay; these messages were equal in both the compared algorithms.

Document Collections. We evaluated the algorithms on two large document collections: (a) the Reuters Corpus Volume I (RCV1) dataset [20], which includes more than 800.000 Reuters articles, and (b) the Reuters-21578 collection, which includes 21578 Reuters articles. Both collections come with a subject classification, which we used as reference to compute F-Measure and entropy. We do not include the results for the smaller collection in this paper as there was no significant difference in the outcome. For the RCV1 collection, we filtered out very short documents (less than 50 words). From the remaining documents we chose the first 130.000 for our experiments.

4.1 Comparison with Gossiping-based K-Means

Gossiping-based K-Means is currently the most advanced P2P clustering algorithm. The results reported in [18] show good clustering quality for network sizes up to 1000 peers. However, P2P algorithms based on random networks, as Gossiping-based K-Means is, are prone to limited scalability. We compared Gossiping-based K-Means and top-α K-Means to investigate their scalability with respect to clustering quality.

For this experiment series, we first randomly selected 100.000 of the 130.000 documents of the collection, and distributed them uniformly to N peers. To form the random network which is assumed by gossiping-based K-Means, each of the N peers established links to $\log(N)$ other peers. For top-α K-Means, the peers formed a DHT as explained in Section 3. We ran the simulation for different network sizes $N = \{500, 1000, 2000, 4000\}$, as well as for different number of clusters $K = \{25, 50, 100\}$. Each setup was executed 4 times. After each iteration, we introduced churn by randomly selecting 20% of the peers and replacing them with an equal number of new peers, which were initialized by using some of the remaining 30.000 documents. The collections of the removed peers were returned to the pool of extra documents.

Figure 2 presents averages of the results for varying network sizes, for $K = 50$ (the results with $K = 25$ and $K = 100$ were similar). It shows entropy (lower is better) and F-measure (higher is better) after 20 iterations, when quality scores

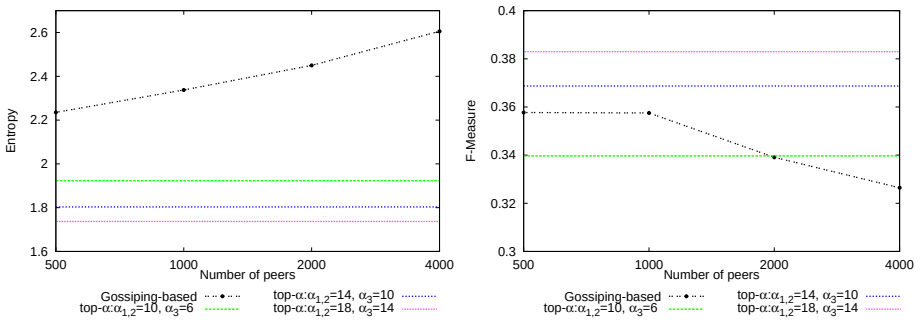

Fig. 2: Gossiping Vs Top-α clustering quality: (a) entropy, (b) F-measure

had stabilized. For the top-α algorithm, we present results for some sample configurations; see next subsection for more details. For small networks, gossiping-based K-Means achieves a good F-measure. However, when the network grows larger than 1000, the quality degrades significantly. On the other hand, the effectiveness of top-α K-Means is orthogonal to the network size. For networks larger than 2000 peers, top-α K-Means outperforms the gossiping-based approach even for small values of $\alpha_{\{1|2|3\}}$. The entropy score of gossiping-based K-Means is worse than the score of top-α K-Means in all cases.

We repeated the experiment also without churn, with similar results. Both algorithms achieved slightly better F-measure and entropy values, but the relation between the two did not change.

Summarizing, the experiments show that gossiping-based K-Means has problems scaling up to large P2P networks. In particular, gossiping-based K-Means starts failing for network sizes over 1000 peers. In contrast, the effectiveness of our algorithm is independent of the number of peers.

4.2 Comparison with Basic Distributed K-Means

In the second experiment series, we compared the top-α K-Means with the basic distributed K-Means. The latter algorithm computes the same clustering solution as the central variant and serves as quality and efficiency baseline.

We constructed the experiments in the same way as described in Section 4.1, for network sizes N between 2000 and 100.000 and number of clusters K between 25 and 100. Values for the approximation parameters $\alpha_{\{1|2|3\}}$ varied between 2 and 22. We do not report all the results here in detail, but focus on $N = 100.000$ and $K = 50$. The other results, which were very similar, are only summarized.

Effectiveness: Figure 3 shows how the choice of approximation parameters influences clustering quality. It plots the average entropy (lower is better) and F-measure (higher is better) for the two algorithms. For comparison purposes we include the results for gossiping-based K-Means with 1000 peers in the same plot.

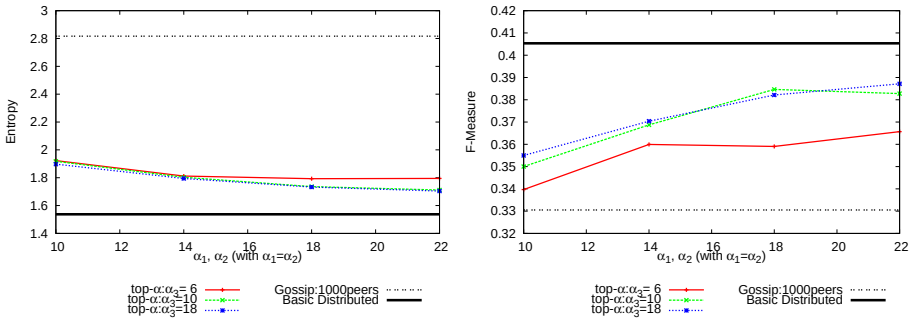

Fig. 3: Top-α Vs Basic Distributed K-Means clustering quality: (a) entropy, (b) F-measure

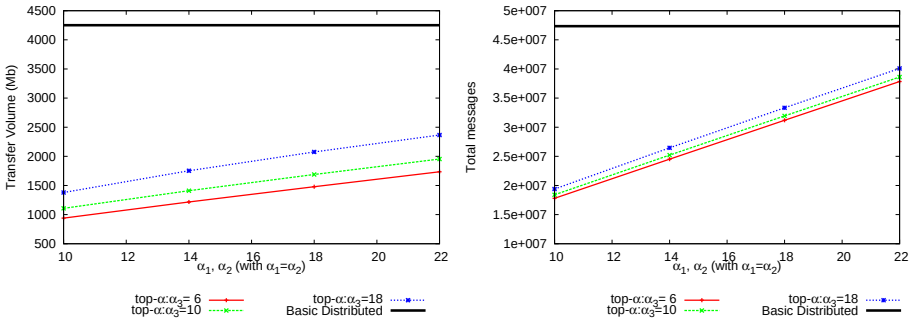

Fig. 4: Top-α Vs Basic Distributed K-Means cost:(a) Transfer Volume (b) Total messages per iteration

Top-α closely approximates the basic distributed K-Means algorithm in terms of quality. The optimal value for α_3 is around 10. Lower values cause a decrease of quality, and higher values do not have a significant effect. Setting α_1 and $\alpha_2 \geq 14$ already yields more than 90% of the effectiveness of original K-Means.

Efficiency: Figure 4 presents the effect of parameter choice on communication costs. It plots the network transfer volume and number of messages (both including DHT lookups) per iteration. Top-α K-Means is always much less expensive than the basic distributed algorithm. Savings are especially high in terms of transfer volume. The reason is that the largest part of the transfer volume is used for sending the documents to the cluster holders for comparison, and top-α significantly reduces these comparisons. The values of $\alpha_{\{1|2|3\}}$ also make a significant difference on the cost. For low $\alpha_{\{1|2|3\}}$ values, the cost is low, but the quality (Figure 3) is also not sufficient. Moderate values for $\alpha_{\{1|2|3\}}$ (around 10) give sufficient quality (approximate the optimal by 90%) at less than half the cost of the basic algorithm. A further increase of the approximation parameters has only a minor effect in the quality of the algorithm.

Cost-quality tradeoff – the 90% point: We also found which minimal values of α_1, α_2 and α_3 give solutions with 90% of the quality (measured with average F-measure over all repetitions) of the original K-Means. Table 2 summarizes the results for 100.000 peers and $K = 25, 50, 100$. We see that the increase on the number of clusters only requires a logarithmic increase on the values of α_1, α_2, α_3 to keep the same quality level.

Configuration				Cost Savings	
K	α_1	α_2	α_3	Messages	Transfer volume
25	10	10	6	74%	73%
50	14	14	10	54%	37%
100	18	18	10	34%	2O%

Table 2: 90% point: Cost savings compared to basic K-Means in a 100.000 peer network

Varying the network size has no effects on clustering quality. However, the network size does affect the cost of the two algorithms, e.g., for $K = 50$ top-α K-Means with 4000 and 2000 peers required 35% and 32% of the respective transfer volume of the Basic Distributed K-Means algorithm to reach 90% quality. Varying the size of the collection to 25.000, 50.000 and 75.000 documents did not have significant effects on the clustering quality ratio of the compared algorithms.

Summarizing, Top-α K-Means easily achieves over 90% of the quality of the optimal K-Means results with only a fraction of the cost (half of the messages and 2/5 of the transfer volume of the basic distributed K-Means for 50 clusters). This result is independent of churn and the number of peers, and the savings increase with the number of clusters.

5 Conclusions

This paper presents a novel P2P document clustering algorithm based on K-Means. The algorithm is highly scalable and suitable for text and other high-dimensional data that follow a Zipf distribution. In contrast to prior algorithms, its communication costs are not linearly dependent on the number of clusters. Extensive experiments with up to 100.000 peers show that the algorithm outperforms the current state-of-the-art P2P clustering algorithm. Top-α K-Means is highly efficient and achieves nearly the same quality as the original K-Means.

Our future work will focus on a probabilistic model for the algorithm. In particular, we plan to extend the algorithm such that peers can dynamically adapt α_1, α_2, and α_3 to keep the clustering approximation error in a given bound. A prerequisite for such dynamic parameter optimization is a full probabilistic model. We are already on the way to such a model for our algorithm. The purpose of this analysis is to enable each peer to autonomously determine the optimal parameter settings. We are also performing evaluations with different

data sets and other document collections to experimentally validate the broad applicability of our approach.

References

1. van Rijsbergen, C.J.: Information Retrieval. Buttersworth, London, UK (1989) second edition.
2. Cutting, D.R., Pedersen, J.O., Karger, D.R., Tukey, J.W.: Scatter/gather: A cluster-based approach to browsing large document collections. In: SIGIR. (1992)
3. Zamir, O., Etzioni, O., Madani, O., Karp, R.M.: Fast and intuitive clustering of web documents. In: KDD. (1997)
4. McQueen, J.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability. (1967) 281–297
5. Steinbach, M., Karypis, G., Kumar, V.: A comparison of document clustering techniques. In: KDD Workshop on Text Mining. (2000)
6. Jin, R., Yang, G., Agrawal, G.: Shared memory parallelization of data mining algorithms: Techniques, programming interface, and performance. TKDE **17** (2005)
7. Baeza-Yates, R.A., Navarro, G.: Block addressing indices for approximate text retrieval. JASIS **51** (2000) 69–82
8. Blake, C.: A comparison of document, sentence, and term event spaces. In: ACL. (2006)
9. Stoica, I., Morris, R., Karger, D., Kaashoek, F., Balakrishnan, H.: Chord: A scalable Peer-To-Peer lookup service for internet applications. In: SIGCOMM. (2001)
10. Zipf, G.K.: Human Behavior and the Principle of Least-Effort. Addison-Wesley, Cambridge, MA (1949)
11. Crovella, M.E., Taqqu, M.S., Bestavros, A.: Heavy-tailed probability distributions in the World Wide Web. In: A practical guide to heavy tails: statistical techniques and applications. Birkhauser Boston Inc. (1998) 3–25
12. Kumar, R., Novak, J., Raghavan, P., Tomkins, A.: On the bursty evolution of blogspace. In: WWW. (2003)
13. Dhillon, I.S., Modha, D.S.: A data-clustering algorithm on distributed memory multiprocessors. In: Workshop on Large-Scale Parallel KDD Systems. (1999)
14. Forman, G., Zhang, B.: Distributed data clustering can be efficient and exact. SIGKDD Explor. Newsl. **2** (2000) 34–38
15. Eisenhardt, M., Müller, W., Henrich, A.: Classifying documents by distributed P2P clustering. In: INFORMATIK, Frankfurt, Germany (2003) 286–291
16. Hsiao, H.C., King, C.T.: Similarity discovery in structured P2P overlays. In: ICPP. (2003)
17. Hammouda, K., Kamel, M.: HP2PC:scalable hierarchically-distributed peer-to-peer clustering. In: SDM. (2007)
18. Datta, S., Giannella, C., Kargupta, H.: K-means clustering over a large, dynamic network. In: SDM. (2006)
19. Bender, M., Michel, S., Triantafillou, P., Weikum, G.: Global document frequency estimation in peer-to-peer web search. In: WebDB. (2006)
20. Lewis, D.D., Yang, Y., Rose, T.G., Li, F.: Rcv1: A new benchmark collection for text categorization research. J. Mach. Learn. Res. **5** (2004) 361–397