Collaborative Document Clustering

Khaled Hammouda* Mohamed Kamel†

Abstract

Document clustering has been traditionally studied as a centralized process. There are scenarios when centralized clustering does not serve the required purpose; *e.g.* documents spanning multiple digital libraries need not be clustered in one location, but rather clustered at each location, then enriched by receiving more information from other locations. A distributed collaborative approach for document clustering is proposed in this paper. The main objective here is to allow peers in a network to form independent opinions of local document grouping, followed by exchange of cluster summaries in the form of keyphrase vectors. The nodes then expand and enrich their local solution by receiving recommended documents from their peers based on the peer judgement of the similarity of local documents to the exchanged cluster summaries. Results show improvement in final clustering after merging peer recommendations. The approach allows independent nodes to achieve better local clustering by having access to distributed data without the cost of centralized clustering, while maintaining the initial local clustering structure and coherency.

1 Introduction

Information is better utilized when it is processed to be easier to find, better organized, or summarized for easier digestion. Areas dealing with such problems are at the cross-roads of information retrieval, machine learning (*e.g.* classification and clustering), and statistical analysis. Text and web mining problems in particular use methodologies often spanning those areas.

Document clustering is an area that deals with the unsupervised grouping of text documents into meaningful groups, usually representing topics in the document collection. It is one way to organize information without requiring prior knowledge about the classification of documents, and could be used as a base for document categorization by forming an initial classification.

Document clustering has many applications, such as clustering of search engine results to present organized and understandable results to the user (*e.g.* Vivisimo[1]), clustering documents in a collection (*e.g.* digital libraries), automated (or semi-automated) creation of document taxonomies (*e.g.* Yahoo and Open Directory styles), and efficient information retrieval by focusing on relevant subsets (clusters) rather than whole collections. Perhaps the most popular application of document clustering is the Google News[2] service, which uses document clustering techniques to group news articles from multiple news sources to provide a combined overview of news around the Web.

Traditionally, document clustering has been studied as a centralized process; *i.e.* all data is assumed to be present at a central site, then a single process applies clustering to the data. A more wider view of how clustering can be applied in distributed environments is outlined in Table 1.

Table 1: Types of data and clustering process distribution

	Centralized Data	Distributed Data
Centralized Clustering	CD-CC	DD-CC
Distributed Clustering	CD-DC	DD-DC

Centralized Data - Centralized Clustering (CD-CC) This is the standard approach where the clustering process and data both reside on the same machine.

Distributed Data - Centralized Clustering (DD-CC) Data might be dispersed across different machines, a typical case in the Web domain, while the clustering process runs on a single machine.

Centralized Data - Distributed Clustering (CD-DC) Data is stored in one location, with clustering processes running on different machines accessing the same data; a typical case of *parallel processing*.

Distributed Data - Distributed Clustering (DD-DC) The highest level of distribution, where both the data and the clustering processes are distributed.

*Systems Design Engineering, Pattern Analysis and Machine Intelligence Research Group, University of Waterloo, Waterloo, Ontario, Canada. Email: hammouda@pami.uwaterloo.ca

†Electrical & Computer Engineering, Pattern Analysis and Machine Intelligence Research Group, University of Waterloo, Waterloo, Ontario, Canada. Email: mkamel@pami.uwaterloo.ca

[1] www.vivisimo.com

[2] news.google.com

The problem with centralized clustering is that sometimes raw data is distributed across different databases, making it either infeasible or impossible to apply centralized clustering. In other situations having a global clustering solution of the distributed data is not required; a local clustering solution is sufficient, which can be augmented or enhanced by having access to *summarized cluster information* from peer nodes.

To better motivate the above scenarios, consider a set of digital libraries (*e.g.* archived articles from online publishers). Each digital library can form an opinion about the topic groups found in its collection by applying local clustering. To enrich the local clustering solution, each library can receive recommendations from other libraries on what articles, which it currently does not carry, can fit into its current clustering solution. Thus achieving wider accessibility to previously unavailable data and expanding its clustering solution, while maintaining its initial structure which is based on the local data.

Another scenario is when access to peer data is restricted. In this case we do not have a full global dataset to operate upon. Instead, we can still achieve a local clustering that includes pointers to peer data. This is possible by allowing peers to look at local cluster summaries and then recommend to us those documents that they think can fit within our clustering.

In this paper we are presenting an approach that enables *collaborative clustering* among distributed nodes. By collaborative we mean the ability of one node to benefit other nodes based on their needs. The stricter term "cooperative clustering" is not being used as the aim of cooperative clustering is to achieve a common benefit (*e.g.* a global clustering solution). However, in the collaborative approach we are aiming at enriching the local clustering solution of each individual node based on recommendations from peer nodes.

The collaborative approach presented here can be considered one approach to "distributed clustering". There are many approaches that rely on a distributed framework for doing the task; the collaborative approach tries to utilize the distributed framework to achieve better local clustering through peer involvement and recommendation.

Figure 1 illustrates the collaborative clustering system presented in this paper. From the point of view of system architecture, we are adopting a *peer-to-peer* framework. Each node in the network has access to part of the whole document collection. The network is a connected graph, in which every node is connected to every other node.

Every node starts by generating a local cluster model of the documents to which it has access. The goal is to enhance the clustering solution at each node by allowing peers to receive recommended documents so that the local clustering quality is maximized.

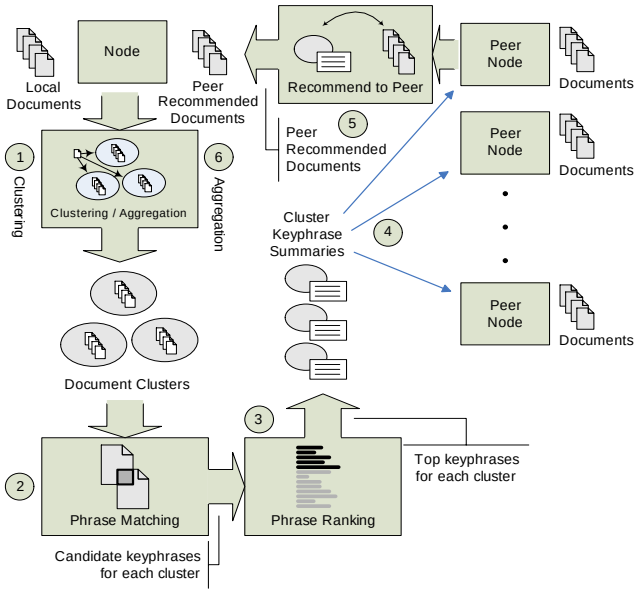

Figure 1: Collaborative Document Clustering System

The information exchanged between nodes is minimized by forming of cluster summaries, represented as a vector of core keyphrases in each cluster. This step involves an elaborate multi-document keyphrase extraction algorithm that accurately extracts those keyphrases that best describe each cluster [10].

Each node applies a similarity calculation between received cluster summaries and its own set of documents. It then recommends to each peer what set of documents can benefit the peer's clustering solution. Finally, the receiving peers decide which documents should be merged with their own clusters. Eventually, each node should have an improved set of clusters over the initial solution, facilitated by having access to global information from its peers.

The rest of this paper is organized as follows. Section 2 presents related work, and positions the work within the literature. Section 3 presents the collaborative document clustering model. Section 4 presents the algorithms for initial cluster generation, cluster summarization using keyphrase extraction, and collaborative clustering. Section 5 includes explanation and discussion of the experimental results. Finally, a conclusion and future research directions are outlined in Section 6.

2 Related Work

The literature is rich on data mining topics. Text mining in particular receives good attention due to its wide applicability in many domains. Clustering as a methodology in general has roots in the statistical analysis as well as the machine learning fields. However, we have found very little number of attempts in the area of distributed clustering, which is considered a fairly new area due to recent advances in networking and fast processors. In this section we try to relate the work presented here to other work in the literature, and position the work appropriately.

Data clustering is an established field. Jain *et al* [11, 12] cover the topic very well from the point of view of cluster analysis theory, and they break down the methodologies mainly into *partitional* and *hierarchical* clustering methods. In our work the clustering algorithm used falls under the partitional category, since we do not build hierarchies of clusters. However we allow clusters to overlap. Chakrabarti [5] also discusses various types of clustering methods and categorizes them into partitioning, geometric embedding, and probabilistic approaches. Since our algorithm is mainly dependent on statistical analysis, we can position it as a probabilistic algorithm.

Kosala and Blockeel [14] touch upon many aspects of web mining, showing the differences between web content mining, web structure mining, and web usage mining. They also discuss what kind of methodologies are used in web mining. Since we are analyzing the actual text content of web pages, as opposed to web link structure or server logs, we can say that our work falls under the category of web content mining.

Text mining research in general rely on a vector space model, first proposed in 1975 Salton *et al* [19, 18, 17] to model text documents as vectors in the feature space. Features are considered to be the words in the document collection, and feature values come from different term weighting schemes, the most popular of which is the TF-IDF (Term Frequency-Inverse Document Frequency) term weighting scheme. This model is simple but assumes independence between words in a document, which is not a major problem for statistical-based methods, but poses difficulty in phrase-based analysis. Thus, we rely on a model, the Document Index Graph, which explicitly represents phrases as a graph structure [9]. This model is the basis for efficient keyphrase extraction in our work.

Keyphrase extraction can be divided into keyphrase extraction from single documents, often treated as a supervised learning problem [20], and extraction from multiple documents, often treated as an unsupervised problem [6]. The work by Barzilay *et al* [1] seems to have common approach to ours in terms of multi-document summarization. However, they focus on one domain, news articles, to detect the same event in multiple documents, and they use natural language processing techniques to supplement summaries with synthesized text. Again, most research in this area actually treats keyphrases as individual words, again breaking the phrase structure, which is a key in properly labeling clusters, as is used in this paper.

Posing the problem as a distributed data mining problem as opposed to the traditional centralized approach was largely due to Kargupta *et al* [13]. They proposed a distributed agent architecture in which each agent has a local model of the world. Agents then cooperate to achieve better global solution. In many aspects this is very related to our work. The main difference is that we do not cast the problem into a multi-agent paradigm, but rather as straightforward distributed processing problem. In addition, our method does not directly try to achieve a better global solution, but rather achieves this indirectly by maximizing the local solutions through collaboration of peers.

The work presented by Eisenhardt *et al* [7] seems to have very similar goal to ours: to solve the document clustering problem using a distributed peer-to-peer network. They use the k-means clustering algorithm, modified to work in a distributed fashion. The algorithm works by propagating the cluster mean vector to peers, which merge the mean vector with their own solution and report back the merged mean to the sending peer. They report improvement in speed up compared to centralized clustering, but they do not provide an evaluation of clustering quality in the distributed environment. A similar system can be found in [15], but the problem is posed from the information retrieval point of view.

3 Collaborative Document Clustering

The proposed framework consists of a number of models that describe different aspects of the problem. In this section we discuss how we represent the distributed nodes in the system, how to represent the data distribution, how to represent clusters of documents, and finally how the documents themselves are represented.

3.1 Node Distribution Model The node distribution model follows a peer-to-peer model. The network is represented as a connected graph $\mathbf{G}(\mathbf{N}, \mathbf{L})$

where $\mathbf{N}$: is the set of *nodes* $\{n_i\}$, $i = 1, \ldots, N_N$.
$\mathbf{L}$: is the set of *links* $\{l_{ij}\}$ between every pair of nodes n_i and n_j, $i, j = 1, \ldots, N_N$, $i \neq j$. The number of links in the network $N_L = N_N(N_N - 1)/2$.

This kind of connectedness is chosen for the sake of simplicity, since the typical scenarios for our approach involves only several nodes.

The number of nodes is static; *i.e.* N_N is fixed for a certain configuration and cannot be dynamically changed. We, however, test with different network sizes for different values of N_N.

Node links are assumed to be equally weighted. Nodes are assumed to have identical roles in the network, hence the peer-to-peer framework. The model also suggests that each node is assumed to have information about how to find all other nodes.

3.2 Data Distribution Model The global view of the data is a collection of documents $\mathbf{D} = \{\mathbf{d}_k\}$, $k = 1, \ldots, N_D$. The original classification of the documents is assumed to be known, but not used during clustering. It is only used during evaluation.

The set of topics[3] are represented as $\mathbf{T} = \{t_c\}$, $c = 1, \ldots, N_T$. Each document is assumed to belong to one topic; *i.e.* Topic$(\mathbf{d}_k) \in \mathbf{T}$.

The document collection is assumed to be randomly, but evenly, distributed among the network nodes; *i.e.* each node holds the same number of documents, which is a percentage α of the total number of documents N_D. We refer to the parameter α as the *distribution ratio*. There could be overlap between the nodes in terms of the documents they hold; *i.e.* $1/N_N \leq \alpha \leq 1$. Thus, node i will hold a set of documents $\mathbf{D}_i \subseteq \mathbf{D}$, such that $|\mathbf{D}_i| = N_{D_i} = \alpha \cdot N_D$.

This type of document distribution mimics realistic scenarios, where each node can have access to either an exclusive part of the document collection, or there could be an overlap between the documents in each node.

3.3 Cluster Model Upon clustering the documents, each node will have created a set of document clusters that best fit its local document collection. Thus, each node n_i maintains a set of clusters $\mathbf{C}_i = \{\mathbf{c}_r\}$, $r = 1, \ldots, N_{C_i}$. A cluster contains a subset of the documents in the node; *i.e.* $\mathbf{c}_r$ contains $\mathbf{D}_r \subseteq \mathbf{D}_i$.

Clusters are allowed to overlap; *i.e.* each document can belong to more than one cluster, either in the same node or across different nodes. Thus, $\forall \mathbf{d}_k$, it is possible that $\exists \mathbf{c}_1, \mathbf{c}_2 \in \mathbf{C}$, such that $\mathbf{d}_k \in \mathbf{c}_1$ and $\mathbf{d}_k \in \mathbf{c}_2$.

Document-to-cluster membership is represented as a relation $R(\mathbf{D}, \mathbf{C})$, which is a binary membership matrix:

$$\mathcal{M} = [m_{k,r}]$$

$$m_{k,r} = \left\{ \begin{array}{ll} 1 & \text{iff } \mathbf{d}_k \in \mathbf{c}_r \\ 0 & \text{otherwise} \end{array} \right\}$$

A projection of $\mathcal{M}$ over the documents dimension yields the the number of documents in each cluster:

$$[\mathcal{M} \downarrow \mathbf{D}]_r = \sum_k m_{k,r}$$

Similarly, a projection of $\mathcal{M}$ over the clusters dimension yields the the number of clusters to which each document belongs:

$$[\mathcal{M} \downarrow \mathbf{C}]_k = \sum_r m_{k,r}$$

In the process of collaborative clustering, there will be a need to represent cluster summaries in the form of keyphrase vectors, so that they can be exchanged between peers. The summary of cluster $\mathbf{c}_r$ is represented as a keyphrase vector $\mathbf{p}_r$, and is referred to as the cluster *core*. The set of cluster cores corresponding to the set of clusters at node n_i is $\mathbf{P}_i = \{\mathbf{p}_r\}$, and will be referred to as the *node summary*. The keyphrases in each cluster core are the top k keyphrases extracted from the cluster using the algorithm described in Section 4.2.

3.4 Document Data Model Document features are represented as sentences rather than individual words, as is the case in most text mining approaches. The rationale for choosing phrase-based modeling is the inherent tendency of this model to make keyphrase extraction from document clusters straightforward (see section 4.2 for details).

The model assumes that the constituents of a document is a set of sentences, which in turn are composed of a set of terms. A document is represented as a vector of sentences, each of which is a vector of consecutive terms:

$$\mathbf{d} = \{\mathbf{s}_i\}, \quad (3.1a)$$

$$\mathbf{s}_i = (\mathbf{t}_i, w_i, l_i), \quad (3.1b)$$

$$\mathbf{t}_i = \{t_{ij}\}, \quad (3.1c)$$

$$l_i = |\mathbf{t}_i| \quad (3.1d)$$

$\mathbf{s}_i$: is *sentence* i in the document,
$\mathbf{t}_i$: is the *term* vector of sentence $\mathbf{s}_i$,
t_{ij}: is the *term* j in sentence $\mathbf{s}_i$,
w_i: is the *weight* associated with sentence $\mathbf{s}_i$, and
l_i: is the *length* of sentence $\mathbf{s}_i$.

Sentence weights are assigned according to their significance in the document (*e.g.* title, headings, *etc.*). A standard pre-processing is applied to the documents, in which stop words are removed, and different forms

[3] The term *topic* is used to refer to the document class to avoid confusion in mathematical notation.

of the same word are removed through stemming using the standard Porter algorithm [16].

4 Collaborative Document Clustering Algorithm

4.1 Initial Cluster Generation Each node starts by forming an local clustering solution, to be enriched later by collaborative clustering. A number of traditional clustering solutions were investigated and achieved comparable results. These included hierarchical, single-pass, k-nearest neighbor, and similarity histogram-based clustering (SHC) [9]. Hierarchical clustering usually performed slightly better than the other three, but since the collaborative clustering algorithm is a collaborative derivative of SHC, we chose it to do the initial clustering for the sake of consistency. The decision was also largely based on the fact that we are more interested in the improvement observed after merging peer recommendations to the local clustering solution, rather than in the quality of the initial clustering itself.

In SHC, the coherency of a cluster is represented as a **Cluster Similarity Histogram**.

> *Cluster Similarity Histogram*: A concise statistical representation of the set of pairwise document similarities distribution in the cluster. A number of *bins* in the histogram correspond to fixed similarity value intervals. Each bin contains the count of pair-wise document similarities in the corresponding interval.

A coherent cluster should have high pairwise document similarities. Figure 2 shows a typical cluster similarity histogram, where the distribution is almost a normal distribution. A perfect cluster would have a histogram where the similarities are all maximum, while a loose cluster would have a histogram where the similarities are all minimum.

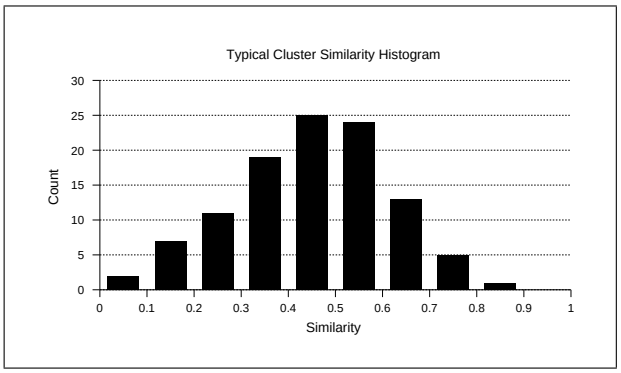

Figure 2: Cluster Similarity Histogram

We judge the quality of a similarity histogram (cluster cohesiveness) by calculating the ratio of the count of similarities above a certain similarity threshold R_T to the total count of similarities. The higher this ratio, the more cohesive is the cluster. Let N_{D_c} be the number of the documents in a cluster. The number of pair-wise similarities in the cluster is $N_{R_c} = N_{D_c}(N_{D_c}+1)/2$. Let $\mathbf{R} = \{r_i : i = 1, \ldots, N_{R_c}\}$ be the set of similarities in the cluster. The histogram of the similarities in the cluster is represented as:

$$\mathbf{H}_c = \{h_i : i = 1, \ldots, B\} \quad (4.2a)$$

$$h_i = \text{count}(r_k) \qquad r_{li} \le r_k < r_{ui} \quad (4.2b)$$

where B: the number of histogram bins,
h_i: the count of similarities in bin i,
r_{li}: the lower similarity bound of bin i, and
r_{ui}: the upper similarity bound of bin i.

The histogram ratio (*HR*) of a cluster is the measure of cohesiveness of the cluster as described above, and is calculated as:

$$HR(\mathbf{c}) = \frac{\sum_{i=T}^{B} h_i}{\sum_{j=1}^{B} h_j} \quad (4.3a)$$

$$T = \lfloor R_T \cdot B \rfloor \quad (4.3b)$$

where $HR(\mathbf{c})$: the histogram ratio of cluster $\mathbf{c}$,
R_T: the similarity threshold, and
T: the bin number corresponding to the similarity threshold.

The algorithm works by maintaining high *HR* for each cluster. New documents are tested against each cluster, adding them to appropriate clusters if they do not degrade the *HR* of that cluster significantly. Provisions are also made so as not to allow a chain reaction of "bad" documents being added to the same cluster, thus bringing its cohesiveness down significantly.

The algorithm works incrementally by receiving a new document, and for each cluster calculates the cluster histogram before and after simulating the addition of the document (lines 4-6). The old and new histogram ratios are compared and if the new ratio is greater than or equal to the old one, the document is added to the cluster. If the new ratio is less than the old one by no more than ε and still above $HR_{\min}$, it is added (lines 7-9). Otherwise it is not added. If after checking all clusters the document was not assigned to any cluster, a new cluster is created and the document is added to it (lines 11-15).

4.2 Cluster Summarization Using Keyphrase Extraction Once the initial clusters have been formed at each node, a process of cluster summarization is conducted. The summary of each cluster is represented

Algorithm 1 Similarity Histogram-based Incremental Document Clustering

```
1: C_i ← Empty List {Cluster List of Node i}
2: for each document d_k ∈ D_i do
3:    for each cluster c_r ∈ C_i do
4:       HR_old = HR(c_r)
5:       Simulate adding d_k to c_r
6:       HR_new = HR(c_r)
7:       if (HR_new ≥ HR_old) OR ((HR_new > HR_min) AND
         (HR_old − HR_new < ε)) then
8:          m_{k,r} ← 1 {Add d_k to c_r}
9:       end if
10:   end for
11:   if [M ↓ C_i]_k = 0 {d_k was not added to any cluster}
      then
12:      Create a new cluster c_new
13:      C_i ← {C_i, c_new} {Add c_new to C_i}
14:      m_{k,new} ← 1 {Add d_k to c_new}
15:   end if
16: end for
```

as a set of core keyphrases that accurately describe the topic of the cluster. We use the **CorePhrase** keyphrase extraction algorithm [10], as opposed to traditional keyword-based cluster summarization, since it produces more accurate representation of clusters. CorePhrase works by first constructing a list of candidate keyphrases for each cluster, scoring each candidate keyphrase according to its features, ranking the keyphrases by score, and finally selecting a number of the top ranking keyphrases for output.

Extraction of Candidate Keyphrases. Candidate keyphrases naturally lie at the *intersection* of the document cluster. The CorePhrase algorithm compares every pair of documents to extract matching phrases. This process of matching every pair of documents is inherently $O(n^2)$. However, by using a proven method of document phrase indexing graph structure, known as the Document Index Graph (DIG), the algorithm can achieve this goal in near-linear time [9].

In essence, what the DIG model does is to keep a cumulative graph representing currently processed documents: $G_i = G_{i-1} \cup g_i$, where g_i is the subgraph representation of a new document. Upon introducing a new document, its subgraph is matched with the existing cumulative graph to extract the matching phrases between the new document and all previous documents. That is, the list of matching phrases between document $\mathbf{d}_i$ and previous documents is given by $M_i = g_i \cap G_{i-1}$. The graph maintains complete phrase structure identifying the containing document and phrase location, so cycles can be uniquely identified. This process produces complete phrase-matching output between every

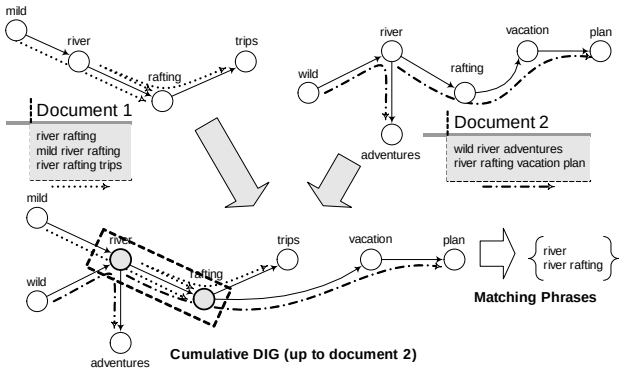

Figure 3: Phrase Matching Using Document Index Graph

pair of documents in near-linear time, with arbitrary length phrases. Figure 3 illustrates the process of phrase matching between two documents. In the figure, the two subgraphs of two documents are matched to get the list of phrases shared between them.

Phrase Features. Quantitative features are needed to judge the quality of the candidate keyphrases. Each candidate keyphrase p is assigned the following features:

df**: document frequency**; the number of documents in which the phrase appeared, normalized by the total number of documents.

$$df = \frac{\mid \text{documents containing } p \mid}{\mid \text{all documents} \mid}$$

w**:** average **weight**; the average weight of the phrase over all documents. The weight of a phrase in a document is calculated using structural text cues. Examples: title phrases have maximum weight, section headings are weighted less, while body text is weighted lowest.

pf**:** average **phrase frequency**; the average number of times this phrase has appeared in one document, normalized by the length of the document in words.

$$pf = \arg avg \left[\frac{\mid \text{occurrences of } p \mid}{\mid \text{words in document} \mid} \right]$$

d**:** average phrase **depth**; the location of the first occurrence of the phrase in the document.

$$d = \arg avg \left[1 - \frac{\mid \text{words before first occurrence} \mid}{\mid \text{words in document} \mid} \right]$$

Phrase Ranking. Phrase features are used to calculate a *score* for each phrase. Phrases are then

ranked by score, and a number of the top phrases are selected as the ones describing the cluster topic. The score of each phrase p is:

$$\text{score}(p) = (\sqrt{w \cdot pf} \cdot d^2) \times -\log(1 - df) \tag{4.4}$$

The equation is derived from the tf×idf term weighting measure; however, we are rewarding phrases that appear in more documents (high *df*) rather than punishing those phrases. By examining the distribution of the values of each feature in a typical corpus, it was found that the *weight* and *frequency* features usually have low values compared to the *depth* feature. To take this fact into account, it was necessary to "expand" the *weight* and *frequency* features by taking their square root, and to "compact" the *depth* by squaring it. This helps even out the feature distributions and prevents one feature from dominating the score equation.

Word weight-based score assignment. Another method for scoring phrases was used, which is based on individual word weights. This method will be referred to as CorePhrase-M:

- First, assign *initial* scores to each phrase based on phrase scoring formulas given above.
- Construct a list of unique individual words out of the candidate phrases.
- For each word: add up all the scores of the phrases in which this word appeared to create a word weight.
- For each phrase: assign the *final* phrase score by adding the individual word weights of the constituent words and average them.

4.3 Collaborative Document Clustering At this stage, each node i has an initial local clustering solution $\mathbf{C}_i$, and cluster summary information in the form of the core kephrases of each cluster $\mathbf{P}_i$, which we will refer to as *cluster cores*.

Algorithms 2 and 3 describe the collaborative document clustering process. Algorithm 2 recommends documents to peers based on the received peer cluster summaries. It starts by exchanging cluster cores between peers. Each node receives N_{N-1} peer summaries; each peer summary is a list of cluster cores $\mathbf{P}_j$ at peer n_j. The receiving node compares the cluster cores to its own documents and builds a similarity matrix $\mathcal{S}$ between its documents and peer cores. The node then offers to each peer those documents that are most similar (above similarity threshold R_T) to the peer core summaries; we call these documents *peer-positive* documents.

Algorithm 2 Recommend to Peers

for each peer n_j, $j = 1, \ldots, N_N$, $j \neq i$ **do**
Require: $\mathbf{C}_j$ and $\mathbf{P}_j$ have been calculated at peer n_j
  Receive peer cluster cores $\mathbf{P}_j$
  $\mathcal{S} = R(\mathbf{D}_i, \mathbf{P}_j) \leftarrow \emptyset$ {Similarity matrix between own docs and peer cores}
  for each document $\mathbf{d}_k \in \mathbf{D}_i$ and cluster core $\mathbf{p}_r \in \mathbf{P}_j$ **do**
    $\mathcal{S}(\mathbf{d}_k, \mathbf{p}_r) \leftarrow (\mathbf{s}_k \cap \mathbf{p}_r)/(\mathbf{s}_k \cup \mathbf{p}_r)$
  end for
  for each cluster core $\mathbf{p}_r \in \mathbf{P}_j$ **do**
    $\mathbf{D}^+_{i,r} \leftarrow \{\mathbf{d}_k \in \mathbf{D}_i \mid \mathcal{S}(\mathbf{d}_k, \mathbf{p}_r) > R_T\}$
    Send $\mathbf{D}^+_{i,r}$ to peer n_j
  end for
end for

Algorithm 3 Merge Peer Recommendation

for each peer n_j, $j = 1, \ldots, N_N$, $j \neq i$ **do**
  Receive recommendations $\{\mathbf{D}^+_j\}$ from peer n_j
  for each recommendation $\mathbf{D}^+_{j,r} \in \{\mathbf{D}^+_j\}$ **do**
    for each document $\mathbf{d}_k \in \mathbf{D}^+_{j,r}$ **do**
      {Skip documents that already belong to this node}
      if $\mathbf{d}_k \ni \mathbf{D}_i$ **then**
        $\mathbf{c}_{ir} \leftarrow$ cluster corresponding to recommendation $\mathbf{D}^+_{j,r}$
        $HR_{\text{old}} = HR(\mathbf{c}_{ir})$
        Simulate adding $\mathbf{d}_k$ to $\mathbf{c}_{ir}$
        $HR_{\text{new}} = HR(\mathbf{c}_{ir})$
        if ($HR_{\text{new}} \geq HR_{\text{old}}$) OR (($HR_{\text{new}} > HR_{\text{min}}$) AND ($HR_{\text{old}} - HR_{\text{new}} < \varepsilon$)) **then**
          $m_{k,ir} \leftarrow 1$ {Add $\mathbf{d}_k$ to $\mathbf{c}_{ir}$}
        end if
      end if
    end for
  end for
end for

Algorithm 3 merges the recommendations of each peer. It starts by receiving recommended documents from each peer. It then follows the same logic of the SHC clustering algorithm described in section 4.1, except that it does not create new clusters for documents that were not assigned to any cluster.

This process is expected to enhance the clustering solution for peer nodes, by allowing nodes to receive documents that expand their clustering solution to encompass data that was previously unavailable, while at the same time maintaining their clustering solution structure and coherency.

5 Experimental Results

5.1 Experimental Setup The collaborative document clustering system presented here was evaluated

on a number of document data sets. We evaluated two aspects of the system: keyphrase extraction accuracy and collaborative clustering accuracy improvements.

Three data sets have been used in the evaluation, which are listed in Table 2. The first data set is a collection of web pages from the University of Waterloo intranet web sites. The second data set is a collection of web pages retrieved through the Google search engine by submitting six different queries about Canada, and collecting the top 20-30 results. The first and second data sets have moderate degree of overlap between the different categories, and was used by [9]. Results on the accuracy of keyphrase extraction were obtained from those two datasets, and can be found in [10]. The third data set is a collection of 2340 Yahoo! news articles, and was used in [2, 4, 3]. The categories of the data set come from the Yahoo categorization of Reuters news feed, and the category distribution is rather unbalanced. The overlap between classes in this data set is quite low.

5.2 Evaluation Measures The metrics used for evaluation are extrinsic measures that rely on labeled data. For evaluation of clustering results, we use *F-measure* and *Entropy*.

F-measure combines the *Precision* and *Recall* ideas from the Information Retrieval literature. The precision and recall of a cluster j with respect to a class i are defined as:

$$P = \text{Precision}(i,j) = \frac{N_{ij}}{N_j} \tag{5.5a}$$

$$R = \text{Recall}(i,j) = \frac{N_{ij}}{N_i} \tag{5.5b}$$

where N_{ij}: is the number of members of class i in cluster j,
N_j: is the number of members of cluster j, and
N_i: is the number of members of class i.

The F-measure of a class i is defined as:

$$F(i) = \frac{2PR}{P+R} \tag{5.6}$$

With respect to class i we consider the cluster with the highest F-measure to be the cluster j that maps to class i, and that F-measure becomes the score for class i. The overall F-measure for the clustering result $\mathbf{C}$ is the weighted average of the F-measure for each class i:

$$F_{\mathbf{C}} = \frac{\sum_i (|i| \times F(i))}{\sum_i |i|} \tag{5.7}$$

where $|i|$ is the number of objects in class i. The higher the overall F-measure, the better the clustering, due to the higher accuracy of the clusters mapping to the original classes.

Entropy provides a measure of "goodness" for unnested clusters. Entropy tells us how homogeneous a cluster is. The higher the homogeneity of a cluster, the lower the entropy is, and vice versa. The entropy of a cluster containing only one object (perfect homogeneity) is zero.

For every cluster j in the clustering result $\mathbf{C}$ we compute p_{ij}, the probability that a member of cluster j belongs to class i. The entropy of each cluster j is calculated using the standard formula $E_j = -\sum_i p_{ij} \log(p_{ij})$, where the sum is taken over all classes. The total entropy for a set of clusters is calculated as the sum of entropies for each cluster weighted by the size of each cluster:

$$E_{\mathbf{C}} = \sum_{j=1}^{N_C} (\frac{N_j}{N} \times E_j) \tag{5.8}$$

where N_j is the size of cluster j, and N is the total number of data objects.

To achieve better clustering, we would like to increase the F-measure and decrease the Entropy.

5.3 Collaborative Document Clustering Results Table 3 shows the results obtained for three network configurations: 3-Node, 5-Node, and 7-Node. The results are obtained through experiments on the third data set, DS3, due to its larger size. For each configuration the results of the initial and final (after aggregating peer recommendations) F-measure (F) and Entropy (E) per node are reported, along with averages over the nodes. The difference between the final and initial measure values is reported as the improvement in accuracy. The results reported were obtained using a data distribution ratio $\alpha = 1/N_N$; *i.e.* the data was equally partitioned over all nodes. In the 3-Node case, each node received 33% of the data; in the 5-Node case, each node received 20% of the data; and in the 7-Node case, each node received 14% of the data. In each case, there was no overlap between the nodes in terms of their local documents. Figure 4 illustrates the same results (F-measure only), showing initial F-measure (solid bars) and final F-measure (difference bars).

It should be noted that the evaluation of the final clustering after aggregation takes into consideration that the initial document class distribution at the local node has been changed, and updates the final number of documents in each class. This is essential for the initial and final F-measure and Entropy calculations to be correct.

Table 2: Data Sets

Data Set	Name	Type	# Docs.	Categories	Avg. words/doc
DS1	UW	HTML	162	4	432
DS2	Canada	HTML	152	6	506
DS3	Yahoo! news	HTML	2340	20	289

Table 3: Collaborative Document Clustering Results

	3-Node				**5-Node**				**7-Node**			
	Initial		Final		Initial		Final		Initial		Final	
Node	F	E	F	E	F	E	F	E	F	E	F	E
1	0.59	0.21	0.66	0.19	0.50	0.24	0.56	0.20	0.35	0.34	0.51	0.26
2	0.61	0.17	0.67	0.16	0.43	0.29	0.51	0.22	0.31	0.30	0.42	0.25
3	0.57	0.19	0.62	0.15	0.47	0.25	0.52	0.19	0.28	0.30	0.40	0.21
4					0.42	0.31	0.54	0.26	0.26	0.37	0.39	0.28
5					0.40	0.26	0.61	0.21	0.30	0.28	0.43	0.19
6									0.32	0.31	0.52	0.23
7									0.28	0.42	0.48	0.20
Average	0.59	0.19	0.65	0.17	0.44	0.27	0.55	0.22	0.30	0.33	0.45	0.23
Imrpovement			+6%	-2%			+11%	-5%			+15%	-10%

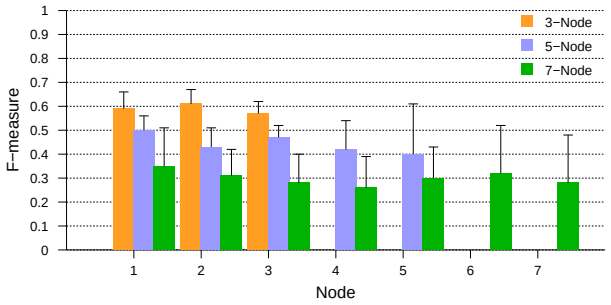

Figure 4: Collaborative Clustering Improvement – F-measure

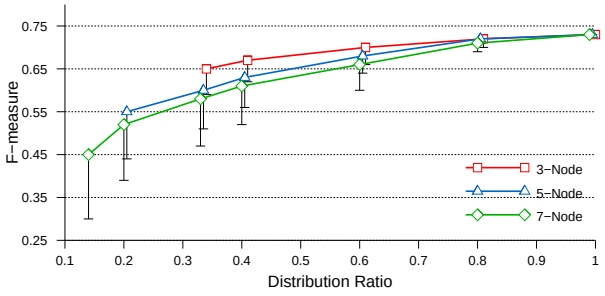

Figure 5: Effect of Data Distribution Ratio

The results are interesting, and show an improvement in both evaluation measures per node and on average. A very important observation is that networks with fewer nodes appear to have higher absolute accuracy (both initial and final) compared to networks with larger number of nodes; *e.g.* the 3-Node network has initial F of 59% and final of 65%, as opposed to 30% and 45% in the 7-Node network. However, networks with larger number of nodes exhibit greater improvement in the final clustering compared to the initial clustering, as can be seen from the improvement percentages; *e.g.* the 7-Node network has an improvement of 15% in F compared to just 6% improvement in the 3-Node network.

This observation is attributed to the fact that each node in networks with fewer nodes has access to larger percentage of the data than those in larger networks. This results in smaller networks being able to build better clusters. The ideal case is when there is only one node, in which case the one and only node has complete knowledge about all the data and thus can build the best clusters. However, the real benefit of the system is observed in situations involving larger networks (typical case). Each node in larger networks has limited view of the global data collection, but through the collaborative clustering algorithm they are able to have better visibility of the global data, and get recommendations from their peers that benefit their clustering solution, and thus results in higher improvements.

We have conducted experiments to investigate the effect of increasing the data distribution ratio α on the clustering quality. Figure 5 illustrates this trend. The line graphs in the figure represent the final F-measure of each network configuration for various distribution ratios. The vertical error bars show the difference be-

tween the initial and final clustering F-measure quality. The Entropy has a similar trend, and thus is not shown.

As noted earlier, the figure confirms that networks with fewer nodes (*e.g.* 3-Node) maintain higher accuracy for the range of distribution ratios. However, the figure also confirms our observation that networks with more nodes exhibit greater increase in clustering quality after aggregating peer recommendations. It is also interesting to note that as the distribution ratio reaches its maximum (all nodes having access to 100% of the data), all networks tend to have the same (maximum) performance. This is logical since at this maximum distribution ratio ($\alpha = 1$) each node has access to the full data set, and produces the same clustering solution, and any peer recommendation would be superfluous.

Notice also that at the same distribution ratio, networks with higher number of nodes result in higher improvements in the final clustering result quality than those with fewer number of nodes. This is attributed to the larger coverage of data found in networks with higher number of nodes, thus peer nodes are able to recommend more and better documents, which nodes in smaller networks cannot recommend because they may not have access to enough data. Consider for example the 5-node network with $\alpha = 40\%$ versus the 3-node network with the same α. In the 5-node case, each node receives recommendation from 4 peers each covering 40% of the data. In the 3-node case, it receives recommendation from 2 peers each covering 40% of the data. There is a higher chance in the 5-node case that the peers (collectively) will recommend documents that the peers in the 3-node case do not have access to.

6 Conclusion and Future Work

We have introduced a collaborative distributed approach for document clustering based on cluster keyphrase summarization. The major contribution of this work lies in the peer-to-peer document clustering algorithm based on exchanging cluster summaries between peers, and the recommendation of documents to peers to enhance their local clustering quality. The collaborative approach proved to enrich the local clustering solutions by making available data that is otherwise hard to access, or in situations when no global solution is required but rather an enrichment of local ones.

The results show significant improvement in the final clustering quality after merging peer recommendations over the initial clustering solutions. The algorithm exhibits better improvements in final clustering in networks with larger number of nodes, by allowing nodes to expose more useful data to their peers, and thus enhancing the global view of data of each node.

Currently each node receives cluster summaries from peer nodes, and after comparing the summary of a certain peer with its own clusters it recommends documents for that peer. It would be interesting to let each node *combine* its view of all peer summaries, then decide which peers can best benefit from which documents.

Finally, it would be interesting to achieve more fine-grained summary exchange, by letting nodes summarize documents rather clusters, then utilizing such summarized documents to further enhance the clustering process through exchanging document summaries.

References

[1] R. Barzilay, K. R. McKeown, and M. Elhadad. Information fusion in the context of multi-document summarization. In *Proceedings of the 37th conference on Association for Computational Linguistics*, pages 550–557, Morristown, NJ, USA, 1999. Association for Computational Linguistics.

[2] D. Boley. Principal direction divisive partitioning. *Data Mining and Knowledge Discovery*, 2(4):325–344, 1998.

[3] D. Boley, M. Gini, R. Gross, S. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. Document categorization and query generation on the World Wide Web using WebACE. *AI Review*, 13(5-6):365–391, 1999.

[4] D. Boley, M. Gini, R. Gross, S. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. Partitioning-based clustering for web document categorization. *Decision Support Systems*, 27:329–341, 1999.

[5] S. Chakrabarti. *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan Kaufmann Publishers, 2003.

[6] C. Clifton, R. Cooley, and J. Rennie. TopCat: data mining for topic identification in a text corpus. *IEEE Transactions on Knowledge and Data Engineering*, 16(8):949–964, August 2004.

[7] M. Eisenhardt, W. Muller, and A. Henrich. Classifying documents by distributed p2p clustering. In *Informatik 2003: Innovative Information Technology Uses*, Frankfurt, Germany, 2003.

[8] K. Hammouda and M. Kamel. Document similarity using a phrase indexing graph model. *Knowledge and Information Systems*, 6(6):710–727, November 2004.

[9] K. Hammouda and M. Kamel. Efficient phrase-based document indexing for web document clustering. *IEEE Transactions on Knowledge and Data Engineering*, 16(10):1279–1296, October 2004.

[10] K. Hammouda and M. Kamel. CorePhrase: Keyphrase extraction for document clustering. In *IAPR 4th International Conference on Machine Learning and Data Mining (MLDM'2005)*, pp. 265-274, Leipzig, Germany, July 2005.

[11] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, Englewood Cliffs, N.J., 1988.

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

[13] H. Kargupta, I. Hamzaoglu, and B. Stafford. Distributed data mining using an agent based architecture. In *Proceedings of Knowledge Discovery and Data Mining*, pages 211–214. AAAI Press, 1997.

[14] R. Kosala and H. Blockeel. Web mining research: a survey. *ACM SIGKDD Explorations Newsletter*, 2(1):1–15, 2000.

[15] J. Li and R. Morris. Document clustering for distributed fulltext search. In *2nd MIT Student Oxygen Workshop*, Cambridge, MA, August 2002.

[16] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, July 1980.

[17] G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison Wesley, Reading, MA, 1989.

[18] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill computer science series. McGraw-Hill, New York, 1983.

[19] G. Salton, A. Wong, and C. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, November 1975.

[20] P. D. Turney. Learning algorithms for keyphrase extraction. *Information Retrieval*, 2(4):303–336, 2000.