The Role of Semantic Locality in Hierarchical Distributed Dynamic Indexing

Fabien D. Bouskila and William M. Pottenger

Fabien D. Bouskila
National Center for Supercomputing Applications
fbouskil@ncsa.uiuc.edu

William M. Pottenger, Ph.D.
Lehigh University and NCSA
Bethlehem, PA 18015
610-758-3454, 610-758-6279 (fax)
billp@{eecs.lehigh.edu,ncsa.uiuc.edu}

Keywords: information retrieval, machine learning, HPC, knowledge management, HDDI

Abstract

The global growth in popularity of the World Wide Web has been enabled in part by the availability of browser-based search tools, which in turn have led to an increased demand for indexing techniques and technologies. This explosive growth is evidenced by the rapid expansion in the number and size of digital collections of documents. Simultaneously, fully automatic content-based techniques of indexing have been under development at a number of institutions. The time is thus ripe for the development of scalable knowledge management systems capable of handling extremely large textual collections distributed across multiple repositories.

This paper introduces a technique for identifying regions of semantic locality *within a collection of documents represented as a semantic network. Semantic locality imbues semantics to statistically based measures of graph connectedness. Such semantics become the basis for several applications in scalable knowledge management.*

1 Introduction

The explosive growth of digital repositories of information has been enabled by recent developments in communication and information technologies. The global Internet/World Wide Web exemplifies the rapid deployment of such technologies. Despite significant accomplishments in internetworking, however, scalable indexing techniques for distributed information lag behind the rapid growth of digital collections.

In the 21^{st} century, a significant amount of information useful in the practice of science will be available via such computer communications networks. The appearance of focused digital libraries on the World Wide Web demonstrates the willingness of scientists and engineers to distribute detailed information beyond that traditionally available in the published literature (e.g., [2]). It is critical that new information infrastructure be developed that enables effective search in the huge volume of distributed information emerging in digital form.

Traditional methods of indexing combine multiple subject areas into a single, monolithic index. There are already enough documents on the Web that such indexing technology is often failing to perform effective search. The difficulty lies in the fact that since so many documents and subjects are being combined together, retrieving all the documents that match a particular word phrase often returns too many documents for effective search. This problem has been known for some time [3].

In order to properly address this problem, a paradigm shift is needed in the approach to indexing. First and foremost, it is clear that digital collections are now and will continue to be distributed. Our first premise is thus that *indexes* must also be distributed.

Secondly, it must be realized that the information contained in these distributed digital repositories is

hierarchical in nature[1]. Traditionally, knowledge hierarchies, or ontologies, have been created with human expertise[2]. Such an approach does not scale to the tremendous amount of emerging digital information for two reasons: as knowledge increases, new topics are emerging at a greater rate, and both this and the sheer volume of information preclude manual approaches to indexing. Our second premise is thus that distributed indexes must properly reflect the hierarchical nature of knowledge.

Thirdly, due to the vast increase in communications bandwidth and computing and online storage capabilities mentioned above, digital collections are frequently updated. This process reflects a key characteristic of 21^{st} century collections: namely, they are dynamic in nature. Our third premise is thus that any new information infrastructure must include *dynamic* indexing capabilities.

In the final analysis, these three technologies must be integrated into a cohesive whole. The goal of our research is thus to architect a knowledge management prototype based on HDDI: *Hierarchical Distributed Dynamic Indexing*.

As the HDDI technology unfolds, we are discovering novel approaches to addressing the various issues of managing distributed digital information in the context of the aforementioned paradigm shift. This paper introduces a technique for identifying regions of *semantic locality* within a collection of documents represented as a semantic network. Semantic locality imbues semantics to statistically based measures of graph connectedness. Such semantics become the basis for applications employing HDDI technologies.

In the following sections we review the related work in graph partitioning, discuss sLoc, our algorithm for determining regions of semantic locality, and present conclusions about the applicability of using sLoc to build a hierarchical distributed dynamic index.

2 Background and Related Work

Central to the construction of a hierarchical distributed dynamic index is the formation of a knowledge base. A knowledge base is a graph representing the concepts (noun phrases) in a single node of an HDDI linked with arcs of which the weights represent the similarities between concepts. The resulting weight assignments from knowledge base creation are context-sensitive, and are used to determine regions of semantic locality (i.e., conceptual density) within each node of the HDDI hierarchy. During this phase focused clusters of concepts within each knowledge base in the hierarchy are detected. The result is a hierarchy of knowledge bases composed of regions of high-density clusters of concepts – subtopic regions of semantic locality. In simple terms, these regions consist of clusters of concepts commonly used together that collectively create a *knowledge neighborhood*.

The motivation for the use of *semantic locality* derives from the premise that grouping similar concepts leads to increased effectiveness and efficiency in particular in query search and retrieval (see Sparck Jones [10]). Also, in the perspective of the HDDI hierarchy, semantic locality enables *tracking* of similar topics across the hierarchy.

In this section we review the traditional clustering methods used in information retrieval, their purpose, advantages and drawbacks. We then present other algorithms used in graph partitioning and see why these were not applicable for the particular case sLoc addresses. We might note here that classification and clustering do not usually refer to the same thing. Classification tends to consider classes that are either given or computed at the beginning, and tries to classify new objects into these existing classes. Clustering is generally thought of as a dynamic process that creates and modifies the classes with respect to the objects it wants to classify. Clustering and classification will however be used synonymously in this article.

2.1 The use of clustering in information retrieval

In information retrieval the two most used classification schemes are: *term* classification and *document* classification. Notice that Sparck Jones's *keyword* classification [10], or HDDI's *concept* classification refer to the same idea: clustering terms rather than documents. Following Salton's description in [8]:

Term classification is designed to group terms into (synonym) classes in the hope of producing greater matching capacities between queries and document terms.

Document classification groups documents together to improve recall, but also to increase the response time of the retrieval system.

The two classifications are not independent because the terms assigned to the documents must necessarily form the basis for the classes produced by a document grouping procedure.

Term classification groups related low-frequency terms into common thesaurus classes (clusters, or regions of semantic locality). The terms included in a

[1] This point was made recently by NSF CISE Assistant Director Ruzena Bajcsy.

[2] One popular form is the thesaurus (e.g., the National Library of Medicine's MeSH thesaurus).

common class can then be substituted for each other in retrieval, and the recall output produced may be improved in this way.

Document classifications make it possible to restrict the searches to the most interesting document classes only, thereby providing high-precision output [8].

2.1.1 The cluster hypothesis

Any sort of clustering in information retrieval (document, or keyword clustering) is based on the assumption that *closely associated documents (respectively keywords, or concepts in the HDDI terminology) tend to be relevant to the same request.* Van Rijsbergen [4] refers to this assumption as the *cluster hypothesis.*

2.1.2 Criteria of a good clustering

The first property we expect from a clustering method is that it provides a set of clusters that actually fit the constraints and requirements of our application semantics. However, additional properties are needed in order for a given method to scale to multiple types of input sets. Jardine and Sibson [27] stated the major requirements of a good clustering method:

1. The classification should be defined such that a given result is obtained for any given body of data.
2. The classification should be independent of scale because a multiplication by a constant of the property values identifying the objects should not affect the classification.
3. Small errors in the description of the objects must not lead to big changes in the clustering.
4. The method must be independent of the initial ordering of the objects.
5. Objects exhibiting strong similarities should not be split by being assigned to separate classes.

We argue that although true in the 1960s, these requirements may have to be reviewed in the context of the 1990s where the amount of data increases on a daily basis. The fact that the amount of data increases every day invalidates the need for the classification to be independent of scale; we rather want the classification to adapt to the scale, that is to create new classes or delete some to work best with the actual size of the current collection.

At the same time, another essential criterion in the assessment of a clustering method is the *efficiency* achieved by the algorithm. As stated in [1] efficiency is usually measured in terms of the computer resources used such as memory and storage requirements and CPU time.

Over the past 40 years, various methods have been used to achieve good efficiency and effectiveness. In 1979, van Rijsbergen identified two major approaches [4] to information clustering and they are still good starting points for research in IR today:

1. The clustering is based on a measure of similarity between the objects to be clustered. This typically refers to the work of Sparck Jones [10] in keyword clustering.
2. The cluster method proceeds directly from the object descriptions. See Rocchio's algorithm [18] for an example of this method.

2.2 Existing algorithms in graph partitioning

In our effort to find an algorithm for clustering knowledge bases, we also wanted to consider alternative approaches to classical IR clustering algorithms. In one sense, clustering a knowledge base can be cast as a graph partitioning problem; we therefore looked at the different problems that current graph partitioning techniques can help to solve, and evaluated the use of these techniques in HDDI.

Graph partitioning is an NP-hard problem with numerous applications. It appears in various forms in parallel computing, sparse matrix reordering, circuit placement and other important disciplines. Several advanced software solutions are available from the internet. The most well-known are Chaco [28] and Metis [29].

Traditional graph partitioning algorithms compute a k-way partitioning of a graph such that the number of edges that are cut by the partitioning is minimized and each partition has an equal number of vertices. The task of minimizing the edge-cut can be considered as the objective and the requirement that the partitions be of the same size can be considered as the constraint.

Much of the current research in graph partitioning finds its application in parallel computing and load balancing. Therefore, the focus is mostly on getting a fixed number of clusters (that may correspond to the number of processors for example) and an equal number of vertices, in a nonoriented graph.

Hinrich Schütze uses clustering for context-group discrimination and disambiguation in computational linguistics [34]. In this study, similarity in word

space is based on *second order co-occurrence*. Quoting Schütze:

> Words, contexts and clusters are represented in a high-dimensional real-valued vector space. Context vectors capture the information present in *second-order co-occurrence*. Instead of forming a context representation from the words that the ambiguous word directly occurs with in a particular context (first-order co-occurrence), we form the context representation from the words that these words in turn co-occur with in the training corpus. Second-order co-occurrence information is less sparse and more robust than first-order information.

This tells us that, depending on the context, allowing some transitivity in the similarity relation can improve the results of clustering based on similarity measures.

In data mining applications, Han, Karypis and Kumar [35] propose a clustering method based on hypergraphs that is expected to handle large input data sets better than the traditional k-means [36] or Autoclass [37]. The idea of the hypergraph model is to map the relationship of the original data in high dimensional space into a hypergraph. The hypergraph model is therefore based on the concept of a hyperedge. Quoting Kumar et al. [35]:

> A hyperedge represents a relationship (affinity) among *subsets* of data and the weight of the hyperedge reflects the strength of this affinity. A hypergraph partitioning algorithm is used to find a partitioning of the vertices such that the corresponding data items in each partition are highly related and the weight of the hyperedges cut by the partitioning is minimized.

[35] used a number of partitions that was set *a priori*, and a different measure of similarity, which do not fit our requirements for sLoc. However [35] does introduce *fitness* and *connectivity* measures, which are akin to metrics used in sLoc (see [1] for a detailed comparison of these approaches).

In the final analysis we determined that none of the available algorithms were suitable for our application. Thus, the decision was made to base development on an existing linear graph partitioning algorithm that could be readily adapted to our needs [38].

2.3 Test collections

In the following sections we will use various test collections to illustrate or validate our approach. Let me introduce them briefly here.

- Patterns 300 is a collection of 300 postings to a discussion list about computer software systems.
- MED, CISI and ADI are three collections of respectively 1033, 112 and 82 abstracts commonly employed in retrieval experiments. MED is a sample from MEDLINE, the National Library of Medicine's database of references to articles published in biomedical journals. The CISI and ADI collections both contain information science documents.
- The Grainger collection contains 60,000 abstracts from the Grainger Engineering Library at University of Illinois at Urbana-Champaign.

3 The Semantic Locality Finder: sLoc

In this section we discuss the details of the algorithm, system design and implementation of the Semantic Locality Finder (sLoc).

The function of sLoc is to identify regions of semantic locality. For now, we can qualitatively characterize a region of semantic locality as follows:

- Concepts inside a region of semantic locality are similar to each other.
- Concepts belonging to different regions of semantic locality are dissimilar.

Similarity between concepts is defined quantitatively in [25, 26]. It is a mapping from one concept to another that quantitatively determines how similar they are semantically. We term the resultant mapping a *knowledge base*.[3] A knowledge base is represented as an asymmetric directed graph in which nodes are concepts and arc weights are similarity measures. Determining regions of semantic locality in such a graph involves clustering or splitting the input knowledge base into a set of subtopic regions - smaller knowledge bases, if you will. These subtopic regions are regions of semantic locality or conceptual density. The number of concepts in each region of semantic locality may vary and the total number of regions is determined dynamically by sLoc. sLoc processes a knowledge base and outputs a *knowledge neighborhood* that consists of multiple regions of semantic locality.

3.1 Contextual transitivity in sLoc

The similarity relation is by definition not transitive. The theoretical basis for sLoc is the concept of what we term *contextual transitivity* in the similarity relation. In essence, this means that depending on the

[3] Note that follow-on work by Chen (of [25]) terms this a *concept space*.

context (structure and distribution of the similarities in the knowledge base), a threshold is decided upon and transitivity is constrained accordingly. Contextual transitivity extends Schütze's conceptualization of *second order co-occurrence* [34] by using n-order co-occurrence, where n varies with the underlying semantic structure of the model.

In order to group concepts together in a region of semantic locality, sLoc uses contextual transitivity. Here is a simple example. Let us consider the three concepts *Java*, *applet* and *e-commerce*. If the concepts occur in the same document, they are called co-occurring concepts. Let us assume that *Java* and *applet* co-occur and that *Java* and *e-commerce* also co-occur. This means that there is at least one document that contains both *Java* and *applet*, and that there is at least one document that contains both *Java* and *e-commerce*, but *applet* and *e-commerce* do not co-occur necessarily. However it may be natural to gather the three concepts in one unique class and extrapolate that *e-commerce* and *applet* are actually related. That is what contextual transitivity means.

Note that similarity is not a boolean relation for it can take on a range of values. Two concepts can thus be more or less similar. In sLoc, aspects of the knowledge base structure lead to a heuristic minimum similarity value for two concepts to be in the same cluster, and therefore to *constrain* transitivity. It is called *contextual* because this minimum similarity value depends on the structure and distribution of the similarities in the knowledge base.

3.2 Design

The core of sLoc is based on an algorithm due to Tarjan [38, 39]. Tarjan's algorithm uses a variant of depth-first search to determine the strongly connected components of a directed graph. This was the first algorithm to solve the problem in linear time. This is an important feature due to the fact that graph clustering is a NP-hard problem and the only heuristics we are aware of are not linear. The theory can be found in [39]. Figure 1 shows how Tarjan's algorithm identifies strongly connected regions (R_1, R_2, R_3), in a simple graph.

3.2.1 Notation

Before tackling the algorithm itself we must first introduce the following notation:

- Let $\mathcal{N}$ be the set of nodes i in the input graph, and let N be the total number of nodes.
- Let $\mathcal{A}$ be the set of arcs in the input graph, A the total number of arcs. An arc $a_{i,j} \in \mathcal{A}$ goes from node i to node j.
- Let $\mathcal{W}$ be the set of arc weights in the graph, $w_{i,j}$ is the weight of the arc going from node i to node j.

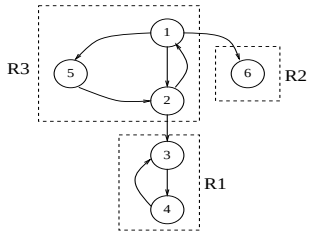

Figure 1: Strongly connected regions of a graph

Therefore $\mathcal{W} = \{w_{i,j}\}_{(i,j)\in\mathcal{N}^2}$. A knowledge base is an asymmetric graph and thus $w_{i,j}$ may differ from $w_{j,i}$. Moreover, if $a_{i,j} \notin \mathcal{A}$ then $w_{i,j} = 0$; in particular, for all i, $w_{i,i} = 0$. Now, let M be the mean of the arc weights:

$$M = \frac{1}{A} \sum_{(i,j)\in N^2} w_{i,j}$$

We term the standard deviation of the distribution of arc weights SD:

$$SD = \sqrt{\frac{1}{A-1} \sum_{(i,j)\in N^2} (w_{i,j} - M)^2}$$

3.2.2 The algorithm, step by step

Figure 2 depicts the three steps of the sLoc process. These steps are outlined in the following sections on the algorithm and its applications.

The first step in sLoc is to statistically "prune" the input graph before applying Tarjan's algorithm. Arcs of weight smaller than a certain *threshold* value τ are virtually pruned. This pruning and the process described in [4] are very much alike. The difference here is that similarities are asymmetric; that is, there can be potentially two arcs between nodes in a given pair. Therefore, the arc from concept a to concept b can be pruned while the arc back from b to a remains.

The second step involves the actual identification of the clusters within the graph. Tarjan's algorithm is applied to find strongly connected regions. At this stage each strongly connected region is a cluster. The size of a given cluster is the number of nodes (concepts) it contains.

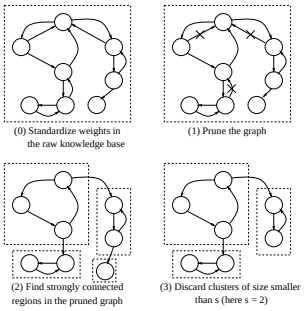

Figure 2: The sLoc process

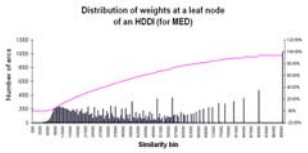

Figure 3: Distribution of weights at a leaf-level node

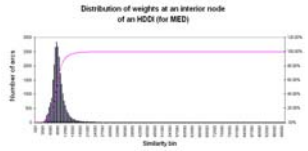

Figure 4: Distribution of weights in an interior node

During the third step, clusters of size smaller than parameter s are discarded (they are assumed to be outliers). We interpret the remaining clusters as regions of semantic locality in the knowledge base.

The greater τ, the more arcs are cut off, and therefore the smaller in size the strongly connected regions. Thus the greater τ the smaller in size and the more focused will be the regions of semantic locality.

Two questions have arisen in the course of our work: (1) Given an input knowledge base, what threshold τ should be applied? (2) Is the usability of a given clustering dependent on the application?

Let us tackle the first question on how to determine the value of τ yielding an optimum clustering. Our premise is that the optimum τ can be determined statistically as a function of the mean M, the standard deviation SD and other knowledge base dependent variables (e.g., size, maximum weight, etc.).

For leaf-level collections in a hierarchical distributed dynamic index, i.e., for those collections for which the knowledge base is directly computed from the co-occurrence frequencies in the collections, we use

$$\tau(\alpha) = \mathrm{Max}(\mathcal{W}) - \alpha \times SD \qquad (1)$$

In the equation above, τ is the cut-off weight used to prune the graph and α is a number of standard deviations. Thus, $\tau(\frac{1}{2})$ is the threshold corresponding to the maximum weight in the graph minus half of the standard deviation of the distribution of arc weights. [1] summarizes our results for a range of values of the parameter α. Figure 3 shows the distribution of arc weights at a leaf-level HDDI node, for the MED collection.

However, as we consider higher level nodes in the HDDI, the underlying distribution of weights changes to approximate more closely a normal distribution (see Figure 4). For the interior nodes, we use

$$\tau(\alpha) = M + 2SD - \alpha \times SD \qquad (2)$$

We did not factor SD in the equation above in order to emphasize that the maximum value τ can take is $M + 2SD$, the value below which 97.5% of a normal distribution occurs. It makes more sense to use this value here, rather than $\mathrm{Max}(\mathcal{W})$, because $\mathrm{Max}(\mathcal{W})$ does not make much statistical sense when the distribution of $\mathcal{W}$ is normal.

At this point in time we are still investigating answers to the second question posed above. Apparently, there is no ready-made definition. Our intuition tells us that the quality of a given clustering is related to the application: in other words, the *application* will likely play a key role in determining whether a given clustering is appropriate or not. If so, a given input knowledge base will have many optima, each one suited to a particular application (e.g., search engine, detection of emerging conceptual content, targeted marketing, etc.).

3.3 Clustering measures

In order to find the optimum where the threshold τ leads to satisfactory regions of semantic locality, we introduce some measures on the resulting knowledge neighborhood.

We distinguish between what we call "micrometrics" assessing the interior structure of the clustered graph and "macrometrics" assessing the knowledge neighborhood quality based on measures that are independent of its internal structure.

Macrometrics do not take into account the underlying graph structure. We can draw a parallel between these measures and the input parameters used in the

second approach to clustering described in Section 2.1. The main macrometrics are the number of clusters, the size of the clusters, and the coverage; they will be described later in this section.

Micrometrics give some feedback about the *internal structure* of the clustered graph; they measure the similarity of concepts in a cluster, and the dissimilarity of concepts in different clusters. However, these measures are very expensive computationally.

Moreover, the quality of these micrometrics is very dependent on the graph structure. In order to avoid getting a huge set of measures, one for each cluster for example, we want to use a "reduced" measure for each micrometric. The reduced measure represents the distribution of a given metric across the whole knowledge neighborhood. However, the less homogeneous the underlying graph, the less these measures make sense, because one average cannot represent the complexity of a heterogeneous distribution.

In defining the micrometrics, we take an approach similar to that of Sparck Jones [10] and compute measures similar to those outlined in further detail in the previous work section of [1]. The *intercluster* and *intracluster* densities for a given knowledge base are computed over all clusters remaining after the completion of phase 3 (see Figure 2).

Let us call $C_1, C_2 \ldots C_n$ the n clusters of nodes found by sLoc. Notice that, mathematically, $(C_1, C_2 \ldots C_n)$ is *not* a partition of $\mathcal{N}$: we must add all the clusters of size smaller than s to get a real partition of $\mathcal{N}$. $(C_1, C_2 \ldots C_n)$ only covers part of the input knowledge base; the clusters of size smaller than s, rather than being forced into a larger cluster, are simply discarded. Traditional clustering methods such as Rocchio's [18] try to force the outliers into existing clusters. In HDDI we want to leverage our knowledge of outliers and allow sLoc to play a role in "trimming" the knowledge base. Therefore, all the concepts in clusters of size smaller than s will be ignored for any further use at the leaf level of the hierarchy.

3.3.1 Coverage

At the end of the process depicted in Figure 2 clusters of size smaller than s are discarded. For large values of τ (respectively, small values of α) the number of nodes discarded can be fairly high. In other words, the ratio of the number of nodes in clusters of size greater than s to the total number of nodes present in the knowledge base originally is fairly low for large values of τ. We call the this latter ratio the coverage c in HDDI terminology. The coverage c is the percentage of the knowledge base actually represented in the regions of semantic locality sLoc outputs.

$$c = \frac{\sum_{k \in [1,n]} N_{C_k}}{N} \tag{3}$$

For example, a coverage of .9 means that 90% of the nodes in the input knowledge base will actually appear in a cluster of the knowledge neighborhood. The larger c the larger the number of concepts that will be retrievable through the knowledge neighborhood.

3.3.2 Number of clusters

The number of clusters n at the end of phase 3 (see Figure 2) is a good experimental indicator of distribution of clusters. We show in [1] how the quality of the clustering relates to the number of clusters. Notice that n not only depends on the threshold τ, but it is also determined by the minimum cluster size s that we impose. As for the other measures, the way s is set must reflect the application needs. For some applications, or some input sets $s = 2$ might be too small.

3.3.3 Cluster size

The distribution of cluster sizes is characteristic of the clustering too. In [1] we give an interpretation of how this distribution changes with the threshold τ.

3.3.4 Intracluster densities

The intracluster density $\rho(C_k)$ of a given cluster C_k can be easily computed from the mean M_{C_k} and standard deviation SD_{C_k} of the weights of arcs inside cluster C_k.

$$\rho(C_k) = \frac{M_{C_k}}{1 + SD_{C_k}} \tag{4}$$

Let us call N_{C_k} the number of nodes in cluster C_k. An alternative formula for intra-cluster density is

$$\rho(C_k) = \frac{1}{N_{C_k} \times (N_{C_k} - 1)} \sum_{(i,j) \in C_k} w_{i,j} \tag{5}$$

Formula (4) will assign a greater intracluster density to clusters of homogeneously high weight. Formula (5) was designed to take care of situations described in Figure 5. In the case presented, using formula (4) we would get $\rho(A) > \rho(B)$; to put it simply, formula (4) favors homogeneity over connectivity. Therefore we introduced formula (5) for cases when

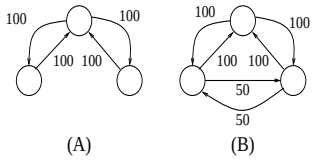

Figure 5: Homogeneous vs. tightly connected clusters

connectivity fits more closely to the application needs. It is very important to notice that in Equation (5) the denominator $N_{C_k} \times (N_{C_k} - 1)$ corresponds to the maximum number of arcs a cluster of N_{C_k} nodes can have (that is $2 \times \binom{N_{C_k}}{2}$). This makes the assumption that the *best* clusters have to be *fully connected* (cliques). We show the limitations of this assumption in the section dedicated to metrics evaluation in [1].

3.3.5 Intracluster density reduction measure

For a given value of the threshold τ, sLoc outputs a set of clusters $C_1, C_2 \dots C_n$ and their respective intracluster densities $\rho(C_1), \rho(C_2) \dots \rho(C_n)$. In order to compare the clusterings performed for each value of τ we need a reduction measure that will summarize the distribution of intracluster densities in a single metric. Therefore, we define ρ, the reduced intracluster density for a given value of τ, with the following formula:

$$\rho = \frac{1}{c \times N} \sum_{k \in [1,n]} N_{C_k} \times \rho(C_k) \tag{6}$$

In the formula above, from formula (3) we know that $c \times N$ is the total number of nodes that belong to a cluster of size larger that s. Thus, the reduction measure ρ is simply a weighted average of the intracluster densities where the weight assigned to a cluster is proportional to its size N_{C_k}. The use of this particular reduction measure can be justified by the fact that, assuming all the concepts have the same probability of being searched for, the probability of returning a given cluster as a response to a query is proportional to the number of nodes this cluster contains.

3.3.6 Intercluster density

In order to compute the intercluster density P, let $\mathcal{A}_{inter} \subset \mathcal{A}$ be the subset of $\mathcal{A}$ such that

$$\mathcal{A}_{inter} = \{a_{i,j} \in \mathcal{A} \mid i \in C_p, j \in C_q, p \neq q\}$$

The set $\mathcal{A}_{inter}$ contains all the arcs going from one cluster to another. What we call A_{inter} is the number of elements in $\mathcal{A}_{inter}$. Then M_{inter} and SD_{inter} are

$$M_{inter} = \frac{1}{A_{inter}} \sum_{a_{i,j} \in \mathcal{A}_{inter}} w_{i,j}$$

$$SD_{inter} = \sqrt{\frac{1}{A_{inter} - 1} \sum_{a_{i,j} \in \mathcal{A}_{inter}} (w_{i,j} - M_{inter})^2}$$

A first definition for the intercluster density P is

$$P = \frac{M_{inter}}{(1 + SD_{inter})} \tag{7}$$

An alternative definition can be

$$P = \frac{1}{n \times (n-1)} \sum_{a_{i,j} \in \mathcal{A}_{inter}} w_{i,j} \tag{8}$$

As with (4) and (5), (7) and (8) differ by the patterns they emphasize in the clustering. Formula (7) will assign a greater intercluster density when the arcs in $\mathcal{A}_{inter}$ are of similar weight, whereas (8) will emphasize the connectivity pattern of the clustering.

Here, notice the denominator in Equation 8, $n \times (n-1)$, where n is the number of clusters after the end of phase 3 in Figure 2; it can be viewed as an assumption made on the clustered graph structure. This measure makes much more sense if the clusters in the set are *fully connected*, that is, if the weights $w_{i,j}$ are distributed fairly homogeneously between clusters.

3.3.7 A theoretical definition of the optimum

The intracluster density measures the similarity of concepts within a cluster. As a result, it is our premise that reduced intracluster density ρ should be maximum at the optimum. At the same time, the intercluster density P measures how similar concepts are across different clusters. Therefore, we want the intercluster density to be minimized at this same optimum. At the same time, the closer the coverage c is to 100%, the larger the number of concepts represented in the knowledge neighborhood. These three metrics vary in completely different ranges; to make them comparable we have them all range between zero and one by simply dividing them by the largest value they actually take across the range of alpha. We then define standardized reduced intracluster density ρ_0 and intercluster density P_0:

$$\rho_0(\tau) = \frac{\rho(\tau)}{Max_{\tau \in \mathcal{T}}(\rho(\tau))} \tag{9}$$

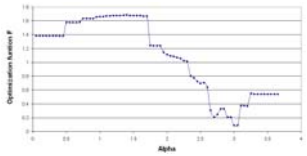

Figure 6: Optimization function F vs. α

$$P_0(\tau) = \frac{P(\tau)}{Max_{\tau \in \mathcal{T}}(P(\tau))} \tag{10}$$

A satisfactory clustering will have a ρ_0 close to 1, a P_0 close to 0, that is, $(1 - P_0)$ close to one, while the coverage has to be as close to 1 as possible too. Therefore, we arrive at the empirical formula for the optimization function F:

$$\mathrm{F} = (\rho_0 + (1 - P_0)) \times c \tag{11}$$

The optimum τ makes F reach a maximum. The above formula aims at determining the clustering that yields an optimum balance between coverage, intra- and intercluster densities. Further experimental evidence is needed, however, on the impact on performance in application environments to justify the use of such an optimum.

This definition of a theoretical optimum thus does not intend to actually fit every real-world application sLoc would be used for. It is rather intended to provide a theoretical baseline, a reference to which other optima can be compared. For instance, the "optimum" clusters may be too large to be properly managed by a dynamic query engine that matches a query vector to one or more clusters. In addition, if concepts inside a given cluster are to be used as suggestions for an interactive search refinement, clusters of size greater than 10 or 20 may not be very user-friendly.

3.4 Results for the optimization function F

Figure 6 shows how the function F varies for different values of α for a leaf-level HDDI node. In all the experiments we conducted, the curve has about the same shape, always exhibiting a maximum for a value of α ranging between 1.25 and 2.

For small values of α, $\rho_0 + (1 - P_0)$ is fairly large and is maximum when alpha is just above zero. Small clusters (size of 2 or 3) are more prone to be fully connected with arcs of maximum similarity than they would be for larger values of α (see Figure 7). Indeed, when α is just above zero, the only arcs that are not pruned out have a weight of Max($\mathcal{W}$); this makes the intracluster densities very large. At the same time, intercluster densities are fairly low. This can be explained by the way weights are assigned; it is fairly uncommon to see $w_{i,j} = \mathrm{Max}(\mathcal{W})$ and $w_{j,i} < \mathrm{Max}(\mathcal{W})$ at the same time. However, the coverage c is very low in this area of the graph, that is, a relatively small proportion of the knowledge base is represented in the knowledge neighborhood. The coverage is therefore used as a *filter* when it multiplies $\rho_0 + (1 - P_0)$ to balance the impact of ρ_0 and P_0 on the optimization function F.

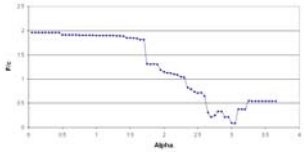

Figure 7: $\rho_0 + (1 - P_0)$ vs. α

In Figure 8, one can see that c has a logarithmic shape, reaching 85% for $\alpha = .6$, its filtering action on $\rho_0 + (1 - P_0)$ then lessens, so that intra- and intercluster densities really lead the way F varies.

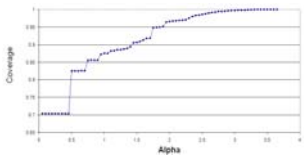

Figure 8: Coverage c vs. α

For large values of α, that is, when the thresholding is not aggressive, some clusters start merging into a couple of very large ones. These very large clusters have very poor intracluster density because they are far from fully connected. At the same time the intercluster density is increasing. This is due in particular to two factors. First, the number of clusters is fairly low, and therefore in formula (8) the sum of the arc weights between clusters is not balanced by the denominator. Second, when $w_{i,j}$ is low, chances are higher that node j is not connected at all to node i, which increases the probability of this arc being an intercluster arc.

It is very interesting to see that the definition of F still fits interior HDDI nodes, despite the different structure and weight distribution of their knowledge

bases. In Figure 9, F exhibits a clearer peak than for leaf-level nodes.

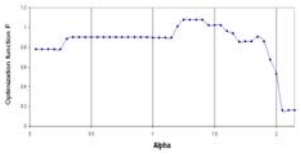

Figure 9: Optimization function F for an interior HDDI node (from Patterns 300)

4 Summary and Conclusion

In order to build an HDDI, we need to cluster the knowledge base graph of each HDDI node. After a thorough study of existing IR clustering techniques, we decided to design our own algorithm to cluster HDDI knowledge bases: sLoc. This algorithm meets the requirements of HDDI in terms of complexity; it is linear and therefore allows *dynamic* identification of regions of semantic locality.

In order to assess the quality of the clustering we designed micro- and macrometrics. We arrived at an optimization function F which uses a subset of these metrics to define a desirable theoretical baseline clustering. Finally, we conducted experiments to gain an understanding of how these metrics and optimization function work and demonstrated scalability across leaf-level and interior nodes of a hierarchical distributed dynamic index.

Now, we need to assess these theoretical measures in a real-life application environment in order to validate our approach. Ongoing work at Lehigh University and the National Center for Supercomputing Applications involves the development of several applications based on HDDI technologies which we are employing to validate our approach.

Acknowledgments

We gratefully acknowledge the assistance and contributions of the staff in the Emerging Technologies and Automated Learning Groups at the National Center for Supercomputing Applications and the staff in the Electrical Engineering and Computer Science Department at Lehigh University[4].

References

[1] F. D. Bouskila, "The Role of Semantic Locality in Hierarchical Distributed Dynamic Indexing and Information Retrieval" PhD thesis, University of Illinois at Urbana-Champaign, Department of Electrical and Computer Engineering, 1999.

[2] Los Alamos National Laboratory, "arXiv.org e-Print archive." http://xxx.lanl.gov, 1999.

[3] National Research Council CSTB, *Computing The Future*. Washington DC: National Academy Press, 1992.

[4] C. J. van Rijsbergen, *Information Retrieval*. London: Butterworths, 1979.

[5] R. B. Korfhage, *Information Storage and Retrieval*. New York: Wiley, 1997.

[6] R. B. Korfhage and T. G. DeLutis, "A basis for time and cost evaluation in information systems," *The Information Bazar. Proceedings of the Sixth Annual Colloquium on Information Retrieval*, pp. 293–326, 1969.

[7] C. W. Cleverdon, "On the inverse relationship of recall and precision," *Journal of Documentation*, vol. 28, pp. 195–201, 1972.

[8] G. Salton, *Dynamic Information and Library Processing*. Englewood Cliffs, New Jersey: Prentice Hall, 1975.

[9] G. Salton and M. McGill, *Introduction to Modern Information Retrieval*. New York: McGraw-Hill, 1983.

[10] K. S. Jones, *Automatic Keyword Classification for Information Retrieval*. London: Butterworths, 1971.

[11] H. P. Luhn, "A statistical approach to mechanised encoding and searching of library information," *IBM Journal of Reasearch and Development*, vol. 1, pp. 309–317, 1957.

[12] M. E. Maron and J. L. Kuhns, "On relevance, probabilistic indexing and information retrieval," *Journal of the ACM*, vol. 7, pp. 216–244, 1960.

[13] H. F. Stiles, "The association factor in information retrieval," *Journal of the ACM*, vol. 8, pp. 271–279, 1961.

[14] M. E. Stevens, V. E. Guiliano, and L. B. Heilprin, *Statistical Association Methods for Mechanized Documentation*. Washington DC: National Bureau of Standards, 1964.

[4] The second author, William M. Pottenger, Ph.D., would also like to express his deep gratitude to his Lord and Savior, Jesus Christ, for His continual love and support.

[15] I. J. Good, "Speculations concerning information retrieval," Tech. Rep. PC-78, IBM Research Centre, New York, 1958.

[16] R. A. Fairhorne, *The Mathematics of Classification*. London: Butterworths, 1961.

[17] L. B. Doyle, "Is automatic classification a reasonable application of statistical analysis of text?" *Journal of the ACM*, vol. 12, pp. 473–489, 1965.

[18] J. J. Rocchio, "Document retrieval systems – optimization and evaluation," PhD thesis, Harvard University, 1966. Report ISR-10 to National Science Foundation, Harvard Computation Laboratory.

[19] S. T. Dumais, M. W. Berry, and G. W. O'Brien, "Using linear algebra for intelligent information retrieval," *SIAM Review*, vol. 37, no. 4, pp. 573–595, 1995.

[20] C. W. Cleverdon, "Report on the testing and analysis of an investigation into the comparative efficiency of indexing systems," tech. rep., College of Aeronautics, Cranfield, UK, 1962.

[21] G. Salton, "Automatic text analysis," *Science*, vol. 168, pp. 335–343, 1970.

[22] S. E. Robertson, "Retrieval and relevance: On the evaluation of IR systems." The ISI Lazerow Lecture, November 1993.

[23] D. K. Harman (ed.), "The first text retrieval conference (trec-1)," *NIST Special Publication 500-207*, 1993.

[24] C. Lynch and H. Garcia-Molina, "Interoperability, scaling, and the digital libraries research," in *1995 IITA Digital Libraries Workshop*, May 1995.

[25] H. Chen and K. J. Lynch, "Automatic construction of networks of concepts characterizing document databases," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 22, pp. 885–902, September 1992.

[26] W. M. Pottenger, "Theory, techniques, and experiments in solving recurrences in computer programs," PhD thesis, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., May 1997.

[27] N. Jardine and R. Sibson, "A model for taxonomy," *Mathematical Biosciences*, vol. 2, pp. 465–482, May 1968.

[28] B. Hendrickson and R. Leland, "The Chaco user's guide: version 2.0," Tech. Rep. SAND94–2692, Sandia, 1994.

[29] G. Karypis and V. Kumar, "Multilevel k-way hypergraph partioning," Tech. Rep. 98-036, University of Minnesota CS&E, 1998.

[30] B.-H. Wang, "Texture segmentation of SAR images," in *Proceedings of Image Understanding Workshop*, May 1997.

[31] J. F. Sowa, *Conceptual Structures: Information Processing in Mind and Machine*. Reading, MA: Addison-Wesley, 1984.

[32] G. W. Mineau and R. Godin, "Automatic structuring of knowledge bases by conceptual clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 7, pp. 824–829, October 1995.

[33] R. Sablowski and A. Frick, "Automatic graph clustering (system demonstration)," in *Proceedings of the Symposium on Graph Drawing, GD '96*, Berkeley, CA, September 1996.

[34] H. Schütze, "Automatic word sense discrimination," *Computational Linguistics*, vol. 24, no. 1, pp. 97–124, 1998.

[35] E.-H. Han, G. Karypis, and V. Kumar, "Clustering in a high-dimensional space using hypergraph models," Tech. Rep. 97-063, University of Minesota, 1998.

[36] J. A. Hartigan and M. A. Wong, "Algorithm 136. a k-means clustering algorithm," *Applied Statistics*, vol. 28, p. 100, 1978.

[37] P. Cheeseman, M. Self, J. Kelly, W. Taylor, D. Freeman, and J. Stutz, "Bayesian classification," in *Seventh National Conference on Artificial Intelligence*, Saint Paul, MN, pp. 607–611, 1988.

[38] R. E. Tarjan, "Depth first search and linear graph algorithms," *SIAM J. Computing*, vol. 1, pp. 146–60, 1972.

[39] A. V. Aho, J. E. Hopcroft, and J. Ullman, *The Design and Analysis of Computer Algorithms*. Reading, MA: Addison-Wesley, 1974.