Chapter

HDDI™: Hierarchical Distributed Dynamic Indexing

William M. Pottenger, Yong-Bin Kim, and Daryl D. Meling

Abstract: 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 and data-mining techniques for computational knowledge management lag behind the rapid growth of distributed collections**.** Hierarchical Distributed Dynamic Indexing (HDDI™) is an approach that dynamically creates a hierarchical index from distributed document collections. At each node of the hierarchical index, a knowledge base is created and subtopic regions of semantic locality in conceptual space are identified. This chapter introduces HDDI™, focusing on the model building techniques employed at each node of the hierarchy. A novel approach to information clustering based on the contextual transitivity of similarity between terms is introduced. We conclude with several example applications of HDDI™ in the textual data mining and information retrieval fields.

Key words: Textual data mining, information retrieval, machine learning, computational knowledge management, artificial intelligence, HDDI™

1. Introduction

Current developments in computer technology are radically changing the nature of distributed information management. Clearly, widespread digitization of information and the ubiquity of networking have created fundamentally new possibilities for collecting and distributing information. Just as clearly, it is critical that new information infrastructure be developed that enables effective mining of the huge volume of distributed information emerging in digital form [PCP00]. A recent article estimated that no more than about 16% of the publicly accessible web has been indexed by any single search engine [LG99].

Traditional methods of indexing combine multiple subject areas into a single, monolithic index. There are enough documents on the Web that such indexing technology often fails 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 use. This problem has been known for some time [NRC92]. Solutions based on link analysis have mitigated these difficulties yet fail to adequately address issues of recall [BP98].

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 *indices* must also be distributed. Secondly, it must be realized that the information contained in these distributed digital repositories can be classified in a hierarchical manner. Traditionally, knowledge hierarchies, or ontologies, have been created with human expertise. One popular form is the thesaurus (e.g., the National Library of Medicine's MeSH thesaurus). 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 indices 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 21st 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: HDDI™, *Hierarchical Distributed Dynamic Indexing*. HDDI™ is a novel approach to organizing large quantities of unstructured data in a loosely coupled distributed environment under development at Lehigh University and at the National Center for Supercomputing Applications. The approach is based on the algorithmic creation of subtopic regions of semantic locality in sets of distributed documents; this allows automatic discovery of similarities at a fine level of granularity amongst concepts within documents. In this way, hierarchical indices (such as those created now "by hand" in many places on the web; www.yahoo.com is probably the most well-known example) are generated for topics in documents in a volatile, distributed environment, providing the information seeker with an always up-to-date map of information spaces. The ability to generate large hierarchical indices on the fly allows for a realistic, useful mapping of cyberspace without the need for time-consuming human intervention. This technique is most valuable when applied to items within some institutional zone — to map out, for instance, large sets of corporate or scientific documents. Here, subjective issues relating to "importance" or "quality" can be sidestepped, and the power of the HDDI™ strategy can be fully leveraged—an unstructured set of documents lacking any sort of metadata can be bound to a hierarchical knowledge structure generated automatically based on word frequencies.

HDDI™ is a natural outgrowth of advances in technology that leverage existing, inherited knowledge structures popular in today's web-based networked environment. As HDDI™ technology develops, we are discovering novel approaches that address several issues of managing distributed digital information within the context of the HDDI™ paradigm.

This chapter will focus on the core computational knowledge management algorithms employed in building models at the nodes of hierarchical indices. Although the focus of this book is on data mining of scientific data sets, we do not limit our discussion to scientific data in this chapter, but rather consider textual data in all forms. This is only natural given the ubiquitous employment of the written word in human endeavor.

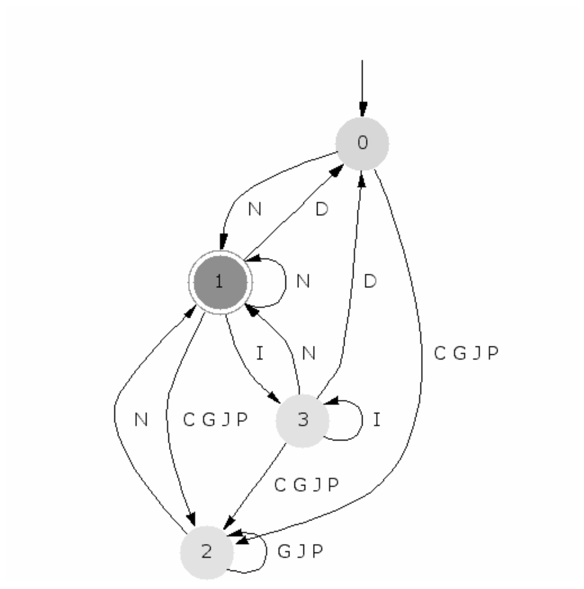
In the following sections we present an outline of HDDI™ technology employed in building hierarchical indices. We close with a summary of applications in which HDDI™ technology is being deployed.

2. Building a Hierarchical Index

The following steps are involved in the process of building a hierarchical index: concept identification/extraction; concept co-occurrence matrix formation; hierarchy construction; knowledge base creation; identification of regions of semantic locality; and hierarchy mapping. Each of these steps is outlined in more detail below. Several aspects of this approach reflect our initial intuition on how the problem should be addressed. Each of these six steps is being addressed in the course of our research in order to refine our approach.

2.1. Concept Identification/Extraction

Our approach to concept identification/extraction includes the following three steps: input item (document) parsing, part of speech tagging and concept identification. The parsing stage takes SGML, HTML or generalized XML tagged items as input. We have utilities to convert from a variety of input formats to XML, including proprietary airline safety data and US government patent data. Based on AI techniques [C00], [B92], our part of speech tagging approach includes the use of both lexical and contextual rules for identifying various parts of speech. After identifying each word's part of speech, we invoke a finite-state machine (pictured below) that accepts maximal length English-language noun phrases. [BCGKMP01], [K96].



A Finite State Automaton for Recognizing Complex Noun Phrases State 0 is the start state, and state 1 the final state. C is a cardinal number, G a verb (gerund or present participle), P a verb (past participle), J an adjective, N a noun, I a preposition and D a determiner.

Our enhanced state-machine identifies and extracts concepts consisting of complex noun phrases composed of multiple modifiers, including gerund verb forms. The final result of these three steps is a reformulation of the original collection that includes a summation of the location and number of occurrences of each extracted concept. The next stage of the process receives this reformulated collection.

2.2. Concept Co-occurrence Matrix Formation

"Co-occurring" defines concepts that occur within the same item. An item can be defined as an intelligently created logical unit of text that is cohesive semantically. Examples include abstracts, titles, web pages, airline safety incident reports, patents, etc. The co-occurrence relation is reflexive and symmetric but not transitive. Given concepts extracted by the above process, we compute concept frequency and co-occurrence matrices. We also compute the frequencies of co-occurrences of concept pairs among all items in the set.

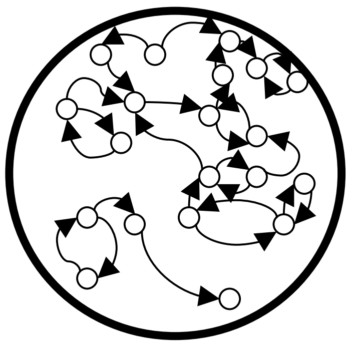
The literature discusses various definitions of co-occurrence [BYRN99]. Our approach incorporates measures based on proximity as well as techniques that dynamically define the extent of sub-items within a given item (see Section 2.5). Our preliminary results indicate that this latter approach is crucial in dealing with the full text of items. We also reported on research in parallelizing the computation of such semantic relations based on the theory of *coalescing loop operators* [P98]. Similar techniques are being applied to scale the computational information management algorithms developed as part of this research to large collections.

2.3. Hierarchy Construction

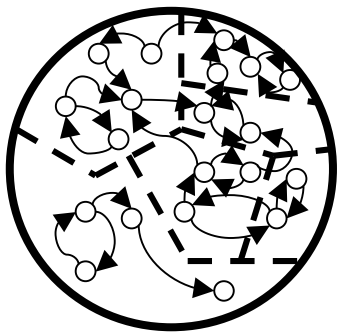
Hierarchy construction is a meta-level organizational process that combines the matrices formed in the previous step. These co-occurrence matrices provide the basis for organizing concepts into an ontology of knowledge based on the content of the collections. Systematic filtering, pruning, and meshing lower level (child) matrices form the hierarchical structure, producing higher level (parent) combined matrices. The process is iterative in that matrices are pruned and merged at successively higher levels. The resulting hierarchical index consists of high-resolution leaf-level index nodes that become increasingly less precise (i.e., more general) as the hierarchy is built. This process is visualized in Section 3 below, and is a topic of ongoing research [K01].

2.4. Knowledge Base Creation

Knowledge base creation is the second meta-level organizational process. For each matrix in the hierarchy constructed in the previous step, and for each concept in each matrix we compute a similarity with other concepts. This one-to-many mapping associates each concept with a list of related concepts ranked by similarity. Co-occurring concepts are ranked in decreasing order of similarity. More general concepts occur lower in the list. Each concept pair is weighted, creating asymmetric measures of pair-wise similarity between



Knowledge Base



Knowledge Neighborhood

concepts. The similarity is a mapping from one concept to another that quantitatively determines how similar they are semantically. We term the resultant mapping a *knowledge base*[1]. A knowledge base is represented as an asymmetric directed graph in which nodes are concepts and arc weights are similarity measures. The knowledge base can be visualized as a graph, illustrated by example on the left below, in which vertices represent concepts and edges represent the pair-wise similarity between concepts.

In [P97] techniques were implemented that produce a knowledge base using an extension of the statistical model developed in [CL92]. The model building techniques are based on a *cluster function* defined as follows [CMNS97]:

$$ClusterWeight(C_j, C_k) = \frac{\sum_{i=1}^{n} d_{ijk}}{\sum_{i=1}^{n} d_{ij}} \times WeightingFactor(C_k)$$

$$ClusterWeight(C_k, C_j) = \frac{\sum_{i=1}^{n} d_{ikj}}{\sum_{i=1}^{n} d_{ik}} \times WeightingFactor(C_j)$$

These two equations indicate the cluster weights, or similarity, from concept C_j to concept C_k (the first equation) and from concept C_k to concept C_j (the second equation). d_{ij} and d_{ik} are the product of concept frequency and inverse document frequency and are defined in a similar manner (this is a variation of the popular *tf*idf* measure used primarily in text-based vector space modeling where *tf* is concept frequency and *idf* is inverse document frequency [S89]). d_{ij}, for example, is defined as:

$$d_{ij} = tf_{ij} \times \log\left(\frac{N}{df_j} \times w_j\right)$$

where N represents the total number of documents in the collection, tf_{ij} is the frequency of occurrence of concept C_j in document i, df_j is the number of documents (across the entire collection of N documents) in which concept C_j occurs, and w_j is the number of words in concept C_j.

d_{ijk} and d_{ikj} represent the combined weights of both concepts C_j and C_k in document i and are also defined in a similar manner. d_{ijk}, for example, is defined as follows:

$$d_{ijk} = tf_{ijk} \times \log\left(\frac{N}{df_{jk}} \times w_j\right)$$

[1] Note that follow-on work that builds on [CL92] terms this a *Concept Space.*

Here tf_{ijk} represents the minimum number of occurrences of concept C_j and concept C_k in document i. df_{jk} represents the number of documents (in a collection of N documents) in which concepts C_j and C_k occur together. The final expression, w_j, is the number of words in concept C_j.

In order to penalize general concepts that appear in many places in the co-occurrence analysis, a weighting scheme similar to the inverse document frequency function is employed. C_j, for example, has the following weighting factor:

$$WeightingFactor\left(C_j\right) = \frac{\log \frac{N}{df_j}}{\log N}$$

Concepts with a higher value for df_j (i.e., more general concepts) have a smaller weighting factor, which results in a lower similarity. Co-occurring concepts are ranked in decreasing order of similarity, with the result that more general concepts occur lower in the list of co-occurring concepts.

Ongoing research includes enhancement of this cluster function to account for several additional factors including, for example, metrics such as the ratio of commonly used to total words in a concept.

Our high-performance implementation for computing knowledge bases employs cluster functions of this nature in a parallel C^{++} application optimized to achieve efficiencies of over 100% on the SGI/Cray Origin2000 and SGI Power Challenge supercomputers [P98]. Based on the theory of *coalescing loop operators*, the outermost loop in this application was parallelized [P97]. In conjunction with this outer-loop parallelism, a custom memory manager enabled the superlinear speedups reported in the table following on the SGI Power Challenge. Runtimes are recorded as wall-clock execution time for repositories of different sizes.

	Num. Procs.	CR h:m:s	CR S_p	DR h:m:s	DR S_p
Serial	1	1:17:19	1	10:30:27	1
Parallel	2	0:27:16	2.84	3:32:49	2.96
	4	0:14:28	5.34	1:53:52	5.54
	8	0:08:41	8.90	1:08:20	9.23
	16	0:05:44	13.49	0:39:22	16.01

Speedups on SGI Power Challenge

2.5. Identification of Regions of Semantic Locality

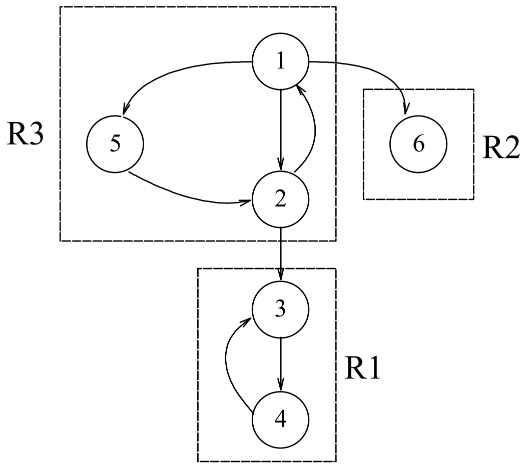
The resulting weight assignments from knowledge base creation are context-sensitive. We use these weights to determine regions of semantic locality (i.e., conceptual density) within each collection. We then detect clusters of concepts within a knowledge base [BP00], [B99], [T72].

The result is a knowledge base consisting of regions of high-density clusters of concepts – subtopic regions of semantic locality. These regions consist of clusters of concepts that commonly appear together and collectively create a *knowledge neighborhood.* Our premise is that we can impute a constrained, *contextual transitivity* to the co-occurrence relation [BP00], thereby forming regions of semantic locality (see illustration on the right in section 4 above). The motivation for the use of the term semantic locality comes from the commonly applied premise that grouping similar concepts together leads to increased effectiveness and efficiency in query search and retrieval [SJ71]. Note however that the similarity relation is by definition not transitive. The theoretical basis for our algorithm, sLoc, is a concept we term *contextual transitivity* in the similarity relation. In essence this means that a threshold is chosen based on the structure and distribution of the similarities in the knowledge base and transitivity is constrained accordingly.

Contextual transitivity extends Schütze's conceptualization of *second order co-occurrence* [S98] by using *n-order* co-occurrence, where *n* varies with the underlying semantic structure of the model. Here is a simple example of how sLoc employs contextual transitivity to group concepts together in a region of semantic locality. 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 necessarily co-occur. It may, however, be useful to gather the three concepts in one unique class and infer that *e-commerce* and *applet* are actually related. That is what we mean by contextual transitivity. Note that similarity is not a boolean relation – rather, it can take on a range of values. Two concepts can thus be more or less similar. The characteristics of the structure of a given knowledge base lead to a heuristic minimum similarity value needed for two concepts to occur 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.

The computational core of sLoc is based on an algorithm due to Tarjan [T72]. 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 an NP-hard problem and the only heuristics we are aware of are not linear. The theory can be found in [AHU74]. The figure below depicts the operation of Tarjan's algorithm as it identifies strongly connected regions (R_1, R_2, R_3) in a simple graph.

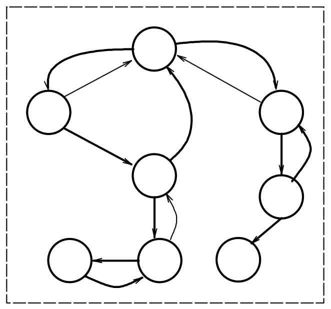
Before tackling the algorithm itself we must first introduce the following notation: Let $\mathcal{N}$ be the set of nodes i in the input graph, and let N be the total number of nodes. Let $\mathcal{A}$ be the set of arcs in the input graph and A the total number of arcs. An arc $a_{ij} \in \mathcal{A}$ connects node i to node j. Let $\mathcal{W}$ be the set of arc weights in the graph and $w_{i,j}$ the weight of the arc going from node i to node j. Therefore $\mathcal{W} = \{w_{i,j}\}_{(i,j) \in \mathcal{N}^2}$.

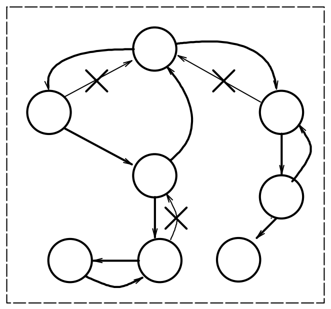


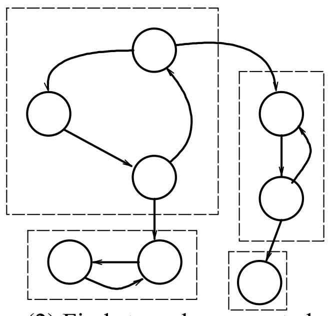
A knowledge base is an asymmetric graph and therefore $w_{i,j}$ may differ from $w_{j,i}$. Moreover, if $a_{i,j} \notin \mathcal{A}$ then $w_{i,j} = 0$; in particular, for all i, $w_{i,i} = 0$. Finally, let M be the mean of the arc weights $M = 1/A \sum_{(i,j)\in N^2} w_{i,j}$ and SD the standard deviation of the distribution of arc weights $SD = \sqrt{1/(A-1) \sum_{(i,j)\in N^2} (w_{i,j} - M)^2}$.

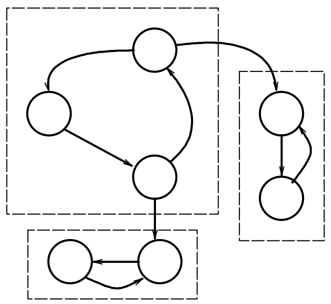
2.5.1. The sLoc Algorithm Step by Step

The figure below depicts the three steps of the sLoc process. Prior to the first step, the weights in the knowledge base are normalized (step 0 below). The first step in sLoc is to statistically prune the input graph. Arcs of weight smaller than a certain threshold τ are virtually pruned. Note that since the similarities are asymmetric, an arc from concept *a* to concept *b* can be pruned while the arc back from *b* to *a* remains. The second step involves the 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. During the third and final 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 $\boldsymbol{\tau}$, the more arcs are cut off, and therefore the smaller in size the


(0) Standardize weights in the raw knowledge base


(1) Prune the graph

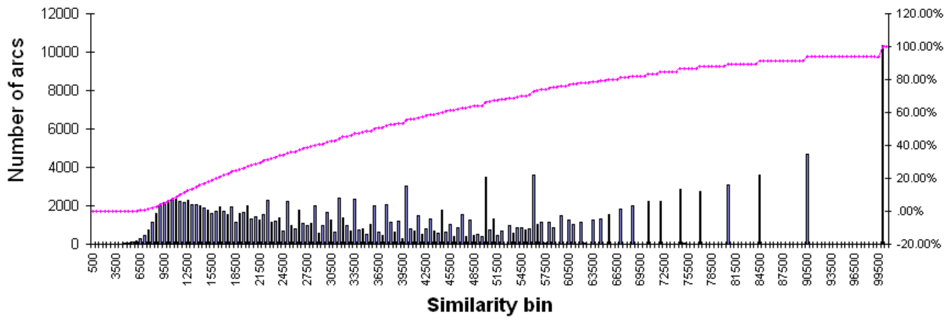

(2) Find strongly connected regions in the pruned graph


(3) Discard clusters of size smaller than s (here s = 2)

strongly connected regions. Thus the greater $\boldsymbol{\tau}$ the smaller in size and the more focused will be the regions of semantic locality.

The question that has arisen is twofold: Given an input knowledge base, what threshold $\boldsymbol{\tau}$ should be applied? Secondly, can we determine a value for the threshold that scales across domain and collection size?

Let us tackle the first question on how to determine the value of $\boldsymbol{\tau}$ yielding an optimum clustering. Our premise is that the optimum $\boldsymbol{\tau}$ 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.). We have conducted preliminary experiments to study the distribution of weights in various knowledge bases. Our preliminary results indicate that the distribution of weights is quite consistent across both subject domain and collection size. The figure above represents one such common distribution.



This was computed from the MED gold standard information retrieval collection [S75].

It should be stressed that we have yet to compute a knowledge base that does not exhibit a distribution of this nature. Given this fact, we have developed the following heuristic for the threshold **τ**: **$\tau(\alpha)$ = max(w) - α * *SD***. In this equation, **τ** is the cut-off weight used to prune the graph and **α** is the number of standard deviations. For example, **τ(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.

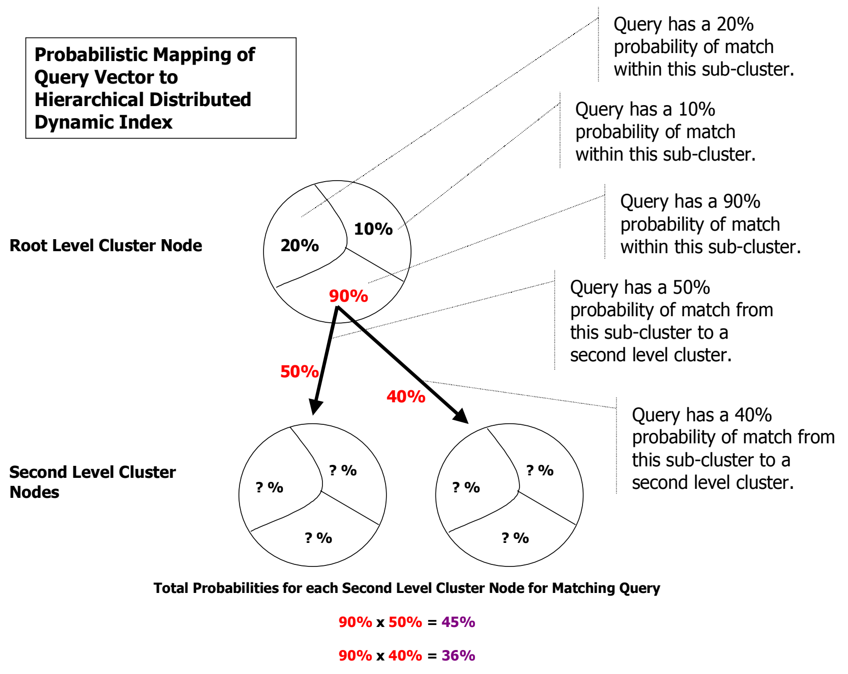
At this point in time we are still investigating answers to the second question posed above. We have conducted initial experiments in some of the application domains mentioned in Section 3 that indicate the range of α between [1.5, 2.0] is suitable [BP00], [Y00], [PY00].

2.6. Hierarchy mapping

The final process is to create a mapping of the information space by developing linkages between the clusters identified during semantic locality detection. These mappings supply the necessary paths between information areas at higher and lower levels of the hierarchy. Links between clusters are based on overlap of content between clusters as well as similarities in topology. This process defines relationships between clusters, so that clusters are interconnected both with the child clusters that they were merged from and the higher-level parent clusters they were used to create. Upon completion of this phase, each level of the hierarchical index contains sufficient information to determine its relative position as a knowledge area in the overall knowledge hierarchy. This is an area of ongoing research [K01].

3. Applications of HDDI™

One of the first applications of HDDI™ is in the organization of large distributed and dynamic data sets into a cohesive hierarchical framework for optimal probabilistic search. Our research objective is to create a distributed system whereby collections are automatically indexed hierarchically for use in concept-based knowledge searches as depicted in the figure below.


Probabilistic Mapping of Query Vector to Hierarchical Distributed Dynamic Index
Query has a 20% probability of match within this sub-cluster.
Query has a 10% probability of match within this sub-cluster.
Query has a 90% probability of match within this sub-cluster.
Query has a 50% probability of match from this sub-cluster to a second level cluster.
Query has a 40% probability of match from this sub-cluster to a second level cluster.
Root Level Cluster Node
20%
10%
90%
50%
40%
Second Level Cluster Nodes
? %
? %
? %
? %
? %
? %
Total Probabilities for each Second Level Cluster Node for Matching Query
90% x 50% = 45%
90% x 40% = 36%

Using the HDDI™ technology infrastructure, we are also investigating the automatic detection of emerging content. Trends in scientific research, technology forecasting, and automatic detection of emerging interpretations in case law are just a few examples of the variety of applications in which ongoing research that employs HDDI™ technology is being conducted [PY00].

We have also recently explored scaling HDDI™ technology from the textual to the numeric domain. HDDI™ techniques cluster *concepts* into classes. An example of a concept in textual data mining is a noun phrase. Our approach to asset management is to identify distinct *financial concepts* in financial data, and apply the HDDI™ model building and clustering-partitioning techniques to group financial concepts into regions exhibiting similar characteristics that can then be employed in predictive financial modeling.

Another area of HDDI™ research involves processing data that records both synchronous and asynchronous interactions between individuals. Using HDDI™ technologies, we are scaling our models to build relationship networks between people that can then be correlated with semantic regions of locality in the topics under discussion. Tools based on these methods provide insight into the types and range of topics and associations in social interactions and can be used, for example, to identify social and semantic links between groups of authors and collections of publications.

A related research initiative involves the design of a multimedia framework for constructive, inquiry-based learning for introductory and upper level computer science courses. The content is drawn from the field of Object-Oriented Software Engineering (OOSE). This National Science Foundation project includes development of HDDI™ textual data mining techniques for establishing temporal context that will be automatically employed by learners to detect emerging trends in OOSE research [BPK01]

4. Conclusion

We have outlined a set of core algorithms for building models in HDDI™, a framework for mining and management of distributed textual data. Current research includes the use of stochastic approximation to determine confidence intervals for optimal values of various parameters used in HDDI™ (such as the parameter α in sLoc), as well as the construction of an agent system that executes HDDI™ in a distributed environment. Ongoing work continues to focus on the validation of HDDI™ model building and indexing techniques in a variety of applications.

5. Acknowledgements

We gratefully acknowledge the assistance of our Lord and Savior Jesus Christ in completing this work. We also gratefully acknowledge the assistance of the many who worked together with us at the National Center for Supercomputing Applications and at Lehigh University to make this a reality.

6. References

[AHU74] V. Aho, J. E. Hopcroft, and J. Ullman, *The Design and Analysis of Computer Algorithms,* Addison-Wesley, Reading, MA.

[BCGKMP01] R. Bader, M. Callahan, D. Grim, J. Krause, N. Miller and William M. Pottenger, The Role of the HDDI^{TM} Collection Builder in Hierarchical Distributed Dynamic Indexing, *Proceedings of the Textmine '01 Workshop, First SIAM International Conference on Data Mining*, April.

[BYRN99] R. Baeza-Yates and B. Ribeiro-Neto, Eds. Modern Information Retrieval, ACM Press, New York.

[BPK01] G. Blank, William M. Pottenger, G. D. Kessler, The CIMEL Project: Constructive, Collaborative, Inquiry-based Multimedia E-Learning, http://www.eecs.lehigh.edu/~cimel.

[B99] F. D. Bouskila, The Role of Semantic Locality in Hierarchical Distributed Dynamic Indexing and Information Retrieval, M.S. Thesis, Department of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign, December (Bouskila's thesis work was supervised by William M. Pottenger).

[BP00] F. D. Bouskila and William M. Pottenger, The Role of Semantic Locality in Hierarchical Distributed Dynamic Indexing, *Proceedings of the International Conference on Artificial Intelligence (IC-AI'2000)*, Las Vegas, NV, June.

[B92] E. Brill, A Simple Rule-based Part of Speech Tagger, *Proceedings of the Third Conference on Applied Natural Language Processing*, ACL.

[BP98] S. Brin and L. Page, The Anatomy of a Large-Scale Hypertextual Web Search Engine, *Proceedings of the Seventh International World Wide Web Conference*, Brisbane, Australia, April.

[CL92] H. Chen and K. J. Lynch, Automatic Construction of Networks of Concepts Characterizing Document Databases, *IEEE Transactions on Systems, Man and Cybernetics*, 22(5):885-902, September/October.

[CMNS97] H. Chen, J. Martinez, T. Ng and B. R. Schatz, A Concept Space Approach to Addressing the Vocabulary Problem in Scientific Information Retrieval: An Experiment on the Worm Community System, *Journal of the American Society for Information Science*, Volume 48, Number 1, January.

[C00] G. Cooke, SemanTag, gcooke@rt66.com.

[K96] L. Karttunen, Directed Replacement. *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics,* Santa Cruz, California.

[K01] Y. B. Kim, The Role of Hierarchical Models in Hierarchical Distributed Dynamic Indexing, M.S. Thesis, Department of Computer Science at the University of Illinois at Urbana-Champaign, June.

[LG99] S. Lawrence and C. L. Giles, Accessibility of Information on the Web, Nature, Volume 400, pages 107—109.

[NRC92] National Research Council, *Computing the Future: A Broader Agenda for Computer Science and Engineering*, National Academy Press.

[P97] William Morton Pottenger, Theory, Techniques, and Experiments in Solving Recurrences in Computer Programs, Ph.D. thesis, Center for Supercomputing Research and Development in the Department of Computer Science at the University of Illinois at Urbana-Champaign, www.eecs.lehigh.edu/~billp/pubs/PhDThesis.ps, May.

[P98] William M. Pottenger, The Role of Associativity and Commutativity in the Detection and Transformation of Loop-Level Parallelism, In the *Proceedings of the 12th International Conference on Supercomputing*, www.eecs.lehigh.edu/~billp/pubs/2057.ps.gz, Melbourne, Australia, July.

[PY00] William M. Pottenger, Detecting Emerging Concepts in HDDI™. *Proceedings of the Computational Information Retrieval Workshop (CIR00)*, Raleigh, NC. October.

[PCP00] William M. Pottenger, M. R. Callahan, and M. D. Padgett, *Distributed Information Management,* Annual Review of Information Science and Technology (ARIST), The American Society for Information Science.

[S75] G. Salton, *Dynamic Information and Library Processing*, Prentice Hall, Englewood Cliffs, New Jersey.

[S89] G. Salton, *Automatic Text Processing,* Addison-Wesley Publishing Company, Inc., Reading, MA.

[S98] H. Schütze, Automatic Word Sense Discrimination, *Computational Linguistics,* vol. 24, no. 1, pp. 97-124.

[SJ71] K. Sparck-Jones, *Automatic Keyword Classification for Information Retrieval*, Butterworths, London, 1971.

[T72] R. E. Tarjan, Depth first search and linear graph algorithms, *SIAM J. Computing*, 1:146-160.

[Y00] T. Yang, Detecting Emerging Conceptual Contexts in Textual Collections, M.S. Thesis, Department of Computer Science at the University of Illinois at Urbana-Champaign, February.