Docking topical hierarchies:A comparison of two algorithms for reconciling keyword structures

Bryan Tower Mark Chaisson
Richard K. Belew
Cognitive Science Dept. (0515)
Univ. California – San Diego
La Jolla, CA, 92093-0515
rik@cogsci.ucsd.edu

11 December 2000 *

Abstract

Hierarchies are a natural way for people to organize information, as reflected by the common use of "broader/narrower" term relation in keyword thesauri. However, different people and organizations tend to construct different conceptual hierarchies (e.g., contrast Yahoo! with the UseNet news hierarchy), and while there are often significant commonalties it is in general quite difficult to fully reconcile them. We are particularly interested in the problem of "docking" a narrower, more focused and refined topical hierarchy into a broader one, and describe two algorithms for accomplishing this task. The first matches hierarchies based on a bipartite matching algorithm of (textual) features of nodes without consideration of their hierarchic organization, and the second is based on an attributed tree matching algorithm which uses both hierarchic structure and node features. We present experimental results showing the performance of both algorithms on a set of very different topical hierarchies, all designed to represent the field of `Computer_Science`. These show that hierarchic structure does indeed allow more accurate matches than nodes alone.

1 Introduction

Especially within the Western analytic tradition, a central feature of intellectual progress has been the progressive division of broad topical areas into narrower, more refined ones. Science and technology has continued to drive the analytic process into an incredible breadth of disciplinary specializations. And within each of these disciplines, generally opaque to outsiders, are even more refined characterizations of narrower topical areas.

*Submitted SIGIR01, January 2001. Released as technical note, 25 Jan 05

Collectively, these various taxonomic classification systems can be imagined as a vast tree, with broad topical areas splitting into more refined branches. The central problem considered in our work is the **docking** of one taxonomic system into another when these two are developed independently. Our system, HierDock, implements the process of docking one topical hierarchy into another, matching comparable categories and preserving as much of the structure as possible.

There is some reason to believe that two hierarchies developed independently will be organized in a structurally similar way. People tend to classify things according to a "broader vs. narrower" relation. Our system leverages the fact that two hierarchies built with this principle will overlap in some places.

1.1 Motivation

There are a number of alternative goals that motivate this research. The original motivation grew out of attempts to integrate high-quality, third-party taxonomic classifications into an over-arching representation like the Encyclopedia Britannica's (EB) Propaedia. For example, early work by Steier and Belew (unpublished) built on that done by Rose on legal texts [9, 8], especially the Westlaw Key Number system used to classify it. But as Steier and Belew moved to work on the EB corpus [10, 11], the question arose how West's key numbering system might match into EB's Propaedia. More recently, reconciling our Computer Science & Engr. department's curriculum with that of professional organizations like the Association for Computing Machinery (ACM) and other universities is proving very useful in our own department's planning. More generally, in a wide variety of fields (e.g., bioinformatics) various investigators and institutions are attempting to classify contents in shared "ontologies" (e.g., of protein functions) that allow shared access to common data sets.

In Section 2 we describe the two algorithms that we use in the experiments, a bipartite matching algorithm and a weighted hierarchic subtree isomorphism algorithm recently developed by Pelillo et al. for the purpose of image matching [7]. Section 3 describes where the data sets came from and how they were collected. In Section 4 we present preliminary results of the two algorithms on the data sets, and in Section 5 we discuss some of the observations and results found in the experiments. Finally we end with some conclusions, and future direction in Section 6.

2 Algorithms for matching hierarchies

A common and natural representation of a topical hierarchy is as a directed graph (digraph): nodes represent a topical area. We associate a **rubric** (short, descriptive textual phrase) with each node and optionally a collection of longer **textual passages** that are considered exemplary of the topical area. Directed arcs encode the **broader-term/narrower-term** (BT/NT) relationship between areas, with directionality running from broader to narrow nodes. A

precise semantics for the BT/NT relation is a topic of ongoing research (cf. Section 3.4), but for now we will assume a form of the *inclusion* relation. We shall refer to this representation of a topical hierarchy as an *HGraph*.

Given two *HGraph*s H_1 and H_2 there are two distinct approaches to the design of an algorithm for matching them. The first is to ignore all textual information associated with the graph's nodes and focus exclusively on matching the two graphs' *edge* structures. The problem then becomes the well-known but NP-complete problem of subgraph isomorphism. Fortunately, because the two graphs are both rooted trees, the complexity of this matching process is polynomial rather than NP-complete as it is in the general case [4, Chapter 4.2]. However, given the semantic attachment we have to the textual features associated with our graph's nodes, there seems little practical interest in solutions based exclusively on structural similarities and so do not consider such methods any further.

The other extreme alternative is to consider only the *nodes* in the two graphs, specifically the textual materials associated with them. The search is then for the pairing of nodes from H_1 with those of H_2 such that the cumulative match score across all pairs of nodes is maximized. This pure formulation becomes the problem of *maximum weighted bipartite matching*, considered in more detail in Section 2.1.

Between these two extreme approaches, focusing exclusively on edges vs. nodes contained in the *HGraph*, are potentially a wide range of techniques that exploit both sources of information. The central hypothesis investigated by this work is that the structural information provided by edges connecting topical nodes provides more information than that contained in the nodes' free text alone.

2.1 Maximum Weighted Bipartite Matching

The unweighted bipartite matching problem is a classical network flow problem [3, Chapter 27.3]. To solve a network flow problem you start with an empty flow (matching) and continually add "augmenting" paths to the flow. An "augmenting" path is a path that will increase the weight of the matching without making the matching invalid.

The weighted bipartite matching problem is slightly different, because with a variation in edge weights it becomes possible to connect one node with more than a single other node with a traditional network flow approach. This violates the definition of a matching problem, because in a matching each node can be matched to at most one other node. Note that the weights prevent it from becoming a pure network flow problem.

The problem is to find the maximum matching between two sets of nodes L and R. There exists a set of edges E, where if $e = (x, y)$ then $x \in L$ and $y \in R$, for all $e \in E$. A maximum matching is a subset of the edges, $M \cup E$, where $|M|$ is maximal. To find a maximum matching M_{max} we start with an empty matching, $M = \epsilon$, and continually added an augmenting path until no more augmenting paths exist. An augmenting path is a path starting with an

unmatched node in L (without loss of generality) ends with an unmatched node in R and

An algorithm was described by Cheng et al. [2] that solves the maximum weighted bipartite matching problem used successfully in computer vision to perform image feature matching. The same general idea is used to solve the weighted bipartite matching problem, that is, start with an empty matching and at each step add an augmenting path. It has been proven that if at each step in the algorithm a maximum augmenting path is added to the matching then the result will be a maximum weight matching [2]. Details on how to search efficiently for a maximum augmenting path with respect to a matching, M, are described in a paper by Hao and Kocur, from the DIMACS implementation challenge on network flows and matching [6].

2.2 Maximum Weighted Subtree Isomorphism

In a recent paper Pelillo et al. show how to convert an "attributed tree" (i.e., nodes have attributes over which a similarity measure has been defined) matching problem into a corresponding Maximum Weighted Clique problem, such that there is a one to one correspondence between solutions to the Maximum Weighted Clique problem and solutions to the maximum attributed tree matching problem [7].

The details of the formulation of this as a continuous quadratic assignment problem are beyond the scope of this conference paper, and only a sketch can be presented here. In brief, matching two hierarchies H_1 and H_2 requires the construction of a square matrix of size n^2, where $n = |V_1| \times |V_2|$ and $|V_1|$ and $|V_2|$ are the number of nodes in H_1 and H_2, respectively. Optimization proceeds by locally optimizing a "characteristic" vector over the standard simplex $\Re^n$. Pelillo et al. report results using replicator equations to solve this problem and report that the basins of attraction of optimal or near optimal solutions are large [7].

Our early experiments were able to use this algorithm successfully over hierarchies representing topical hierarchies, but only on quite small examples (e.g., dmoz-nl1 vs. dmoz-nl2, cf. Section 3). On larger problems this iterative procedure proved to converge very slowly. In this application we also observed a much greater sensitivity to initial conditions and many sub-optimal local solutions. For these reasons, all larger examples used an indefinite quadratic program solver known as QPOPT [5]. On all problems for which QPOPT and replicator equations were both used, both methods obtained similar results.

3 Data Sets

As discussed in Section 3.4, the range of sources from which topical hierarchies can be derived is very broad. In order to focus our experiments, all data sets were focused on the area of `COMPUTER SCIENCE`, a topical area about which the authors are particularly familiar. The data sets used for the experiments

Table 1: HierDock data set statistics

Name	Description	node count	max depth	vocab size
acm2	ACM collapsed to two levels	12	2	975
acm3	ACM collapsed to three levels	93	3	975
dmoz-nl1	1st half of the Natural Language	8	3	4465
dmoz-nl2	2nd half of the Natural Language	8	3	4173
dmoz-ai1	1st half of the Artificial Intelligence	74	5	17286
dmoz-ai2	2nd half of the Artificial Intelligence	75	5	18570
dmoz-cs	Computer Science	137	5	24544
eb	extracted from article	50	5	1678

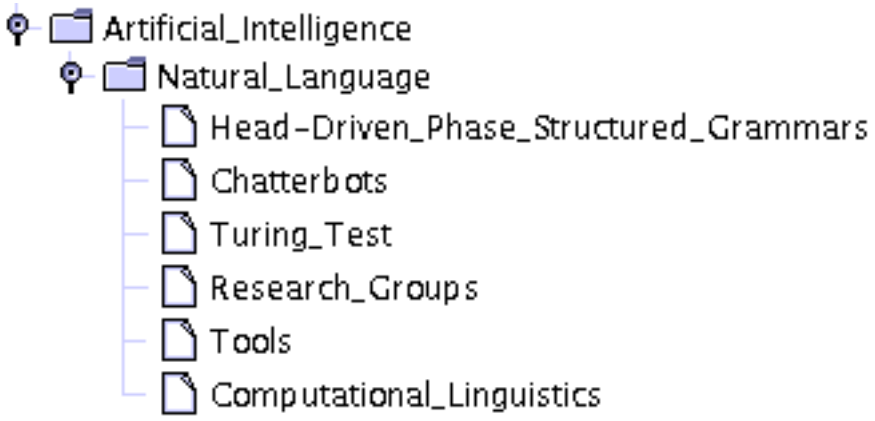


Figure 1: Visual representation of the dmoz-nl data set.

reported here were gathered from three sources: the Mozilla Open Directory project (DMOZ), the ACM Computing Reviews taxonomy, and an article on "Computer Science" produced by Encyclopedia Britannica. A brief discussion of how each one was collected follows, and their basic statistics are collected in Table 1.

3.1 DMOZ

The Mozilla Open Directory project makes their topical hierarchy readily available.[1] For the experiments reported here we extracted several small subtrees from the full DMOZ hierarchy. The first tree we used was the subtree rooted at `TOP/COMPUTERS/ARTIFICIAL_INTELLIGENCE`. The second subtree was rooted at the same node, but all of its children were removed except the `NATURAL_LANGUAGE` node; the second set is shown in Figure 1.

[1]The data is available in Resource Description Format (RDF) at DMOZ at `http://dmoz.org/rdf/`. The version used in these experiments was obtained on 5/24/2000.

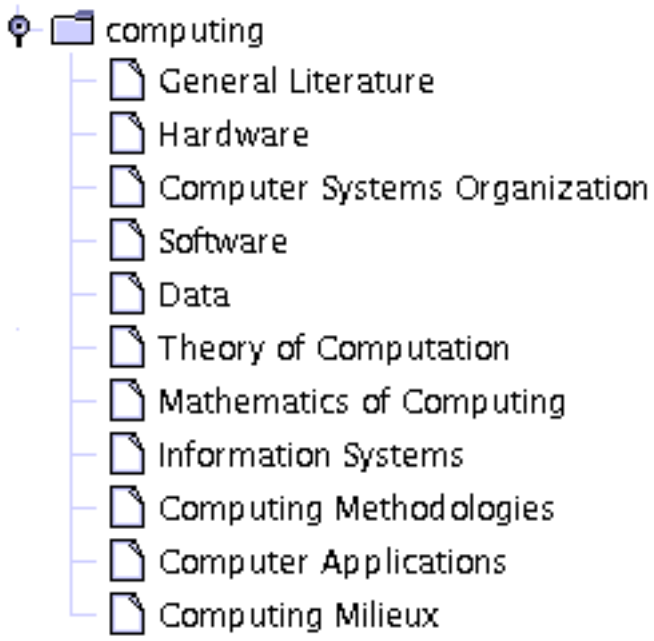


Figure 2: Visual representation of the acm2 data set.

The DMOZ category pages themselves provide very little text that can be associated with the topical area: only the anchor text selected by the editor to point to the relevant URL, and a brief description. In order to provide more extensive textual samples, we also did a simple, one-level crawl from these pages and used these pages' text as well. [2]

Pages relevant to a topic and referenced by DMOZ editors are listed in alphabetic order. The assumption is that while all the pages at one node are highly semantically-related (since the same editor chose to classify them identically), the words that happen to be selected on one page is likely to vary considerably from another, because they are authored by different sources. Therefore, one reasonable test we can make of our hierarchy matching algorithm is therefore to use the first and second halves of this reference list (and the text of the referenced pages) as two *alternative* characterizations of the topical area. We will refer to this partitioning of the data set as dmoz-nl1, dmoz-nl2, dmoz-ai1, and dmoz-ai2 below.

3.2 ACM

The ACM has maintained their own Computing Reviews Taxonomy as an indexing resource for decades. [3] This has been most recently revised in 1998 and contains eleven top level categories as shown in Figure 3.2.

While it appears possible to go from the ACM CR taxonomy to the text of ACM papers indexed by these categories, for the experiments in this paper we chose to rely upon only those short text fields used in the brief description of the categories as the text to represent each node. This provides fewer keywords per node than the other data sets and demonstrates some interesting properties for the ACM data sets.

Two versions of ACM taxonomy are used in the experiments. The first (which is known as acm2 below) consists of only the top level node and its 11 direct children. However, to provide additional text to be associated with each of these children, all the text forming rubrics for all its descendants are merged into a single textual passage associated with this category.[4] This gives us a tree with 12 nodes and a maximum depth of two.

The second data set from the ACM is constructed similarly, except that the structure for the top *three* levels is preserved instead of only the top two. This time all of the descendants below level three are promoted to be the text that describes the level three nodes. This yields a tree with 93 node and a maximum depth of three.

3.3 Encyclopedia Britannica

The data set that we used from Encyclopedia Britannica was taken from the text of an article written about computer science. The table of contents and the structure of the article are used to build the hierarchical information. The article is broken down into sections, subsections, and sub-subsections, and so on. Each section and subsection of the article were used as nodes of the *HGraph*, the section's title as the node's rubric and text from the section as its textual passage. The hierarchy contained 50 nodes, and had a maximum depth of 5. This data set is know as 'eb' in the experiments.

3.4 Semantics of topical hierarchies

The range of sources from which we have drawn our data sets – from an encyclopedic survey article, to the a general purpose Web index (DMOZ) to a

[2]The text was extracted from the web-pages using the lynx web browser with the '-dump' option. We did not use any text from inside the HTML tags (i. e. META tags). The data was then cleaned up by deleting error messages (i. e. server not found) , and deleting nodes that had no pages to describe them.

[3]This data can be found on the ACM web site at http://www.acm.org/class/1998/ccs98.txt.

[4]For example, for the category `HARDWARE`, the text of its subcategories `CONTROL STRUCTURES AND MICROPROGRAMMING, MEMORY STRUCTURES,` ... would be used, as well as the text associated with subcategories of `CONTROL STRUCTURES AND MICROPROGRAMMING`

Table 2: Data set statistics. i stands for identity test, -:not relevant (no overlap of categories), s: see symmetric experiment, BM: Bipartite match identifies correct match, SI: Subtree Isomorphism finds superior match

	acm2	acm3	dmoz-nl1	dmoz-nl2	dmoz-ai1	dmoz-ai2	dmoz-cs	eb
acm2	i	BM	-	-	-	-	-	-
acm3		i	-	-	-	-	??	??
dmoz-nl1			i	BM	s		-	-
dmoz-nl2				i	np	s	np	-
dmoz-ai1					i	p	-	-
dmoz-ai2						i	-	-
dmoz-cs							i	p
eb								i

journal's taxonomic classification system (ACM-CR) – requires us consider the semantics intended by these very different representations quite carefully.

The Open Directory project is an interesting example of how people naturally describe information in a hierarchical fashion. Volunteer editors are responsible for keeping lists of WWW pages that are *about* a given topic in which they have particular expertise. The resulting system of pages creates a hierarchy of topical pages, each pointing to large numbers of high quality WWW pages.

The general survey prose written in the EB article is designed to provide a tutorial and foundation for further reading. The bibliographic citations within this text (and not considered further in the current report) are therefore especially useful.

Some, and traditionally most, hierarchies have been a work of single authorship. Rarely is this author a single individual; typically there is an institution coordinating the activities of a group of (EB editors) or a specially appointed panel (ACM). One reason that DMOZ is of special interest in our work is that its "open" approach to the coordination of independent editors suggests new forms of consensual activity towards a common framework.

There are also some commonalties across these various sources. Several special categories, e.g., `GENERAL` and `MISCELLANEOUS`, often appear in taxonomies, and are found in both EB and ACM-CR. These present special problems for our HierDock procedure, since these sub-categories appear *syntactically* interchangeable with other sub-categories that in fact capture narrower topical scope. Similarly, DMOZ's `PEOPLE, CONFERENCES`, and `PUBLICATION` categories are used commonly across many categories. None of these capture the narrower-topical-area semantics we associate with typical child nodes.

4 Results

We anticipated that for the smaller hierarchies, or hierarchies that are of comparable sizes, that the weighted bipartite matching algorithm would do a com-

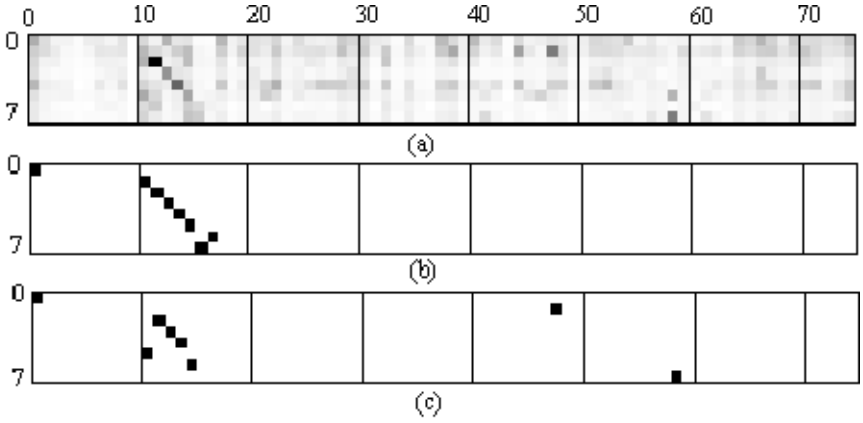


Figure 3: Results from experiment dmoz-nl1 vs. dmoz-ai2. In each of the plots dmoz-nl1 is on the vertical axis, and dmoz-ai2 is on the horizontal axis. The vertical lines are for ease of comparing the three plots. There are ten nodes between each set of lines. (a) The similarity score matrix. (b) The results from the attributed tree matching algorithm. (c) The results from the weighted bipartite matching algorithm. The two algorithms came up with the same match set for this experiment.

parable job.

In this section we present the results of selected experiments. A summary of the experiments conducted can be found in Table 2. All of the experiments were run on a Celeron 850MHz machine with 640M of RAM. The weighted bipartite matching algorithm was implemented in C, and the larger experiments took about one second to complete. The attributed tree matching algorithm was run in MATLAB, and the larger experiments took about one minute to find a solution with the QPOPT optimizer[5].

4.1 Comparing dmoz-* against itself

The simplest match considered is dmoz-nl1 vs. dmoz-nl2, since both hierarchies have exactly the same structure and the only variation results in the vocabulary sampling differences brought about by our arbitrary separation of the classified pages into two subsets. Both the weighted subgraph isomorphism (SI) and weighted bipartite match (BM) solutions successfully came up with the same matching, and it is the matching that was expected (`Artificial_Intelligence` node from both trees, the `Natural_Language` node from both trees, etc.).

4.2 dmoz-nl1 vs. dmoz-ai2

The matching of the dmoz-nl1 subtree against the fuller dmoz-ai2 tree provides a more interesting test. By construction, there is a known correct structural

match that should also yield the highest weighted match, but the appropriate "docking point" for the `Natural_Language` subtree is unknown to the algorithm. Figure 3 shows first the underlying node-pair similarity matrix, then the matches discovered by the SI and BM solutions. The strong diagonal match in columns 10-16 shows that the SI match successfully matched the subtree, while the BM included several other spurious pairs because it ignores structure it was free to match nodes in categories that fall in a different subtree.

A more rigorous test of our system is shown in the experiment of matching the two versions of the entire dmoz-AI subtrees, dmoz-ai1 and dmoz-ai2 against one another. While these two trees are almost identical, the splitting of the test set resulted in several nodes in one version that did not have corresponding nodes in the other. For example, dmoz-ai2 has an extra node `Ontologies` node (column 36 in 4) not found in dmoz-ai1; two other similar differences can be seen in columns 49 and 50. The central result by Figure 4 is that the straightforward pairwise BM method identifies many spurious node pairings while the SI method identifies exactly the matching we would expect. The two trees have exactly the same structure we would expect, except for the three small variations just mentioned.

4.3 eb vs. dmoz-cs

Of course using HierDock to match *heterogeneous* hierarchies, coming from different sources, provides the most realistic test of the practicality of the method. At the same time, identifying the "correct" matching between two independently-authored hierarchies also becomes a matter of opinion. Figure 5 shows the result of matching two very different types of topical hierarchy, the EB's encyclopedic entry on
`Computer_Science` against ACM's detailed classification of this same topic. As with the ai1 vs. ai2 match, BM returns a unstructured, unhelpful series of matching pairs. SI, however, found at least portions of a match that qualifies as at least interesting; the details of this match are presented in Appendix A. While large portions of both hierarchies were left unmatched (the largest clique returned by the SI algorithm contained only 25 nodes), and some matched rubrics may seem odd (e.g., EB's `RELIABILITY` vs. ACM's `database_management`), others seem very encouraging (e.g., EB's `THEORY` vs. ACM's `mathematics_of_computing`).

5 Discussion

When comparing the general structure of the DMOZ hierarchies to the structure of the ACM hierarchy, interesting observations can be made. In the DMOZ structure from the `ARTIFICIAL INTELLIGENCE` node on down, many of the nodes have a child called `PEOPLE`. Another node that is fairly common is a node called `CONFERENCES AND EVENTS`.[5] One reason that DMOZ has evolved to this struc-

[5] `ARTIFICIAL INTELLIGENCE, NEURAL NETWORKS, BELIEF NETWORKS`, and many other nodes have `CONFERENCES AND EVENTS` as a child.

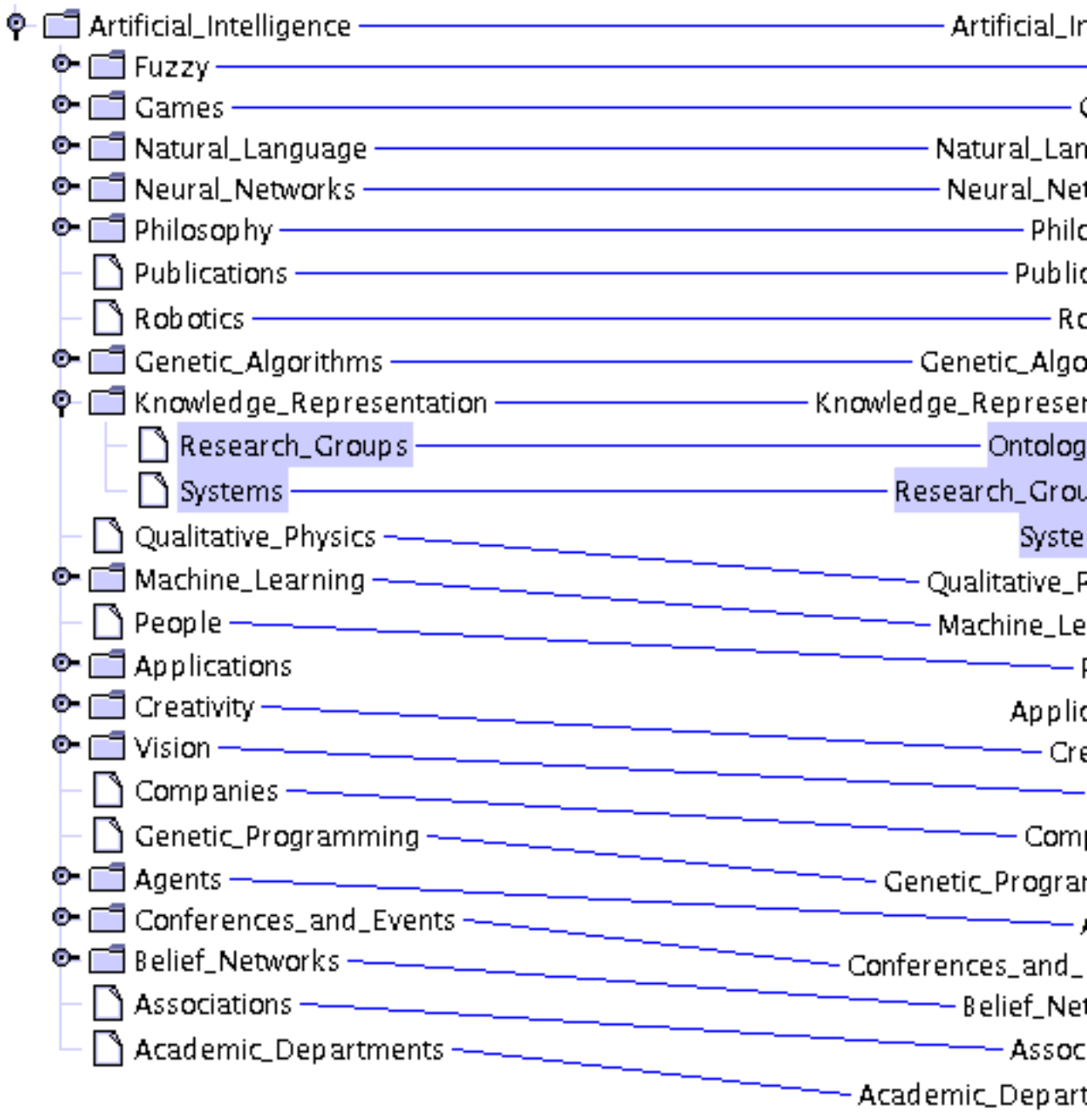
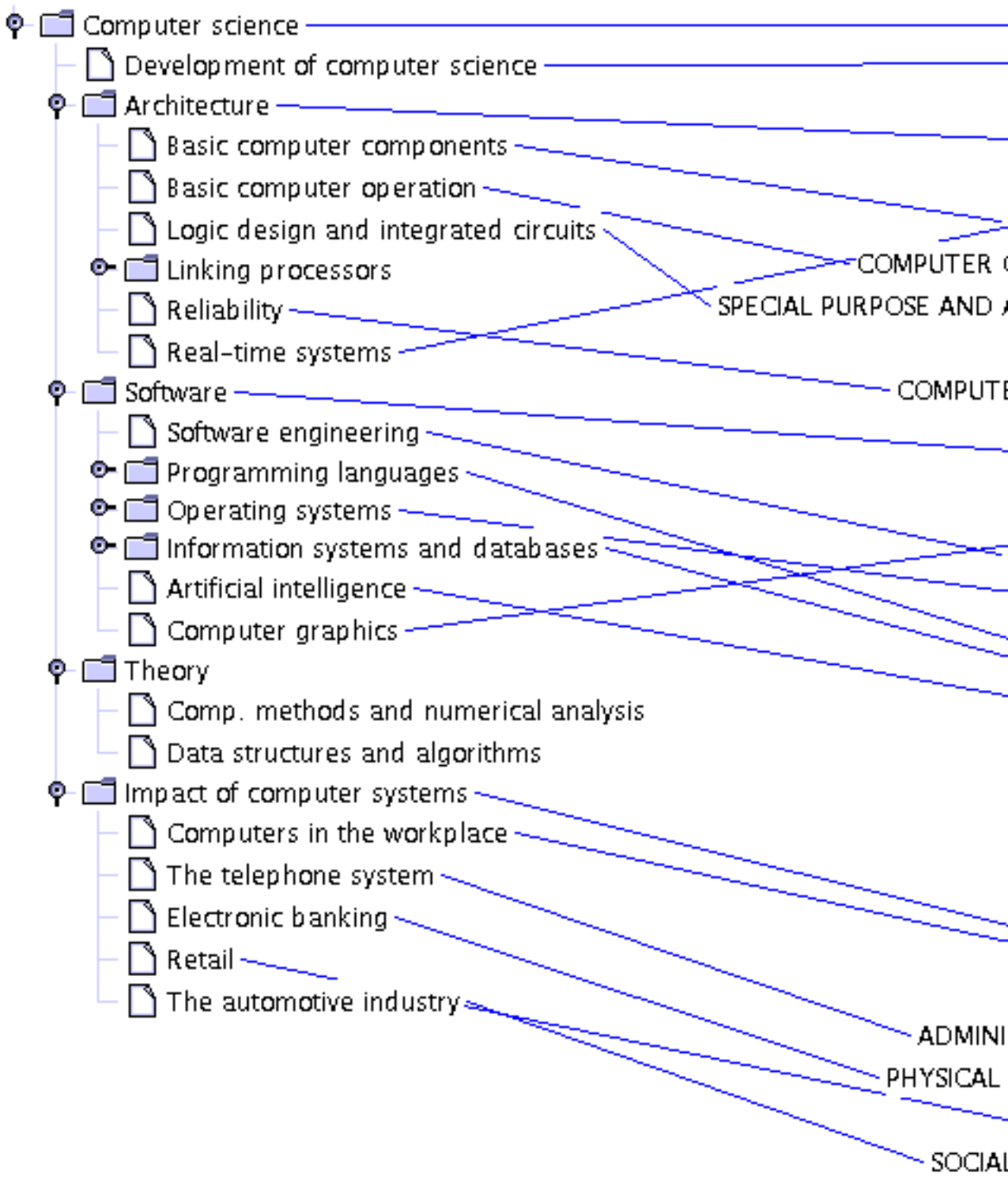


Figure 4: dmoz-ai1 vs. dmoz-ai2


Computer science
Development of computer science
Architecture
Basic computer components
Basic computer operation
Logic design and integrated circuits
Linking processors
Reliability
Real-time systems
Software
Software engineering
Programming languages
Operating systems
Information systems and databases
Artificial intelligence
Computer graphics
Theory
Comp. methods and numerical analysis
Data structures and algorithms
Impact of computer systems
Computers in the workplace
The telephone system
Electronic banking
Retail
The automotive industry
COMPUTER
SPECIAL PURPOSE AND
COMPUTE
ADMINI
PHYSICAL
SOCIAL
COM
CO

ture is that it has a lot of content and it is trying to create a niche for every piece of content that it has available to classify. The content that DMOZ deals with consists entirely of web pages. There are many people who create a home page so, having many `PEOPLE` nodes is potentially a good idea for the DMOZ, but does not fit into the BT/NT relation.

The ACM taxonomy, on the other hand, is the result of a group of experts defining what computing means. These experts do not have the handicap of being tied to the content that they have available to them. The ACM is not as fluid as DMOZ; DMOZ is updated on a daily basis while, the last classification from the ACM on computing is from 1998. It is important that the ACM experts get the classification correct, because it will not change for a while. The ACM experts know that computing will change somewhat over the course of a few years. They introduced categories like `General Literature` and `Miscellaneous` to allow for the general shift of interest in computing, between releases of their computing classifications. Column 54 of Figure 5 (a) is the `Conferences` node below the `Vision` node in the dmoz-ai2 hierarchy. This column is darker than most other columns, and can be recognized as a node with a high level-of-treatment. Another example of this is column 17 in Figure 5 (a); this is the `Neural Network` node. It is a node that has many children and it covers neural networks at a high level so it is shaded darker than most nodes.

Each of the rows represents a node from acm2, and those nodes are more general than most of the nodes of acm3 (the columns). Each square in the dark streak is more general node versus a more specific node.

More generally, we can expect nodes at higher levels of a topical hierarchy should be *about* more things. In the similarity plots shown above, these patterns of *aboutness* show up as a dark streaks, where one hierarchies broad term is nearly-uniformly distributed across another's narrower subtopics. We believe that this will be able to help to determine breadth of coverage for a particular topic. The nodes that are *similar* to many other nodes of a common topic, are most likely to be more general nodes. Recognizing patterns like this could help search engines to retrieve documents that are general to a certain topic. It would allow for someone to specify the level-of-treatment that they desire in a given query.

6 Conclusions and future directions

As mentioned in section 2.2 Pelillo et al. report results using replicator equations to optimize the set linear equations, and report that the basins of attraction of optimal or near optimal solutions are large [7]. In our experiments the basins of attraction to the optimal solutions do not appear to be extremely large in all cases, and will be explored further.

The weighted subtree isomorphism algorithm can not return a matching unless that matching is exactly an isomorphism. We are considering a diverse set of hierarchies, authored by different people. It is natural to believe that these

different hierarchies have differences in their respective shapes. These differences could lead to a "best" match that is not a true isomorphism, but something that might be "perturbed" to produce true isomorphism. We intend to explore "edit distance" heuristics that "correct" for differences between authors. (i. e. removing a node from one hierarchy or adding an extra node to another hierarchy) to yield a higher weight legal match.

6.1 Other applications

Beyond this direct application of HierDock technology, we have also become interested in several unanticipated utilities. The general variations in language use within and across topical areas has been a topic studied by Belew for some time [11]. In brief, we have shown that statistical variations in an a priori identified topical area are different than those across larger topical domains, and these differences can be used to identify better phrasal index items. Belew has discussed the use of "inside vs. outside" vocabularies as mechanisms to mediate search across multiple corpora [1, Sect. 3.3.1]. Our analysis of statistical variations in our topical hierarchies suggests new ways to identify such patterns.

A related goal is to identify level-of-treatment of textual passages. That is, how might we distinguish a general, overview tutorial from the union of a set of texts on individual topical periods? As discussed in Section 5 it may be possible to determine level-of-treatment, based on statistical analysis.

7 Conclusion

In summary, when matching two hierarchies the structural information inherent in the hierarchies convey meaning beyond that captured by the nodes considered as independent topical descriptors. Using an algorithm, such as weighted subtree isomorphism algorithm, that accounts for both hierarchical structure and node similarity, a better matching can be found, than with an algorithm that ignores this hierarchical structure.

Appendix A: EB vs. ACM3 match

Nodes of the EB and ACM3 hierarchy that were part of the maximum clique, as they were matched by HierDock. Nodes taken from the EB are preceeded by a 1 and appear in all capitals; corresponding nodes taken from ACM3 are preceeded by a 2 and are in lower case.

```
1 COMPUTER SCIENCE
2 computing
   1 DEVELOPMENT OF COMPUTER SCIENCE
   2 general literature
   1 ARCHITECTURE
   2 information systems
```

```
   1 BASIC COMPUTER COMPONENTS
   2 information interfaces and presentation
   1 BASIC COMPUTER OPERATION
   2 miscellaneous
   1 LOGIC DESIGN AND INTEGRATED CIRCUITS
   2 general
   1 LINKING PROCESSORS
   2 models and principles
   1 RELIABILITY
   2 database management (e.5)
   1 REAL-TIME SYSTEMS
   2 information storage and retrieval
1 SOFTWARE
2 computer systems organization
   1 SOFTWARE ENGINEERING
   2 general
   1 PROGRAMMING LANGUAGES
   2 processor architectures
   1 OPERATING SYSTEMS
   2 computer-communication networks
   1 INFORMATION SYSTEMS AND DATABASES
   2 special-purpose and application-based
   1 ARTIFICIAL INTELLIGENCE
   2 performance of systems
   1 COMPUTER GRAPHICS
   2 computer system implementation
1 THEORY
2 mathematics of computing
   1 COMPUTATIONAL METHODS AND NUMERICAL ANALYSIS
   2 numerical analysis
   1 DATA STRUCTURES AND ALGORITHMS
   2 miscellaneous
```

8 Acknowledgments

This work was supported in part with funding by Encyclopedia Britannica. Equipment used in this research was supported in part by the UCSD Active Web Project, NSF Research Infrastructure Grant Number 9802219. We would also like to thank Dr. Philip Gill for useful conversions concerning linear optimization techniques and his QPOPT package, Ben Shapiro for his luggable computational platform, and John Mill for early work on the project.

References

[1] R. K. Belew. *Finding Out About: A cognitive perspective on search engine technology and the WWW.* Cambridge Univ. Press, 2000.

[2] Y. Cheng, V. Wu, R. Collins, A. Hanson, and E. Riseman. Maximum-weight bipartite matching technique and its application in image feature matching. In *SPIE Conference on Visual Communication and Image Processing*, 1996.

[3] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Electrical and Computer Science Series. MIT Press, 1992. ISBN 0-262-03141-8.

[4] M. R. Garey and D. S. Johnson. *Computers and Intractability A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979.

[5] P. Gill, W. Murray, and M. A. Saunders. User's guide for QPOPT (version 1.0): a Fortran package for quadratic programming. NA 95-1, University of California, San Diego, 1995.

[6] D. S. Johnson and C. C. McGeoch, editors. *Network flows and matching: first DIMACS implementation challenge*, volume 12. 1993.

[7] M. Pelillo, K. Siddiqi, and S. Zucker. Matching hierarchical structures using association graphs, 1998.

[8] D. Rose and R. Belew. Legal information retrieval: a hybrid approach. In *Second Intl. Conf. on AI and the Law*, 1989.

[9] D. E. Rose. *A symbolic and connectionist approach to legal information retrieval*. Lawrence Erlbaum, Hillsdale, NJ, 1994.

[10] A. M. Steier. *Statistical semantics of phrases in hierarchical contexts*. PhD thesis, Computer Science & Engr. Dept. - Univ. Calif. San Diego, 1994.

[11] A. M. Steier and R. K. Belew. Exporting phrases: A statistical analysis of topical language. In R. Casey and B. Croft, editors, *2d Symposium on Document Analysis and Information Retrieval*, 1994.