Toward Interactive Learning by Concept Ordering

Shantanu Godbole[*]
IBM India Research Lab

Sachindra Joshi
IBM India Research Lab

Sameep Mehta
IBM India Research Lab

Ganesh Ramakrishnan
IBM India Research Lab

ABSTRACT

In this paper we present a visual education tool for efficient and effective learning. The toolkit is based on a simple premise: *simple concepts should be learned before advanced ones.* We propose algorithms to automatically capture such pre-requisite dependence relationships between concepts. We extract concept definitions from the web's hyperlinked environment and create a concept graph arranged in a hierarchical structure and presented to the user in an interactive fashion. Thereafter, the user guides the learning process in a hyperlinked environment, by selecting a target concept, exploring the associated learning graph, learning pre-requisite concepts, and repeating this process till her learning goal is reached. To measure usefulness and correctness of our approach, we conducted a user study with 25 users using precision and recall measures. Overall, the feedback from users was encouraging. We believe this is a positive step toward building user driven interactive learning systems.

1. SYSTEM DESCRIPTION

1.1 **Motivation:** Traditionally, individuals have relied on books, newspapers and other printed material to learn. However, recent past has seen a radical shift in the learning paradigm with the advent of the Internet. In the presence of efficient and accurate search engines, a user searches a target concept on the web and learns it by following relevant (typically top 10) results. This method indeed looks attractive but it unfortunately lacks the structure provided by printed sources. For example, concepts in printed books are arranged in increasing order of difficulty and maturity. Any such ordering is missing from the Search–and–Learn paradigm. While reading about a concept using online resources, the user has to follow hyperlinks and navigate back and forth between related, underlying or pre-requisite concepts. This navigation is ad-hoc without any principled ordering. However, online learning supports very effective browsing between concepts which is not possible in books. *In this paper, we develop a visual educational tool which aims to leverage the positive features of online and print-based approaches while mitigating their negative aspects and helps the user learn effectively and efficiently.*

The tool is based on simple observations about learning: i)learning any (fairly) advanced concept involves learning pre-requisite concepts and ii) outside of the classroom settings, individuals prefer focused learning and would not like to spend time on already known (redundant) or peripheral (non-relevant) concepts. We contend that both these criteria can be satisfied by arranging the concepts in a hierarchical graph-like structure. To be more precise, there is a partial ordering between concepts, which corresponds to a directed acyclic graph (DAG). We employ graph theoretic algorithms to construct these concept graphs automatically, mining and extracting them from the web. The concept graphs are presented to the user in an interactive environment, where the user can see all the relevant concepts and can choose to explore non-redundant (previously unknown) concepts in detail, while ignoring known concepts.

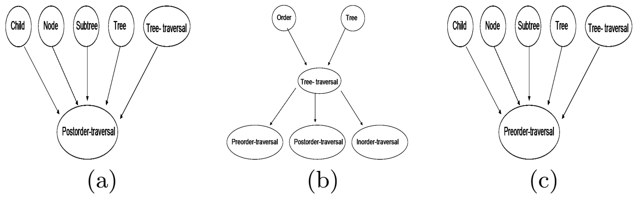
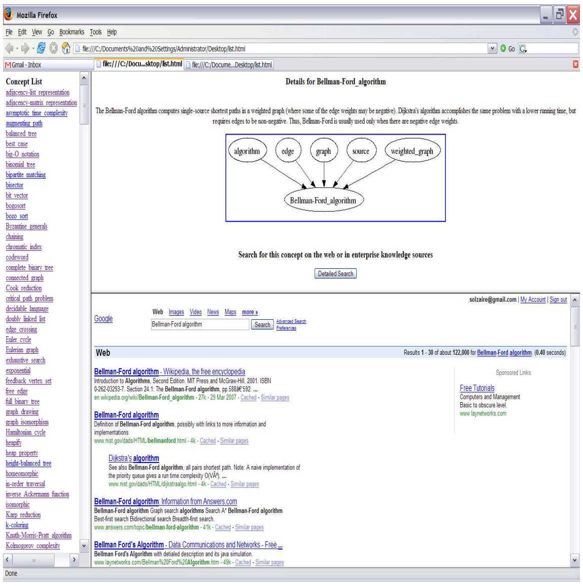


Figure 1: Example showing the concept graphs for types of tree traversal

Before providing the brief summary of the algorithms, we present a concrete example. Assume that the user wants to learn post-order traversal. The corresponding learning graph is shown in Figure 1(a). However, by inspecting the graph, the user soon realizes that she needs to understand tree traversal. She clicks on the tree traversal node and the learning graph shown in Figure 1(b) is presented. The user learns tree-traversal and realizes that she can also learn pre-order and in-order traversal without learning any new concepts. Figure 1(c) is displayed when she selects pre-order traversal. This expository example illustrates the usefulness of our system, wherein the user needs an interactive environment to learn the target concept (post-order traversal) and at the same time also realizes that extra concepts can be learned without much effort.

[*] {shgodbol,jsachind,sameepmehta,ganramkr}@in.ibm.com

Hypertext '07 Manchester, United Kingdom


Mozilla Firefox
File Edit View Go Bookmarks Tools Help
file:///C:/Documents%20and%20Settings/Administrator/Desktop/list.html
Go
Gmail - Inbox
file:///C:/Docu...sktop/list.html
file:///C:/Docume...Desktop/list.html
algorithm
edge
graph
source
weighted_graph
Bellman-Ford_algorithm
Detailed Search
solzaire@gmail.com | My Account | Sign out
Google
Web Images Video News Maps more »
Bellman-Ford algorithm
Search
Advanced Search
Preferences
Web
Sponsored Links
Done

Concept List

adjacency-list representation
adjacency-matrix representation
asymptotic time complexity
augmenting path
balanced tree
best case
big-O notation
binomial tree
bipartite matching
bisector
bit vector
bogosort
bozo sort
Byzantine generals
chaining
chromatic index
codeword
complete binary tree
connected graph
Cook reduction
critical path problem
decidable language
doubly linked list
edge crossing
Euler cycle
Eulerian graph
exhaustive search
exponential
feedback vertex set
free edge
full binary tree
graph drawing
graph isomorphism
Hamiltonian cycle
heapify
heap property
height-balanced tree
homeomorphic
in-order traversal
inverse Ackermann function
isomorphic
Karp reduction
k-coloring
Knuth-Morris-Pratt algorithm
Kolmogorov complexity

Details for Bellman-Ford_algorithm

The Bellman-Ford algorithm computes single-source shortest paths in a weighted graph (where some of the edge weights may be negative). Dijkstra's algorithm accomplishes the same problem with a lower running time, but requires edges to be non-negative. Thus, Bellman-Ford is usually used only when there are negative edge weights.

Search for this concept on the web or in enterprise knowledge sources

Results 1 - 30 of about 122,000 for Bellman-Ford algorithm. (0.40 seconds)

Bellman-Ford algorithm - Wikipedia, the free encyclopedia
Introduction to Algorithms, Second Edition. MIT Press and McGraw-Hill, 2001. ISBN 0-262-03293-7. Section 24.1: The Bellman-Ford algorithm, pp.588â€"592. ...
en.wikipedia.org/wiki/Bellman-Ford_algorithm - 27k - 29 Mar 2007 - Cached - Similar pages

Bellman-Ford algorithm
Definition of Bellman-Ford algorithm, possibly with links to more information and implementations.
www.nist.gov/dads/HTML/bellmanford.html - 4k - Cached - Similar pages

Dijkstra's algorithm
See also Bellman-Ford algorithm, all pairs shortest path. Note: A naive implementation of the priority queue gives a run time complexity O(VÂ²), ...
www.nist.gov/dads/HTML/dijkstraalgo.html - 4k - Cached - Similar pages

Bellman-Ford algorithm: Information from Answers.com
Bellman-Ford algorithm Graph search algorithms Search A* Bellman-Ford algorithm Best-first search Bidirectional search Breadth-first search.
www.answers.com/topic/bellman-ford-algorithm - 41k - Cached - Similar pages

Bellman Ford's Algorithm - Data Communications and Networks - Free ...
Bellman Ford's Algorithm with detialed description and its java simulation.
www.laynetworks.com/Bellman%20Ford%20Algorithm.htm - 49k - Cached - Similar pages

Free Tutorials
Computers and Management
Basic to obscure level.
www.laynetworks.com

Figure 2: Snapshot of the web based visual tool

1.2 Algorithm: Let C be a list of N concepts, and let $C[i]$ refer to the the i^{th} concept in the list. *We assume the availability of a database of concept definitions for a given domain*[1]. Using this definition database, we construct the concept graph C_{DAG} for a particular domain. Our algorithm for constructing the concept DAG consists of two steps. In our first step, we construct a graph C_{graph} by making use of references between concepts in concept definitions. In the second step, we extract a concept DAG C_{DAG} from C_{graph}.
Step 1: We generate a weighted and directed concept graph $C_{graph} = (V, E)$ as follows. For every concept $C[i], 1 \leq i \leq N$, we create a vertex v_i in V. An edge is introduced from v_i to v_j iff concept $C[j]$ is mentioned in some definition of $C[i]$. We say that 'concept C_1 is mentioned in some text' *iff* the lemma form of C_1 matches the lemma form of some token in the text. The weight w_{ij} of an edge (v_i, v_j) is set to the number of mentions of $C[j]$ in the definitions of $C[i]$.
Step 2: Extracting a DAG C_{DAG} from C_{graph} requires the elimination of cycles from C_{graph}. There could be several sets of edges whose removal would lead to elimination of cycles. In our case of a weighted digraph, it makes sense to remove the set of edges with minimum total weight. The problem of removing minimum number of edges in a digraph is a standard problem studied in literature as the *minimum feedback arc-set problem* (FAS) [4]. Unfortunately, the problem is NP-Hard [3]. However, due to the wide applicability of the FAS problem, several heuristics have been proposed for it. For our task, we use a variant of a fast, effective and simple heuristic for FAS proposed by Eades, *et. al.* [1].

1.3 Interactive Visual Interface: Next, we outline the key components of our web-based interface. A snapshot of the interface is shown in figure 2. A ***concept list*** is the list of all concepts in a domain of interest. The list is displayed in the left pane of our tool. The user can select the concept of interest from the list and its associated learning graph is displayed on the right. ***Learning Graph*** shows the concept selected from the concept list and it's pre-requisite nodes. In addition to these ancestor nodes, we also show one level of descendants from our concept index. These are nodes which need the target concept as their pre-requisite. The learning graph is shown in top pane. The nodes of the graph are clickable and can also be used to select the target concept. The process is the same as choosing a concept from the concept list. ***Concept Definitions*** are the actual extracted definitions of the concept shown above the graph. The bottom pane shows the concept's web query search results.

1.4 Overall Process: The concept list is obtained from either online glossaries or indexes present at the back of books. Corresponding concept definition database is built. The list and database are used to derive the concept DAG. This process is executed offline. The list is populated in the visual interface. The user selects a concept from the concept index and the associated learning graph is presented. Next, the user identifies the pre-requisite concept to be learned and clicks on the corresponding node. A new learning graph is displayed in real time. In order to maintain the context sensitivity of the learning graph, the parent node (original target to be learned) is always displayed. In our user study we found that this property helps the user stay focused and reduces navigational time. Once the user has learned all pre-requisites, she reads the definition of the target concept. Subsequently, based on her interest, the user may search other data sources and refine or broaden her understanding.

1.5 Experiment: We designed and conducted a user study with 20 participants to evaluate the quality of our concept graphs. Every user was presented with learning graphs corresponding to 15 concepts and asked to mark redundant (false positive) and missing edges (false negatives). These numbers are used to calculate precision and recall. The average precision and recall were found to be .80 (standard deviation of .16) and .84 (.12) respectively.

1.6 Conclusions: In this paper, we presented an interactive visual toolkit to facilitate the learning process by capturing the pre-requisite dependences between concepts in learning domains. These dependences are modeled hierarchically and presented to the user. She can recursively learn pre-requisite concepts to thoroughly understand target concepts of interest according to learning goals. We evaluated our approach by conducting a moderately large user study using the widely accepted precision and recall metrics to measure correctness and usefulness respectively. Overall, the users found the toolkit to be very useful for learning concepts in a principled and organized fashion. Currently, we are exploring algorithms to incorporate each user's existing skill-set to set up a domain specific baseline, which will further result in focused learning and reduce learning time.

2. REFERENCES

[1] Peter Eades, Xuemin Lin, and W. F. Smyth. A fast and effective heuristic for the feedback arc set problem. *Information Processing Letters*, 47(6):319–323, 1993.

[2] Google. Advanced google search operators, 2002.

[3] R. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Communications*, pages 85–103, 1972.

[4] P. Slater. Inconsistencies in a schedule of paired comparisons. *Biometrika*, 48:303–312, 1961.

[1] We used Google's "define:" query operator [2] to build the database given an index of concepts on the web