Building industry-specific knowledge bases using concept graphs: Methods and Applications

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

Searching and learning are two critical tasks undertaken in fulfilling an individual's information need in an organization. In the context of CRM or service oriented organizations like contact centers, applications like searching appropriate knowledge bases, and learning arbitrary concepts of interest are important building blocks in interacting with customers. In this paper we present two applications that exploit custom built domain specific knowledge bases. First, we delve into knowledge bases built as concept graphs. We propose algorithms to automatically capture pre-requisite dependence graphs among the concepts. We present a visual interactive tool for efficient and effective targeted learning. Second, we present a ranking framework built on top of these knowledge bases that ranks documents. This ranking can be configured to return simpler documents before advanced ones to aid quick learning of concepts for service personnel. The user guides the learning process, by selecting a target concept, exploring the associated learning graph, learning pre-requisite concepts, and repeating this process till the learning goal is reached. The ranking framework we show is customizable to return different documents based on the application need. We measured the usefulness and correctness of the concept graph built using our approaches with a user study involving 25 users. We believe that this effort is a positive step toward exploiting domain-specific knowledge bases in targeted learning scenarios relevant in services scenarios.

1. INTRODUCTION

In service arms of organizations including contact centers and L1/L2 support desks, QoS metrics employed are: percentage of problems solved successfully and the average response time per customer. The same metrics are also used to measure the effectiveness and efficiency of individual agents. However, both these metrics are somewhat conflicting in nature. Obtaining low response time with high success rate is extremely difficult. Some of the reasons for this are i) inability of the customer to precisely state the problem, ii) lack of knowledge on the agents part, and iii) lack of proper databases and interfaces using which the agent can learn more about the customer's query. The first problem cannot be completely solved using technology. The second problem requires the organization to invest heavily in agent training programs. We contend that solving the first two problems completely is implausible (if not impossible). Businesses cannot control customers and owing to the rapid shifts in technology and products, comprehensive training to adjust for each change in not feasible. In this paper, we focus on the third problem, *i.e.*, to design domain specific databases and interfaces which the agent can use while helping the customer. The databases will be useful when the agent has to quickly learn pre-requisite/related concepts which will aid him in helping the customer (who may be on hold on phone) effectively.

In this paper we outline a method for building domain specific knowledge bases and search interfaces that aid (1) in quickly searching relevant information and (2) the dynamic learning process. The domain specific knowledge base is represented as a graph along with a large body of annotated text data. An directed edge from concept A to B implies that A is a pre-requisite for learning B. To be more precise, there is a partial ordering between concepts, which corresponds to a directed acyclic graph (DAG). We present algorithms and heuristics based on graph theoretic principles to derive a learning DAG, given a set of concepts and respective definitions. The algorithms automatically capture and model the pre-requisite dependencies among concepts. Next, we highlight two applications where availability of such learning DAGS will be extremely helpful. The applications we focus on are *Education Portals* and *Search applications for agents in CRM organizations.*

Web based educational portals typically support an on-line chat facility wherein the customer can chat directly with an instructor. In such cases exploiting concept graphs can tremendously boost student satisfaction by facilitating structured learning. Consider a simple example, wherein an instructor is trying to teach *post-order traversal* to a student. The learning DAG shown in Figure 1(a) is presented to the instructor (who may not be a subject matter expert). By inspecting the graph, the instructor soon realizes that she will need to explain (to the student) tree traversal. She clicks on the tree traversal node and the learning graph shown in Figure 1(b) is presented. The agent explains tree-traversal

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

and realizes that she can also teach pre-order and in-order traversal without teaching any new concepts and therefore very low overhead is incurred on both ends. Figure 1(c) is displayed when she selects pre-order traversal. It is clear from this example, that in a practical setting, the usefulness of the concept graph can be very limited without effective presentation. To handle this problem, we design a web-interface, using which the instructor can very quickly view the concept graph. The instructor can also explore pre-requisite and related concepts. Finally, in another incarnation the portal may allow the student to directly interact with the system. This expository example illustrates the usefulness of our system, wherein the instructor uses the learning DAG and a hyper-linked environment to help the student learn better.

The second application is ranking results of searches in the context of service arms of organizations, contact centers, and help-desks. Technical help sites for JAVA, SAP, Oracle, *etc.*, use a search procedure to list all documents given a user query/concept. Typically, the search methods used are based on algorithms that exploit hyper-link structure of documents and support exact/approximate key word matching. The agent searches a target concept on the web (Google or Wikipedia for example) or proprietary enterprise knowledge bases and learns it by following relevant (typically top 10) results. Even if the algorithms perform extremely well in achieving high values for traditional IR metrics like precision and recall, they may not reflect the best choice in scenarios outlined above. The agent has to navigate back and forth between related, underlying or pre-requisite concepts. This navigation is ad-hoc, without any principled ordering.

Even though printed sources solve some of these problems, they pose different kind of difficulties like storage and number of copies needed (sharing books is not feasible in online help sites). We contend that in technical help sites, better domain specific algorithms to rank the search results can be developed. We strongly believe that a natural ordering of web pages is from simple (easy to learn) documents to difficult (hard, needing pre-requisite knowledge) documents. We assign a "hardness" score to a web page by projecting it onto the learning DAG described above. For expository purposes, suppose a contact center technical support agent has to answer a customer's query about installing a software product. The agent needs to quickly help the customer identify if certain pre-requisite software is available on her system. This software may in turn have other pre-requisites, possibly specific hardware, its drivers and so on. In this example scenario, the learning DAG is a software dependency graph with associated searchable product manuals so that the agent can quickly guide the end-to-end installation process.

To summarize, the key contributions of this paper are:

1. We present automatic algorithms to mine a graph structure, given a list of concepts with their associated definitions. Problematic scenarios like presence of cycles are handled to remove ambiguities. These derived graphs can then be browsed in different ways to assimilate different forms of information. This concept graph along with its associated document collection comprises a knowledge base for our applications.

2. We propose an interactive visual interface allowing the users to select, explore and learn concepts. The interface supports web and enterprise search. The toolkit supports the traditional *zoom, filter and details–on–demand* paradigm.

3. We propose a family of ranking functions which are aware of the mined concept graph. Sample functions shown can return either documents containing pre-requisites of a target concept or explaining the target concept in great detail. This component neatly fits the quick targeted learning scenario.

4. We empirically demonstrate the correctness and usefulness of our algorithms for two domains, *viz.*, "algorithms" and "particle physics". We conducted a user study with 25 users and 15 target concepts. High precision and recall were observed demonstrating the correctness and usefulness of the proposed system.

The rest of the paper is organized as follows. In section 2 we present the methods and algorithms used to mine the concept graph from a collection of documents. Section 3 presents the family of ranking functions aware of this concept graph for different applications. In section 4 we show the visual interface we propose to browse and use the knowledge base constructed as this concept graph with its document collection. We evaluate the correctness and usefulness of the concept graph in section 5. We review some related work in section 6 and conclude in section 7.

2. BUILDING CONCEPT GRAPHS

In this section, we build domain specific knowledge bases for our target applications. One of the main components enabling our proposed applications is the concept graph modeled as a DAG. This concept graph along with the associated documents for each concept forms the knowledge base. We would like to note here that a concept graph for a given domain can be hand crafted and tuned by experts. However constructing a large comprehensive graph is bound to be tedious, error-prone and hard. We chose to focus on deriving the graph automatically and developing the associated applications. Profile and collaborative filtering based methods form part of our future initiative. We aim to start with a simple list of concepts and use rich knowledge sources like the web or documents within an enterprise to automatically determine the concept graph. The concept graph can subsequently be customized by a human expert.

Our learning application aims to structure the learning process. We make some simple observations about learning. First, learning any (fairly) advanced concept involves learning pre-requisite concepts. Second, 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. Therefore, there is a strong need to arrange the concepts to satisfy the learning goal in a time effective manner.

We design a *web based interface*, which the user can interact with and learn. When a user selects a concept, its associated graph is displayed. The user can then explore related or pre-requisite concepts at the click of a button. At each click, a new graph is displayed to the user, based on her current

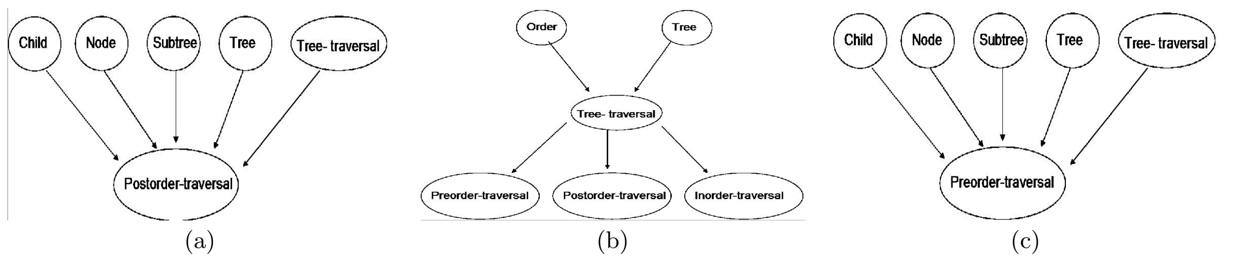


Figure 1: Figures depicting the browsing of a part of the concept DAG for the *algorithms* domain. Figure (a) shows the pre-requisites for the concept 'post-order traversal', one of which is 'tree traversal'. If one follows the pre-requisites for 'tree traversal', one comes across the learning graph in Figure (b). One can also see in Figure (b) that one of the siblings of 'post-order traversal' is 'pre-order traversal. One can further browse the pre-requisites for 'pre-order traversal', as in Figure (c).

browsing context. We describe this property in later sections. Finally, to take complete advantage of various data sources like web and enterprise data, we support dynamic querying, *i.e.*, the user can select a concept and a set of data sources. These data sources will be polled to find more information about the concept of interest.

Figure 2 schematically describes our proposed system. The main components of the system are:

- **Structure Derivation:** This component primarily deals with transforming the raw data (definitions of concepts) into structures. The process starts with analyzing definitions of concepts and automatically learning the dependence between the concepts. The concepts are then represented by nodes and the dependence between two concepts is modeled by a directed edge between the concepts. Deriving structure in this simplistic fashion introduces ambiguities (presence of cycles and multiple paths between a pair of nodes). The solutions to these ambiguous cases amounts to solving NP-complete problems in graph theory. Therefore, we use well-known heuristics to overcome these problems. Apart from being dependence aware, the final structure has the following two properties. First, browsing the structure in top-down fashion results in learning basic (fundamental) to difficult (advanced) concepts. This property provides directional and navigational cues to the user. The second property is that the concepts at same height can be learned in any order. This property coupled with knowledge about ancestors and descendants provides the user with a list of related concepts.

- **User Interface:** The final derived structure is visualized for the learning process. We built a web interface to leverage other sources of on-line information. An alphabetical list of concepts (C_L) is presented to the user. The user can select the concept to be learned. An associated concept tree is presented to the user. The nodes in the tree correspond to parents, children and related concepts of C_L. The pre-requisites for learning C_L and C_L itself are pre-requisite for learning more advanced concepts. *We chose to display this particular set of concepts to educate the user about extra related/parent concepts which can be learned by spending small amount of extra time.* We discuss this aspect in details in subsequent sections. The user can interact with the tree by selecting other concepts and exploring associated concept trees. Currently, our tool is not customizable for individuals. However, existing profile based algorithms can be plugged into our tool and be extremely beneficial. This integration forms part of our future initiatives.

- **Learning and Searching:** The user interacts with the visual interface to explore concepts. Once the user has learned fundamental concepts, she can move to related and advanced concepts. Since this is a web-based tool, we also provide her with the capability to search on-line, enterprise data and knowledge repositories. The user can configure various data sources and the associated query and result formats. The results can then presented to the user for further exploration. The user iterates between second and third components to accomplish set learning goals.

2.1 Constructing the concept graph

Let C be a list of N concepts, and let $C[i]$ refer to the the i^{th} concept in the list. The task is to construct a browsable concept DAG (directed acyclic graph) 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}.

Let C be a list of N concepts, and let $C[i]$ refer to the the i^{th} concept in the list. The task is to construct a browsable concept DAG (directed acyclic graph) 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}.

We assume the availability of a database of concept definitions for the given domain. Using this definition database, we construct the concept graph C_{graph} for that domain. For several domains, definition databases exist in the form of glossaries[1] In our experiments, we used the "define:" query

[1]Some glossaries can be found at `http://www.`

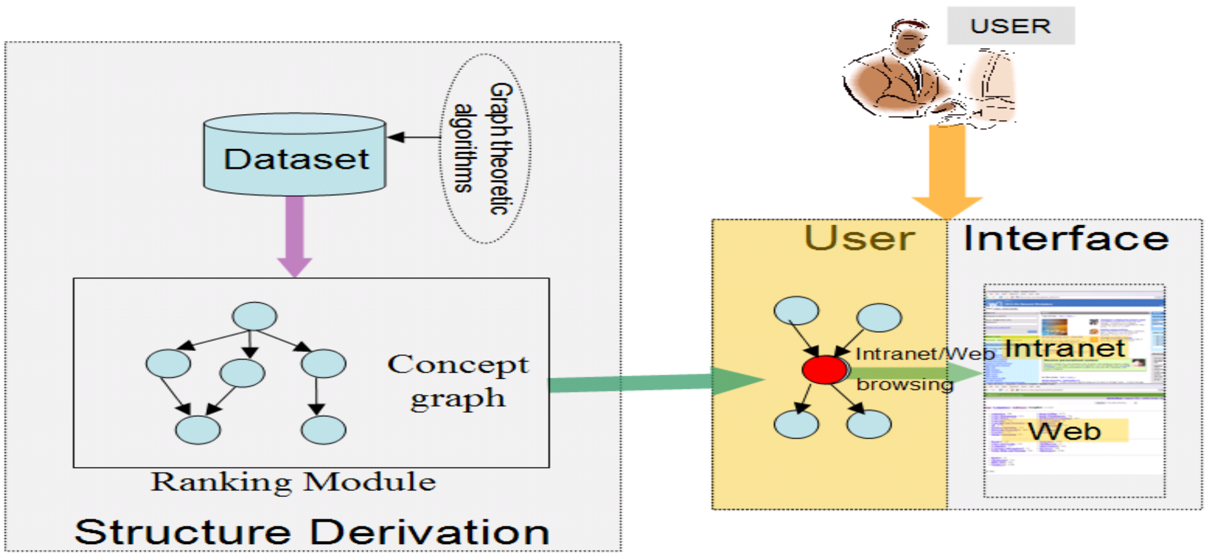


Figure 2: Schematic Overview of our Approach

operator provided by the Google search engine. A query operator causes a keyword to be interpreted differently and produces a specialized output as defined by the operator [6]. The result of a define query consists of a bulleted list of possible definitions of the given term or phrase gathered from multiple sources.

Figure 3 lists a subset of the definitions obtained using the define operator for the concept 'relation'. Typically, higher-level concepts are defined using lower-level concepts. We use this property of definitions in generating the concept DAG. In Figure 3, occurrences of other concepts from the list C are marked in rectangular boxes. The occurrence of a concept in a definition is determined based on the lemma forms of the concept and its occurrence.

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 $C[i]$.

In the Figure 3, concepts such as 'set' and 'order' are lower-level concepts when compared to 'relation'. However, the shaded concepts such as 'binary relations', 'predicate' and 'logic' are not necessary for the understanding of 'relation'. In fact, on the contrary, to understand the notion of 'binary relation', one needs to first understand the concept of 'relation'. The definition of 'binary relation' obtained from Google's define operator refers to the concept 'relation'. Thus, it is evident that the graph C_{graph} could contain cycles. However, what we require for learning concepts is a concept DAG. We next describe our approach for extracting a concept DAG C_{DAG} from C_{graph}.

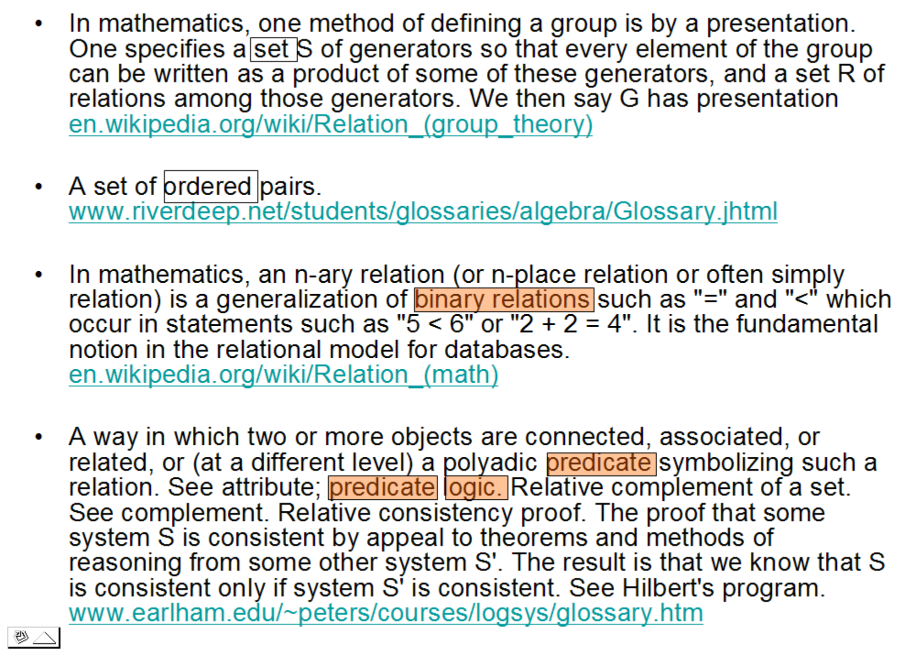
`interactions.org/cms/?pid=1002289`, `http://www.nist.gov/dads/`, `http://en.wikipedia.org/wiki/Particle_physics` and of course, WordNet [9].

2.2 Extracting concept DAG

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. It is intuitive to choose the set of edges with minimum cardinality as this would cause minimal change with respect to the original graph. 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) [11]. Unfortunately, the problem is NP-Hard [7]. 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.* [5].

Suppose the vertices of C_{graph} are ordered in some sequence $S = \{v_1, v_2 \dots v_n\}$. We call an edge $(v_i, v_j) \in E$, a forward edge, iff, $i < j$. Otherwise, we call it a backward edge. The graph G consisting of all forward edges, is an acyclic sub-graph of C_{graph}. Additionally, if the introduction of a backward edge e does not create cycles in G, then e could be added to G.

Thus, following Eades, *et. al.* [5] we pose our problem as that of determining a vertex ordering $v_1, v_2 \dots v_n$ that minimizes the sum of weights $\sum_{(v_i,v_j)\in E}^{j<i} w_{ij}$. Once a 'good' vertex ordering is determined, we construct our DAG following the steps outlined above. Figure 4 outlines our algorithm that takes as input a graph C_{graph} and outputs a vertex sequence s. For a vertex v, let $d(v) = w^+(v) - w^-(v)$, where $w^+(v)$ is the sum of weights of out-going edges from v and $w^-(v)$



- In mathematics, one method of defining a group is by a presentation. One specifies a set S of generators so that every element of the group can be written as a product of some of these generators, and a set R of relations among those generators. We then say G has presentation
  en.wikipedia.org/wiki/Relation_(group_theory)

- A set of ordered pairs.
  www.riverdeep.net/students/glossaries/algebra/Glossary.jhtml

- In mathematics, an n-ary relation (or n-place relation or often simply relation) is a generalization of binary relations such as "=" and "<" which occur in statements such as "5 < 6" or "2 + 2 = 4". It is the fundamental notion in the relational model for databases.
  en.wikipedia.org/wiki/Relation_(math)

- A way in which two or more objects are connected, associated, or related, or (at a different level) a polyadic predicate symbolizing such a relation. See attribute; predicate logic. Relative complement of a set. See complement. Relative consistency proof. The proof that some system S is consistent by appeal to theorems and methods of reasoning from some other system S'. The result is that we know that S is consistent only if system S' is consistent. See Hilbert's program.
  www.earlham.edu/~peters/courses/logsys/glossary.htm

Figure 3: Example definitions for the concept 'relation' as obtained using the Google 'define' operator

is the sum of weights of in-coming edges from v . The algorithm greedily removes vertices (and their incident arcs) in the following order (i) all sinks, (ii) all sources and (iii) vertex with the largest value of $d(v)$.

```
procedure FindBestSequence(Input:$C_{graph}$, Output:$S$)
$S_1 \leftarrow \phi$, $S_2 \leftarrow \phi$;
while $C_{graph} \neq \phi$ do
  while $C_{graph}$ contains a sink do
    Choose a sink $u$; $S_2 \leftarrow u.s_2$; $C_{graph} \leftarrow C_{graph} - u$;
  end while
  while $C_{graph}$ contains a source do
    Choose a source $u$; $S_1 \leftarrow S_1.u$; $C_{graph} \leftarrow C_{graph} - u$;
  end while
  Choose a vertex $u$ for which $w^+(u) - w^-(u)$ is maximum;
  $S_1 \leftarrow S_1.u$; $C_{graph} \leftarrow C_{graph} - u$;
  $S \leftarrow S_1.S_2$
end while
```

Figure 4: A heuristic for determining a vertex sequence S that minimizes the number backward edges in C_{graph}.

In Figure 5, we outline the algorithm that takes as input, a sequence of vertices S and outputs a DAG C_{DAG} that has the same set of vertices as C_{graph} and a subset of its edges $E_{DAG} \subset E$ determined by the sequence S. The algorithm first includes into E_{DAG}, all edges from E that are forward with respect to the sequence S. Next, it iterates over all other edges in E and includes an edge into E_{DAG} if this inclusion does not create a cycle with existing edges in E_{DAG}.

3. RANKING DOCUMENTS

In this section, we present an application built on the knowledge base mined previously. We define a framework of customizable ranking functions that rank documents in the context of the mined knowledge base (concept graph).

```
procedure FindConceptDAG(Input:$S$, Output:$C_{DAG}$)
Let $S = \{v_1, v_2, \ldots, v_n\}$
$E_{DAG} \leftarrow \{(v_i, v_j) | (v_i, v_j) \in E, i < j\}$; $C_{DAG} = (V, E_{DAG})$;
for each $(v_i, v_j) \in E, i > j$ do
  if $(v_i, v_j)$ does not form a cycle with edges in $E_{DAG}$ then
    Add $E_{DAG} \leftarrow E_{DAG} \cup (v_i, v_j)$
  end if
end for
```

Figure 5: The algorithm for extracting a concept DAG C_{DAG} from C_{graph}, using the vertex sequence S determined using the algorithm in Figure 4.

It is common to get hundreds of thousands and even millions of documents in some cases as a response to a search query. The intention of a searcher can be modeled to fulfill certain information needs. In particular, we address domains where learning concepts is the intention of the searcher. The huge number of returned documents make the problem of finding a relevant document from this set quite difficult. Ranking of documents has been one of the most popular way to deal with this problem. The returned set of documents are sorted in the order of their *relevance* and thus the most relevant document is placed on the top. To do this, a *relevance metric* is used to score each document that describes its relevance to the query. Different search engines employ different relevance metrics. The relevance metric *pagerank* has been found quite useful and popular for general Web search. However, we argue that for several domains a pagerank based scoring is not sufficient and different relevance metrics need to be defined based on the objective of the search.

In this paper, we propose a family of relevance metrics which is useful for educational portals, technical help sites or even for a general Web search where the intention of search is to learn a new concept. We use the concept graph to score each returned document. The concept graph for a domain can be learned using the method described in the Section 2. In the following subsections, we first present different examples of ranking objectives that are useful in education domain. We would then propose a general framework based on concept graph that can be used to define appropriate relevance metrics.

3.1 Ranking Objectives for Education

Searching a target concept on the Web, Intranet or on an educational portal is a very popular way of learning new concepts. The returned documents are ranked in a list. Different users would be at different levels of their learning and they may have different likings. Thus a general method of ranking results may not suffice. Consider the case where a user query consists of a concept C_i and the set D_i denote the set of returned documents. In the following list, we present three different ranking objectives for D_i.

- A novice user may find a document that describes the pre-requisite concepts of C_i first, and then describes the concept C_i, the most relevant. Therefore, the documents in D_i should be ranked such that documents that describe some or all of the pre-requisite concepts and then the concept C_i should be scored higher.
- A more exploratory user may not only want to learn the concept C_i but also other concepts that can be derived or learned using the concept C_i.
- A user may require a comprehensive document that not only describes the concept C_i but also describe the pre-requisite concepts and more advanced concepts for C_i.

3.2 Ranking functions

In this section we present a general framework that can be used to realize any of the ranking objectives given in Section 3.1. The framework uses the concept graph that can be learned as described in Section 2. Let the set $D = \{d_1, d_2, \ldots, d_n\}$ denote the returned set of documents for a concept C.

3.2.1 Overview

Figure 6 presents an overview of the ranking framework. The framework consists of two main components. The first component is a *concept spotter* that annotates all the concepts that occur in a document $d_i \in D$. The second component is called a *document scorer* that takes the annotated document and give a score for the document based on the annotated document and position of annotations. We describe both the components in the following sections.

3.2.2 Concept Spotter

Given a document $d_i \in D$ and a concept graph C_{DAG}, concept spotter annotates all the concepts $c_j \in C_{DAG}$ that occur in the document d_i. Note that the labels associated with nodes in the graph C_{DAG} denote different concepts. The annotation process is quite simple. Each word w_k in the document d_i is converted into its lemma form and then it is matched against the lemma form of all the concepts that occur in the graph C_{DAG}. A single concept may be referred in multiple ways. As an example the concept *support vector machine* is also referred as *SVM*. A list of synonyms or a domain dependent taxonomy could be used in the matching process to annotate all the different variants of a concept.

3.2.3 Document Scorer

The document scorer module is responsible to give an overall score to a document. The score of a document captures its relevance based on a user defined objective. Using a domain specific knowledge base represented in a concept graph, we can devise several ranking metrics that capture different ranking objectives. In this section, we give examples of ranking metrics that use concept graphs to score each document. In particular, we present three different ranking metrics that can be used to achieve the example user objectives given in Section refsec:rankobj. Let the set d_c denote all the concepts that occur in a document d and are annotated by the concept spotter module. Let $pre(c_i)$ denote the set of all the immediate pre-requisite concepts for a given concept c_i. Similarly, the set $post(c_i)$ denote all the direct descendant concepts for the concept c_i. For the first objective given in Section 3.1, the following metric can be used:

$$Score(d) = \frac{\sum_{c \in d_c \cap pre(c)} \mathbf{f}(c)}{\sum_{c \in d_c \wedge c \notin pre(c)} \mathbf{f}(c)} \tag{1}$$

Where **f** is some arbitrary function that captures the importance of occurrences of the concept given as its argument. An example of function **f** could be number of times the concept c occur in a document.

Similarly, for the second and third ranking objectives, given in Section 3.1, the metrics given in Equation 2 and Equation 3 can be used respectively:

$$Score(d) = \frac{\sum_{c \in d_c \cap post(c)} \mathbf{f}(c)}{\sum_{c \in d_c \wedge c \notin post(c)} \mathbf{f}(c)} \tag{2}$$

$$Score(d) = \frac{\sum_{c \in d_c \wedge c \in (post(c) \cup pre(c))} \mathbf{f}(c)}{\sum_{c \in d_c \wedge c \notin (post(c) \cup pre(c))} \mathbf{f}(c)} \tag{3}$$

These functions are examples of ranking functions that can order documents containing pre-requisites before those having target concepts and vice-versa. Based on the application at hand the appropriate ranking function can be chosen and even customized. Here we have presented only a few examples of scoring metrics that can be used to achieve certain ranking objectives. Several other metrics can be devised using the concept graph to achieve other ranking objectives.

It is important to note that this framework provides easy and intuitive ways for integration of user personalization in ranking of search results. For a user, a profile can be created that captures his present level of knowledge in the concept graph. The search results in response to a concept given in query, can be ranked based on this profile such that the documents that describe concepts that are pre-requisite concepts to the query and are unknown to the user are scored

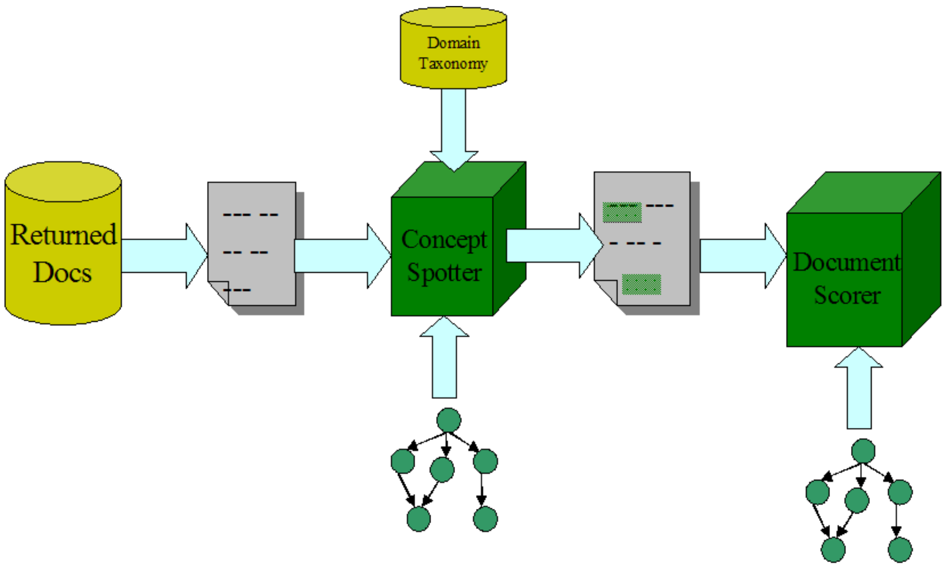


Figure 6: Overview of Ranking Framework

higher. We plan to integrate a user profile based ranking of search results in this framework in our future work.

4. VISUAL REPRESENTATION

In this section, we explain elements of the visual interface of our applications. The algorithms presented in the last section provide us with structure capturing dependence and relationships among concepts. However, such a structure stored in a flat text file (in adjacency matrix or list format) is virtually useless for learning goals. Without any visual aids, the user has to follow a cumbersome process of manually sifting through useful information. We highlight the need for a visual interface by an example.

Suppose a helpdesk agent wishes to help a customer install a specific plug-in in a word-processing application. The agent searches her knowledge base to figure out any dependencies to other components documented for installation. Depending on the customer's scenario, it could happen that the agent realizes she must explain installation of other plug-ins, even operating system components on the fly. Since the agent will not be an expert about these procedures, online troubleshooting becomes a tedious task. The whole process has to be repeated. This installation pre-requisites knowledge base needs to be constructed for effective use. A similar example for learning tree traversal algorithm in computer science was explained in section 1.

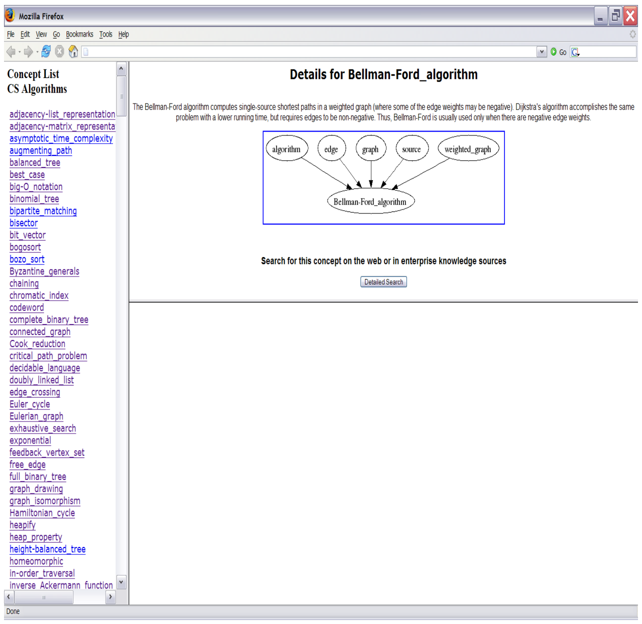
This manual recursive process is not too efficient for learning. Therefore, it is imperative that the derived structure (knowledge base) is presented to the user in an interactive environment. Next, we explain our web-based interface and its key components (along with the motivation behind them) in detail. We also discuss the use of these components in the overall visual learning process.

- **Concept Index:** The concept index is the list of all concepts in a domain of interest. There are two obvious ways of displaying this list; *hierarchical* and *alphabetical*. The hierarchical version is similar to directory structure usually employed in operating systems where the parent folder is displayed and sub-folders are recursively displayed by delving into the parent folder. In this case, the parent folder is a concept and sub-folders are pre-requisite concepts. This scheme is not very suitable here because an advanced concept can have many pre-requisite each of which can in turn have other pre-requisites. The view will be clumsy and the user will have horizontally scroll in our interface. Alphabetical listing circumvents this problem. Moreover, the alphabetical version also maps very nicely to indexes provided at the end of textbooks. This scheme is more traditional and allows an user to concentrate on learning concepts rather than on learning the tool. 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 concept tree is displayed on the right.

- **Concept Tree:** For the selected target concept, it's concept tree shows the target concept and it's pre-requisite nodes. In addition to these ancestor nodes (from a learning point of view), we also show one level of descendants from our concept index. These are nodes which need the target concept as their pre-requisite. This maintains context-sensitivity and the usefulness of showing ancestors and descendants is evident from our earlier example of tree traversal.

- **Concept Definitions:** The actual definitions of the selected concept is shown in the top of the right pane. The view is changed dynamically when the user selects a new concept either from the concept index or by clicking on a node of the concept tree. Clicking nodes in concept trees redirects to equivalent right panes as if the relevant link was clicked in the concept index.

- **Concept Search:** To make use of other information sources like web-repositories or enterprise data, we provide the capability of search. The user specifies the data source, query format and result set. These data



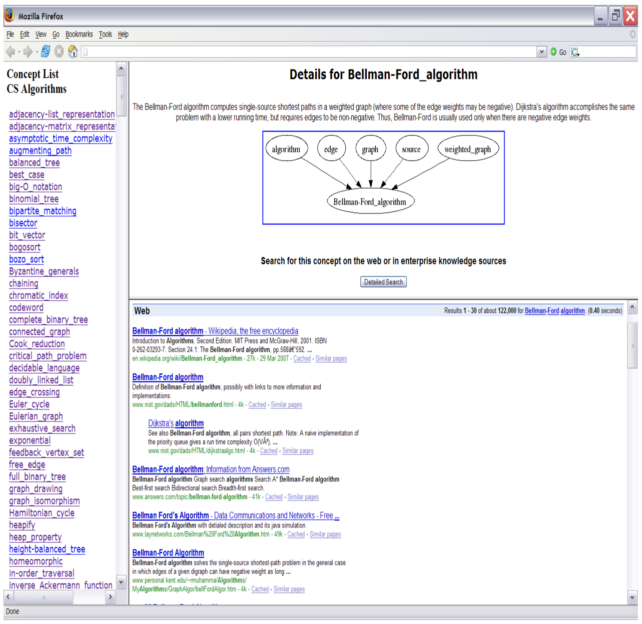
Concept List
CS Algorithms

adjacency-list_representation
adjacency-matrix_representa
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

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



Concept List
CS Algorithms

adjacency-list_representation
adjacency-matrix_representa
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

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

Web 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%20**Algorithm**.htm - 49k - Cached - Similar pages

Bellman-Ford Algorithm
Bellman-Ford algorithm solves the single-source shortest-path problem in the general case in which edges of a given digraph can have negative weight as long **...**
www.personal.kent.edu/~rmuhamma/**Algorithms**/ My**Algorithms**/GraphAlgor/bellFordAlgor.htm - 4k - Cached - Similar pages

Figure 7: Screenshots

sources are then queries for the selected concept and result displayed in lower part of the right pane. Please note that these data sources may provide very different types of information. For example, the enterprise data can be a patent database which returns all the patents where the concept occurred as a key word.

The user selects a concept from the concept index and the associated concept tree is presented. Next, the user identifies the pre-requisite to be learned and clicks on the corresponding node. Now a new concept tree is displayed. *The key point to note here is that due to context sensitivity of the concept tree the parent node (original target to be learned) is 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. Next, based on her interest, the user may search other data sources and refine or broaden her understanding.

5. EXPERIMENTS

In this section we present an evaluation of the ideas proposed in this work. As described, our aim was to explore building domain-specific knowledge bases that facilitate learning, either for it's sake or for fulfilling business needs. We modeled the knowledge base as a concept graph relating concepts of interest and the associated documents containing mentions or descriptions of these concepts. We also presented two sample applications based on this knowledge base (concept graph and associated document collection). In section 2 we presented effective interactive learning facilitated by a simple user interface. In section 3 we presented a family of ranking functions which can be employed to fulfill different application needs.

We wish to evaluate the goodness of our approach to building knowledge bases. Here, we present a user study that brings out some measure of the goodness and usefulness of our mined concept graph. We used the interactive interface described in section 4 for our evaluation. In this paper, we only wish to explore the idea of using concept graphs to build knowledge bases. We present no evaluation of the applications of learning and searching (ranking) themselves; these would be big independent exercises.

We evaluate the *'correctness'* of our algorithms and the *'usefulness'* of the presentation style of the concept graph. Our dataset processing algorithms explained in 2 draw up a directed graph modeling all pre-requisite relations in a large given list of concepts. It is impractical to evaluate such a large graph as a whole (for the computer science algorithms domain we had a sparse graph with about 1000 nodes). We resort to evaluating different aspects of sub-graphs related to individual concepts.

5.1 Datasets

In order to evaluate the correctness and usefulness of the concept ordering generated by our methodology, we picked two domains; *viz*; "algorithms" and "particle physics". For each domain, we extract a list of concepts from Wikipedia [2]. We obtain a total of 522 concepts for the "algorithms" domain and 457 concepts for the "particle physics" domain. We then use the "define" query operator provided by Google to build a concept definition database for each domain. For the "algorithms" domain we obtain 5191 definitions and for "particle physics" domain we obtain a total of 2650 definitions. We note that we can use any knowledge source (including enterprise knowledge bases) to collect definitions or mentions for concepts; we used web search for its simplicity and availability. Our techniques removed 663 edges resulting in cycles (18.6% of all edges) for the "algorithms" domain and 49 edges (3.8% of all) were removed in the "particle physics" domain.

5.2 Evaluation methodology

[2]http://en.wikipedia.org

As described in section 4, we have a navigable web-based interface that lists concepts in selected domains of interest. The user can explore any target concept of interest; it's corresponding *learning graph* is displayed. A target concept's learning graph contains pre-requisite concepts (ancestors) and higher-level concepts (descendents) for which the target concept is a pre-requisite. Only one-level ancestors and descendants are retained for simplicity of the learning graph as described in 4. In this section, we evaluate the 'correctness' and 'usefulness' of these graphs.

We conducted an elaborate user study to try and quantify these abstract notions. We requested 40 colleagues, all computer science graduates at least, to help in evaluating sample concepts along notions of correctness and usefulness. We use the standard information retrieval (IR) measures of *precision* and *recall*. We would like note here that it is not clear what a standard evaluation of such abstract notions would be. We explain our choices next; we believe precision matches the notion of correctness, and recall matches the notion of usefulness.

In IR, retrieval effectiveness of a query is evaluated in terms of precision and recall. An IR system attempts to retrieve a relevant document set for a given query from a large collection of documents. For a query, precision measures the fraction of relevant documents, to all documents retrieved by the system. Congruently, our user's learning need can be viewed as a query and the precision of a learning graph can be measured as $precision = \frac{\#correct\ directed\ edges}{\#edges}$. This can be evaluated with simple yes/no decisions on every directed edge by experts. Volunteers returned precision scores for a set of learning graphs shown in our web-based tool. We report average precision of the graphs, averaged over users, in 5.3.

The recall effectiveness of the system in response to a query is measured as the fraction of relevant documents retrieved, to all relevant documents present in the document collection. To be able to calculate recall, the total number of relevant documents in the document collection needs to be known before hand. In our case, the recall effectiveness of a target concept's learning graph is measured as $recall = \frac{\#correct\ directed\ edges}{\#expected\ edges}$. Estimating the denominator poses several challenges here, as also in IR systems, where it is only possible to define the full set of relevant documents for a specialized set of queries; that too from extensive domain knowledge. We asked volunteers, a different set than those measuring precision, to provide us with a comprehensive list of pre-requisite concepts for each of the target concepts from a sampled list. We collected answers, and cleaned-up and canonicalized concept-names by matching with our domain-specific list. This provided us the denominator for recall calculations, and the numerator was obtained from our learning graphs. We report average recall of the graphs, averaged over users, in 5.3. We also discuss a non-trivial study with novice users to evaluate the usability of our tool in 5.3.1.

5.3 Results and Discussion

We report the average precision and recall of the methods presented in this paper next. Table 1 reports the average precision of 15 learning graphs of target concepts evaluated by 20 expert users. We note that the precision values for a large variety of intermediate and advanced algorithms concepts are very high. Averaged over number of edges and users, we can conclude that most of the learning graphs for these target concepts were accurate.

Target concept	Precision
Complete binary tree	0.95
Manhattan distance	0.94
Post-order traversal	0.91
Minimum spanning tree	0.78
Venn diagram	0.69
Shell sort	0.97
Fibonacci heap	0.94
Recurrence relation	0.75
Undirected graph	0.47
NP-complete	0.81
Clique	0.60
Depth first search	0.90
Substring	0.56
Big-O notation	1.00
Hamiltonian Cycle	0.77

Table 1: Average precision - 20 users for 15 concepts

Table 2 reports the average recall for 15 target concepts and their learning graphs evaluated by 5 different expert users. These users only checked for completeness with respect to the starting list of domain-specific concepts. We see that the learning graphs selected report fairly high recall. However, we would like to state that the deviation across users is somewhat high because of the difficulty stated above in measuring recall.

Target concept	Recall
Complete binary tree	0.83
Manhattan distance	0.80
Post-order traversal	0.80
Minimum spanning tree	1.00
Venn diagram	0.87
Shell sort	0.90
Fibonacci heap	0.70
Recurrence relation	0.57
Undirected graph	1.00
NP-complete	0.85
Clique	1.00
Depth first search	0.90
Substring	0.90
Big-O notation	0.80
Hamiltonian Cycle	0.70

Table 2: Average recall - 5 users for 15 concepts

5.3.1 Measuring recall with novices

Ideally the usefulness of such a learning tool should be evaluated by novices rather than experts, as experts would be implicitly biased in feigning ignorance of pre-requisites for given target concepts. We conducted a study with two novices to try and get a clearer picture of the usability of our tool. We selected two non computer science graduates who had no training in algorithms.

We asked them to start at the Wikipedia page (or at similar web-based knowledge sources) for complete binary trees. As expected, they deemed it necessary to first learn about binary trees and in turn about trees. They had to get familiar with the abstract notion of data structures (with our help and intervention). While learning about trees they had to learn base concepts like root, internal nodes, children, and leaf nodes. They counted about 7 concepts they had to learn on the way to learning about complete binary trees. When shown our learning graph for complete binary trees, they identified 5 of the concepts they had to learn when starting without guidance on the web. We could hence assign a recall score of 5/7 for complete binary trees. We tried to repeat this experiment with other novices and concepts.

We'd like to admit that this study failed, retrospectively, for a variety of reasons. It was unfair to expect novices to actually follow links and try to learn concepts in a completely new advanced field of study. We should have conducted this study with users having some basic familiarity with software and computer science rather than selecting rank novices. We were constrained by dataset availability to be able to do such fair, though time-consuming, user studies. We hence resorted to the judgment of experts for measuring recall. On a similar note, as mentioned earlier, for the particle physics dataset and graphs we had available, we found no suitable experts for evaluation. Part of our future work involves precision and recall validation in different domains, for different starting lists of concepts, and exploiting richer sources of information.

6. RELATED WORK

In this section, we discuss some related efforts to the work presented in this paper. Chi et al. [3, 2] proposed ScentIndex which transforms the traditional book index to meet the particular requirement expressed by the user. The user inputs her request in the form of key words. The system searches for the most relevant index entries and displays it to her. Moreover, the system also highlights associated passages of the book. The final toolkit is very useful. However, no explicit modeling of dependence among concepts is done. It is unclear which of the most relevant index entries actually captures pre-requisite dependence. The focus of ScentIndex was to find conceptually related entries. Moreover, the ScentIndex system does not provide any ordering in which the concepts should be learned.

Chen et. al. [1] proposed an approach that uses journal and conference papers as resource to analyze the relationships among concepts in e-learning domains. They treat keywords listed in a research articles as essential concepts and construct a concept map based on the co-occurrence of keywords. Thus a concept map generated by their method, captures general relationships among concepts and does not focus on pre-requisite relationship.

The most pertinent and closely related work was presented by Khodor et al. [8]. The authors described Biology Concept Framework (BCF) to aid in organizing teaching material and lectures. BCF is organized in a multi level hierarchical fashion (very similar to the structure we derive). The authors define top-level concepts which are most general concepts and need to be learned before proceeding to advanced concepts. Moreover, these concepts are independent of each other in the sense that the concepts can be learned in any order. The next levels in the BCF are concepts and details which require the knowledge of one or more top level concepts. The key difference is that BCF was constructed manually by experts. No automatic algorithms were used to derive the structure, top level concepts and their relation with other advance concepts. Moreover, BCF is intended to be used by course instructors(experts), on the other hand our effort resulted in an interactive tool which non-experts can use and learn from

There are a few methods proposed in the literature that use user profiling techniques for knowledge personalization in e-learning context [12, 10, 4]. The profiling based algorithms can be useful in our approach and can be used as a pre-processing step. However in this paper, we focus on deriving and visualizing a concept graph for a given domain and leave personalization issues for our future work.

7. CONCLUSION AND FUTURE WORK

In this paper, we explored building of domain-specific knowledge bases for enabling certain learning oriented applications in service arms of organizations. We described the usefulness of these knowledge bases modeled as concept graphs capturing pre-requisite relationships between concepts. These concept graphs and their associated document collection serve as an effective knowledge base in certain applications.

We presented algorithms to automatically mine the concept graph in a given domain with little human intervention. The final concept graph can be tweaked by experts. We also presented a visual toolkit to facilitate the learning process where dependencies between concepts are presented to the user interactively. The user can recursively learn pre-requisite concepts to thoroughly understand target concepts of interest according to learning and information goals. The toolkit also allows the user to query on-line or enterprise databases and knowledge repositories. We also presented a family of ranking functions that rank documents while being aware of the concept graph. One application may require documents containing pre-requisites of a target concept to have a high rank while another application may need details about the target concept like definitions, details, and usage.

We evaluated correctness and usefulness of our concept graph mining approaches by conducting a moderately large user study. We used widely accepted evaluation metrics - precision and recall- to measure correctness and usefulness respectively. We did not evaluate the presented applications, leaving such to application builders, rather concentrated on building useful knowledge bases from concept graphs.

Some ongoing and future initiatives planned for extending our toolkit.

- Integration of profile based algorithms to facilitate user personalization forms a key part of future work.
- Incorporating each user's existing skill-set to set up a domain-specific baseline will further result in focused learning and reduce learning time. Currently, we are investigating the use of three levels *beginner, medium* and *expert* to quantify a user's existing skill set.
- We would like to build an integrated appliance incorporating ranking, learning, and visualizing in specific

industrial settings.

- Finally, we would like to conduct a more extensive user study for other domains and with novices to validate the full power of such knowledge bases.

8. REFERENCES

[1] N.-S. Chen, P. Kinshuk, C.-W. Wei, and H.-J. Chen. Mining e-learning domain concept map from academic articles. In *ICALT '06: Proceedings of the Sixth IEEE International Conference on Advanced Learning Technologies*, pages 694–698, Washington, DC, USA, 2006. IEEE Computer Society.

[2] E. H. Chi, L. Hong, J. Heiser, and S. K. Card. ebooks with indexes that reorganize conceptually. In *Proc. of CHI*, 2004.

[3] E. H. Chi, L. Hong, J. Heiser, and S. K. Card. Scentindex: Conceptually reorganizing subject indexes for reading. In *Proc. of IEEE Symposium on Visual Analytics Science and Technology*, 2006.

[4] P. Dolog, N. Henze, W. Nejdl, and M. Sintek. Personalization in distributed e-learning environments. In *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 170–179, New York, NY, USA, 2004. ACM Press.

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

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

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

[8] J. Khodor, D. G. Halme, and G. C. Walker. A hierarchical biological concept framework: A tool for course design. *Cell Biology Education*, 3:111–121, 2004.

[9] G. Miller, R. Beckwith, C. Fellbaum, and M. Gross, D. Wordnet: An on-line lexical database. *International Journal of Lexicography*, 3:235–244, 1990.

[10] E. Mor and J. Minguill&#243;n. E-learning personalization based on itineraries and long-term navigational behavior. In *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 264–265, New York, NY, USA, 2004. ACM Press.

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

[12] C. B. Teo and R. K. L. Gay. A knowledge-driven model to personalize e-learning. *J. Educ. Resour. Comput.*, 6(1):3, 2006.