Discovering Dependencies in Courseware Repositories

Nidhi Malik

Dept. of Comp.Sc. and engg.
Indian Institute of Technology Bombay

Mtech Defense
July 24,2008

- eLearning is a type of education in which medium of instructions is some computer technology.
- huge amount of data available on web in form of wikis, tutorials, blogs etc.
- different types of tools available from simply viewing the content to create lessons with the help of authoring tools.

Problem Definition

- Given a set of lecture files from some content repository, give the user the most relevant lecture module to study for his query.
- Suggest pre-requisites and follow-up modules also.
- We will also present the dependency graph for the whole course.

Outline of the Report

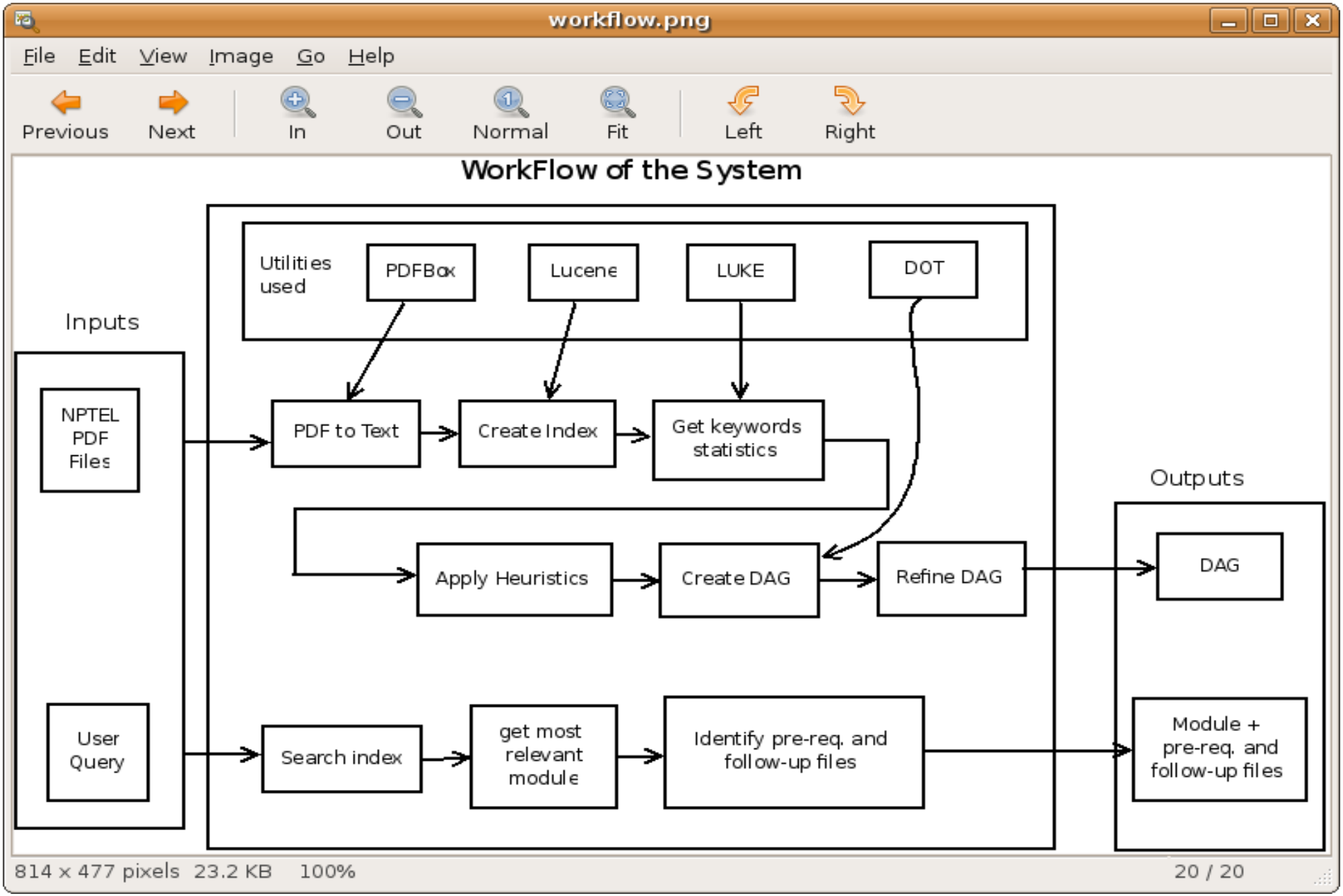
- Literature survey
- Overview of Solution approaches
- Implementation Details
- Evaluation of the System
- Feedback module
- Summary

Related Work

- Different types of LMS available
  - Atutor : available open source, being used internationally, translated into over fifteen languages.
  - OLAT : provide forums, quizzes, chats etc.
  - Other LMS available open source are Moodle, SCORM, eFront etc.

- Some universities/institutes have made thier content available free of cost. For example: NPTEL, MIT's OCW. Stanford University's eLearning initiative.
- Different search engines available based on factors such as model,type of information etc.
- Some of the open source search engines are Nutch, Egothor, Isearch etc.

Workflow of the System



Demo

- 6 courses from NPTEL repository
- Workflow as shown in previous slide
- Dependency DAG generated
- 4 different heuristics evaluated

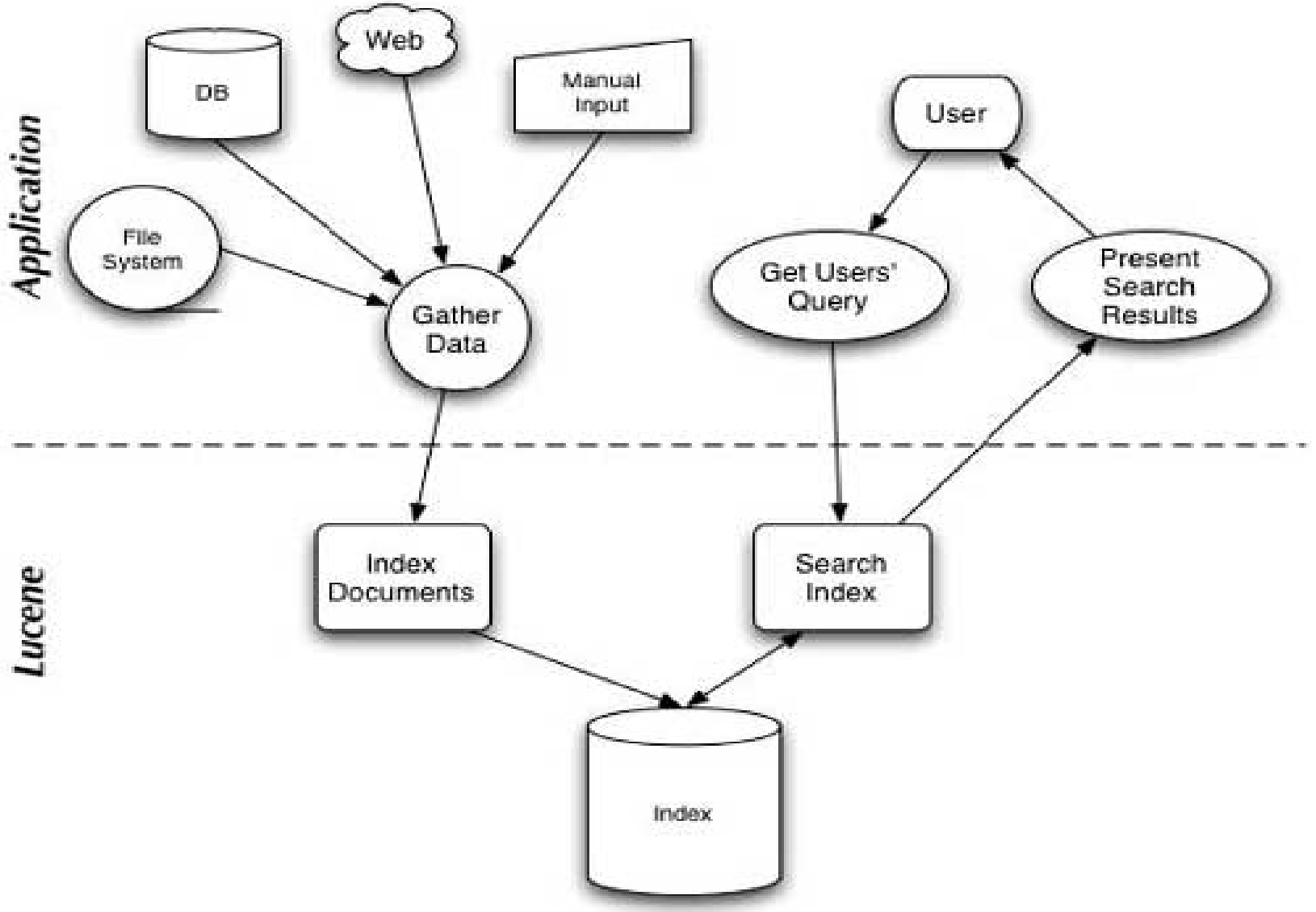
Parsing

- Lucene indexes only text data.
- Pdfbox – java library
- Nutch uses PDFbox for extracting pdf files to text.
- also allows to merge pdf documents, creating images etc.

Indexing

- Lucene is an free open source information retrieval library written in Java.
- Lucene is an API.
- Allows to print the index using LUKE.
- provide keyword statistics such as count of the keyword, frequency of occurrence, highlighting the term etc.
- basic classes of Lucene are indexwriter and indexsearcher.

Architecture of Lucene



NPTEL, content repository

- We have taken Computer Networks course from NPTEL with 40 pdf files in it.
- Indexed using Lucene.
- Got indexed printed using Luke.
- Get pre-requisites and follow-up files for each file.
- For every file, we have count of each keyword in each file.
- We have topkwords of each file.

Refining counts

- We need to refine the counts of keywords as these don't help to identify importance of keywords.
- Mean Threshhold - values less then mean are discarded.
- Percentage Threshold
- helps to get better counts and gives better results than mean threshhold.

- For a given file, with the help of refined counts we will get
  - the topkwords for this file
  - for each word in the topkwords, we will get the topkfiles.
    - Now, we need to order these files in order to get the pre-requisites and follow-up files.

Heuristic 1

- Take count of each Keyword in each file.
- For each file get topKkeywords
- For each keyword sort the file entries and get unique files
- Assign weight to each file based on sum of counts of all keywords appearing in it.
- Order the files according to their weights.
- For files whose index = 1 to i − 1; get the topK files according to weight.
- For files whose index > i;get the topK files according to weight.

Heuristic 2

- Take count of each Keyword in each file.
- For each file get topKkeywords.
- For each keyword get topKfiles.
- Sort the file entries and get unique files.
- For each file take position of the file for each keyword in topKfiles.
- Assign weight as w = K-p+1.
- For files whose index = 1 to i − 1; get the topK files.
- For files whose index > i;get the topK files.

Heuristic 3

- Take count of each Keyword in each file.(percentage threshold).
- For each file get topKkeywords
- For each keyword get topKfiles
- Sort the file entries and get unique files
- Assign weight to each file based on the average of sum of counts of all keywords appearing in it.
- Order the files according to their weights.
- For files whose index = 1 to i − 1; get the topK files according to weight.
- For files whose index > i;get the topK files according to weight.

Heuristic 4

- Take count of each Keyword in each file.
- For each file get topKkeywords
- For each keyword get topKfiles
- Sort the file entries and get unique files
- Multiply all keyword entries of the ith file to those of the others.
- Take sum of the resulting counts.
- For files whose index = 1 to i − 1; get the topK files according to weight.
- For files whose index > i;get the topK files according to weight.

- We have also kept records of the heuristics for the simplest counts(without any threshold) and the meanThreshold counts.

Generating DAG

- The graph is generated with the help of DOT.
- DOT is a graph description language, part of the Graphicviz package.

- After applying the different heuristics, we got pre-requisites and follow-up files for each file.
- We captured all the dependencies from our program in a .dot file.
- digraph graphname {
  - a -> b -> c;
  - b -> d;}

- Several attributes can be applied to control aspects like shape, color etc. in the graph.
- Currently, we are showing 3 pre-requisites and 3 follow-up files for each file.

Refining Graph

- Initially, we showed all dependencies captured from the program.
- The graph becomes messy and it is difficult to figure out the requisites for each file.

- For easy visualization, we refined the graph as follows:
    - There exists a link between X and Y iff X is a pre-requisite for Y and Y is a follow-up of X.

Evaluating the System

- To evaluate the performance of the system, we have compared results generated by our program with those of the program generated results.
- We created goodness metric for each course. We have created goodness metric separately for pre-requisites and follow-ups.

- P_i denotes the no. of pre-requisites generated by the expert.
- F_i denotes the no. of follow-ups generated by the expert.
- X_i denotes the no. of pre-requisites generated by the program.
- Y_i denotes the no. of follow-ups generated by the program.

$$Gp = \sum_{i=1}^{n} \frac{p_i}{x_i}$$

$$Gf = \sum_{i=1}^{n} \frac{F_i}{y_i}$$

$$G = \frac{Gp + Gf}{2}$$

Course	T0 - F0	T0 - F1	H1	H2	H3
Networks	76.87	77.49	78.95	78.54	79.16
AI	60.56	69.91	73.57	72.76	73.17
SE	87.1	90.67	88.69	83.92	85.11
Embedded	81.15	77.97	85.11	76.38	78.96
OS	80.15	81.74	86	77.77	78.57
SAD	92.85	92.85	90.85	91.85	92

Feedback

- Quiz Question bank
- separately stored questions for each topic
- objective in nature
- subject matter expert can view the statistics about the quiz such as how many learners appeared for it, %age of correct and incorrect answers.
- subject matter expert may change the curriculum depending on the feedback.

Summary

- Tried out all heuristics for 6 different courses.
- For some of the requisites there were no expert answers.
- After getting expert answers, we can make DAGs for any number of courses.

References

- Weimin Ge and Yuefeng Chao. Implementation of e-learning system for unu-iist.2005.
- Khan. Managing e-learning: Design, delivery, implementation and evaluation. 2005.
- Erik Hatcher and Otis Gospodnetic. Lucene in Action (In Action series). Manning Publications Co., Greenwich, CT, USA, 2004.
- Mit open courseware http://ocw.mit.edu.
- National programme on technology enhanced learning http://www.nptel.iitm.ac.in.

- http://en.wikipedia.org/wiki/List_of_search_engin
- http://en.wikipedia.org/wiki/OLAT.
- http://en.wikipedia.org/wiki/DOT_language.