CSA402

Lecture 17

The Architecture of a Generic AHS: Personal WebWatcher

References

Mladenic, D. (1996), Personal WebWatcher: design and implementation. Available on-line at
http://www.cs.cmu.edu/afs/cs/project/theo-4/text-learning/www/pww/papers/PWW/pwwTR.ps.Z

Mladenic, D. (1999), Machine learning used by Personal WebWatcher. Available on-line at
http://www.cs.cmu.edu/afs/cs/project/theo-4/text-learning/www/pww/papers/PWW/pwwACAI99.ps.gz

Additional information about Personal WebWatcher can be found at
http://www.cs.cmu.edu/afs/cs/project/theo-4/text-learning/www/pww/index.html

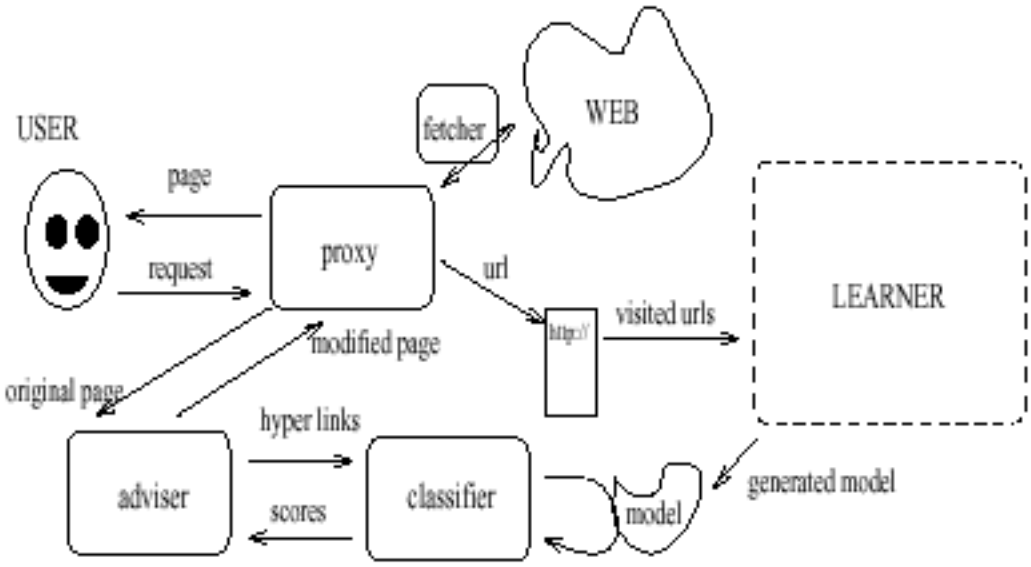
Overview

- We have seen the goals and objectives of Adaptive Hypertext Systems
- We have seen how research into ITS has influenced ITS-based AHS architecture
- We have seen how to represent user interests through User Modeling
- We have seen how Information Retrieval can be used to search for relevant documents based on a user query
- Today, we will see how Personal WebWatcher recommends documents to a user, based on an analysis of the documents that a user browses

Personal WebWatcher

- Personal WebWatcher was inspired by WebWatcher
- WebWatcher analyses user interactions in a Web site to offer an improved service to future users of that site
- WebWatcher is not a personal assistant
- Personal WebWatcher (PWW) *is* a personal search assistant
- PWW observes users of the WWW and suggests pages that they may be interested in
- PWW learns the individual interests of its users from the Web pages that the users visit
- The learned user model is then used to suggest new HTML pages to the user

Architecture of Personal WebWatcher



(From Mladenic, D. 1998 Personal WebWatcher: design and implementation)

- PWW consists of two main parts:
  - a Web proxy server
  - a learner
- The proxy saves URLs of visited documents to disk
- The learner uses them to generate a model of user interests

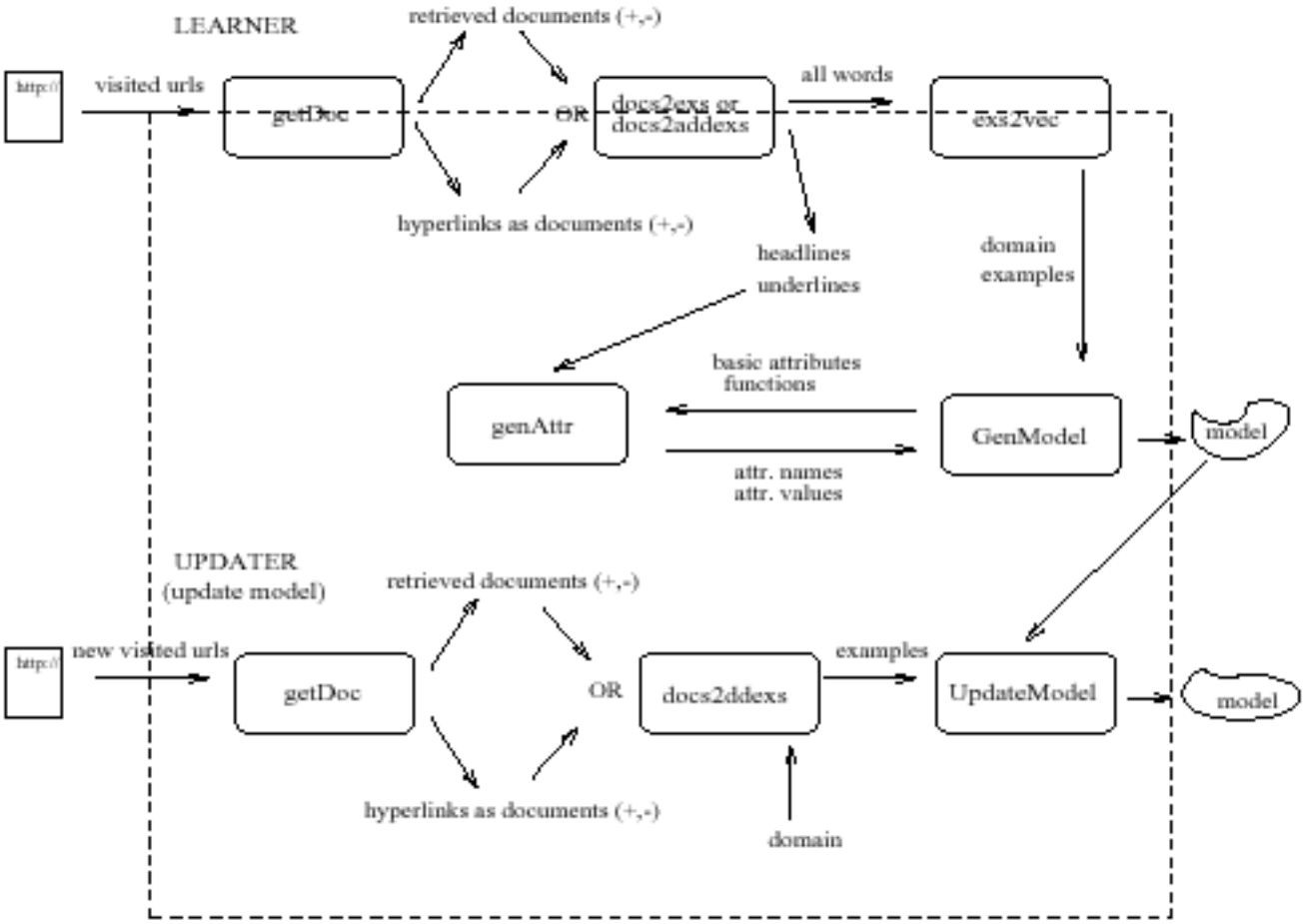
The Proxy Server

- Consists of three main parts:
  - the **proxy** itself (together with a **fetcher** that retrieves a document)
  - the **adviser**
  - the **classifier**
- When a user requests a Web page, the proxy fetches, adding advice if it is in HTML format
- Advice is added by the **adviser**, which extracts hyperlinks from the document and sends them to the **classifier**
- The **classifier** compares the linked-to documents to the user model, and, if there is a close enough match between a user interest and the target document, it will recommend the link to the user

The Learner

- The **learner** has two modes of operation
    - LEARNER: in which it learnes a new user model from scratch
    - UPDATER: in which it updates an existing user model
- In LEARNER mode the learner must decide how to define the domain
- In UPDATER mode, the domain is already defined, and so knows which terms to use to represent a document in vector space.
- In either mode, the learner processes the document that the user visited *and* the documents lying one link away from the visited document
- It processes visited documents as positive examples of the user's interest, and non-visited documents as negative examples
- The learner operates "off-line"

The Learner model



(From Mladenic, D. 1998 Personal WebWatcher: design and implementation)

The Model of User Interests

- The model is used to predict if some document is a positive example (relevant) or a negative example (non-relevant) to a user interest

- The model is applied to documents linked to from a user-requested document

- The hyperlinks in the requested document are marked up accordingly

- However, because it is time-consuming to process the linked-to documents in real-time, an estimation is made based on the hyperlink in the requested document...

$$User_{HL} \rightarrow \{pos, neg\}$$

Is the hyperlink a positive or negative example of the user interests?

- The hyperlink must be marked up in some way to attempt to capture information related to the destination document - PWW uses an extended representation of a hyperlink

- Apart from the hyperlink's anchor text, the extended representation includes underlined words, words in a window around the link source (e.g, the sentence), and words in headings above the hyperlink.

- Because the learner learns off-line, documents that the user has visited and unvisited hyperlinks from those documents can be processed to compare the predicted results to the actual user behaviour

$$User_{DOC} : Document \rightarrow \{pos, neg\}$$

Using the model generated from documents to predict "interestingness" of hyperlinks

- PWW can also predict document content based on a given hyperlink

$$Document_{HL} : Hyperlink \rightarrow document$$

Representing documents in PWW

- PWW has two overall objectives:
  - To learn about user interests from full-text representations of documents
  - To recommend links to user based on predictions of the interestingness of documents at the destination of hyperlinks
- In IR, documents are typically represented as *TFIDF*-vectors
- PWW operates largely on HTML documents, which have a structure
- Users may also want to add terms (which do not appear in the HTML document) to the document representation (e.g., WebWatcher)
- PWW uses TFIDF-vector together with success feedback (did the user follow a recommended link?)

- If the advice was poor, then use additional information to change advice to reflect what the user actually did.

- Term weight in the TFIDF vector is derived using *mutual information* between term frequency and class value (using Quinlan's Decision Trees)

- Mutual information assigns higher weights to terms which are better at distinguishing between interesting and uninteresting documents (based on observation)

The Learning Algorithm

- PWW compares a Naive (simple) Bayesian classifier on frequency vectors with the k-Nearest Neighbour approach to generate a model of user interests

- The Bayesian Classifier on boolean vectors assumes term independence

- It defines the probability of class *c* for some document *doc* that contains term *w* as proportional to

$$P(c)\Pi_w P(c|w)$$

- Yang's variation of k-Nearest Neighbour defines the relevance of class *c* given document *doc* as

$$rel(c|doc) = \sum_{i=1}^{k} similarity(doc,D_i)\, P(c|D_i)$$

- PWW experiments show that k-Nearest Neighbour gave slightly better results than the Naive Bayesian Classifier, but that the overall difference between the two learning algorithms was not significant (Mladenic, 1999)