A Match-Making System for Learners and Learning Objects

H. Boley[1], V.C. Bhavsar[2], D. Hirtle[2], A. Singh[2], Z. Sun[2], and L. Yang[2]

[1] *Institute for Information Technology e-Business, National Research Council of Canada*

[2] *Faculty of Computer Science, University of New Brunswick*
Fredericton, New Brunswick, Canada

Abstract

The AgentMatcher architecture is developed for Canadian Learning Object Metadata (CanLOM) of the eduSource e-Learning project. The LOMGen indexer, described in a companion paper, extracts CanLOM metadata from HTML learning objects (LOs). LOMGen-extracted terms can be selected from the query interface, permitting convenient entry of relevant tree parts and weights. Web-based prefiltering is then performed over the CanLOM metadata kept in the relational database of the KnowledgeAgora e-Learning repository. The prefiltering result is transformed to Weighted Object-Oriented (WOO) RuleML via an XSLT translator, and compared to the WOO RuleML-serialised tree obtained from the query interface. Finally, our similarity algorithm, described in an earlier paper, computes a percentage-ranked LO list, which is presented to the learner.

1. Introduction

We have developed the AgentMatcher system [2] for match-making between buyer and seller agents. This system is applied for searching procurable learning objects (LOs) in an e-Learning environment. Keywords and keyphrases are often used to describe LOs as well as learner queries in such environments. However, such a flat representation does not lend itself to hierarchical LO matching enabled by the Learning Object Metadata (LOM) standard and does not reflect user preferences about the relative importance of the parts of an LOM description. AgentMatcher combines both of these expressive extensions into tree-structured descriptions with arc weights for the queries, enhancing the precision of LO retrieval.

In this paper we describe the Java-based AgentMatcher match-making architecture as applied to the XML-based Canadian Learning Object Metadata (CanLOM) of the Canadian eduSource project [5]. The indexer of this architecture, the Learning Object Metadata Generator (LOMGen) described in companion work [3], extracts CanLOM metadata from HTML learning objects (LOs). LOMGen-extracted terms are offered to learners for selection from a query interface that permits convenient entry of relevant tree components and weights. Web-based prefiltering is then performed over the CanLOM metadata kept in the relational database of the KnowledgeAgora e-Learning repository of TeleEducation New Brunswick (TeleEd). The prefiltering result is transformed to

Weighted Object-Oriented (WOO) RuleML [6] via an XML-to-XML translator. This is then compared to the WOO RuleML-serialised tree obtained from the query interface, using our similarity algorithm [1], and a percentage-ranked LO list is obtained for presentation to the learner.

2. Overview

The AgentMatcher architecture can be applied to match-making [7] between buyer and seller agents in e-Business, e-Learning and other environments. The core of the AgentMatcher consists of similarity computation between metadata descriptions carried by buyer and seller agents. In the AgentMatcher instantiation for e-Learning, buyers are learners and sellers are learning object (LO) providers. We use the e-Learning standard CanLOM to describe learning objects (LOs). Thus, the match-making between buyer and seller agents corresponds to the matching of learner queries and CanLOM descriptions.

The architecture of the AgentMatcher as adapted to e-Learning is depicted in Fig. 1 showing the top-level retrieval and indexing components. There are three retrieval components, the User Interface, Similarity Engine and Translator, while the LOM Generator (LOMGen) performs indexing. Each of these four major components of the system is detailed in the ensuing sections.

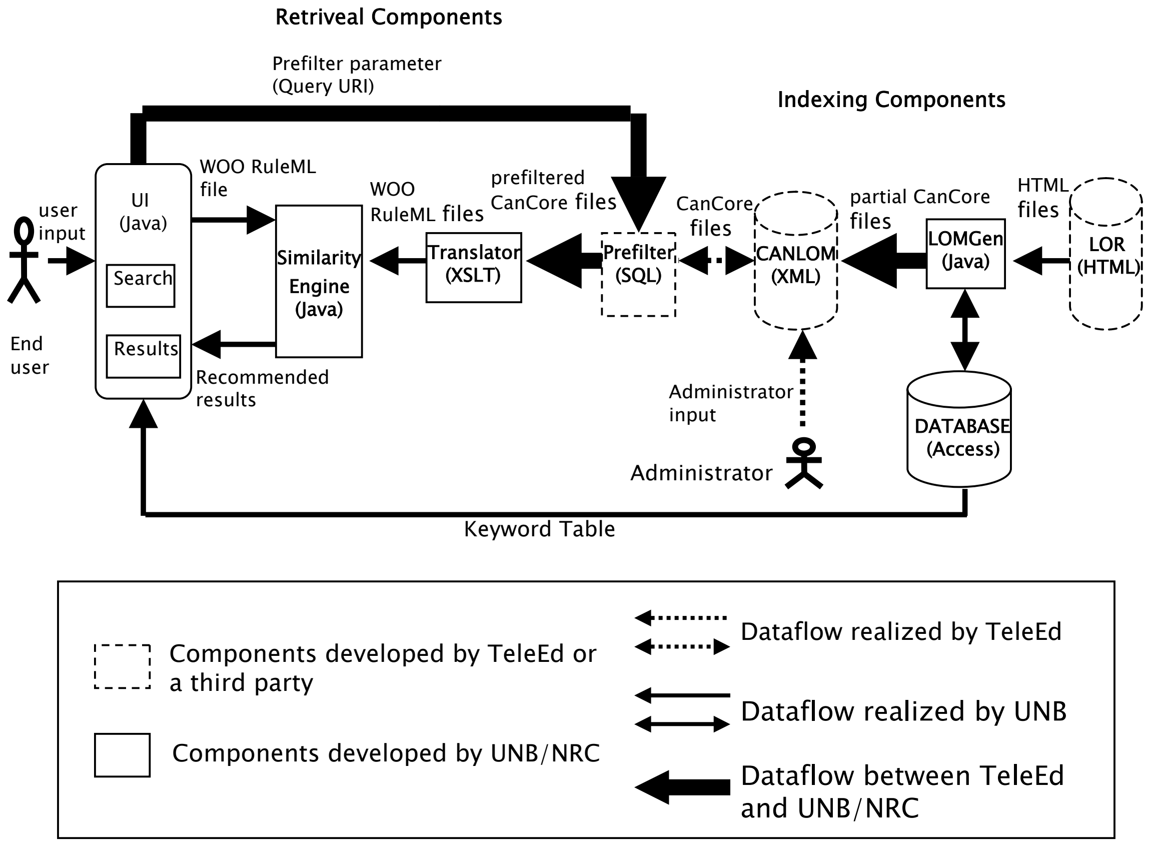
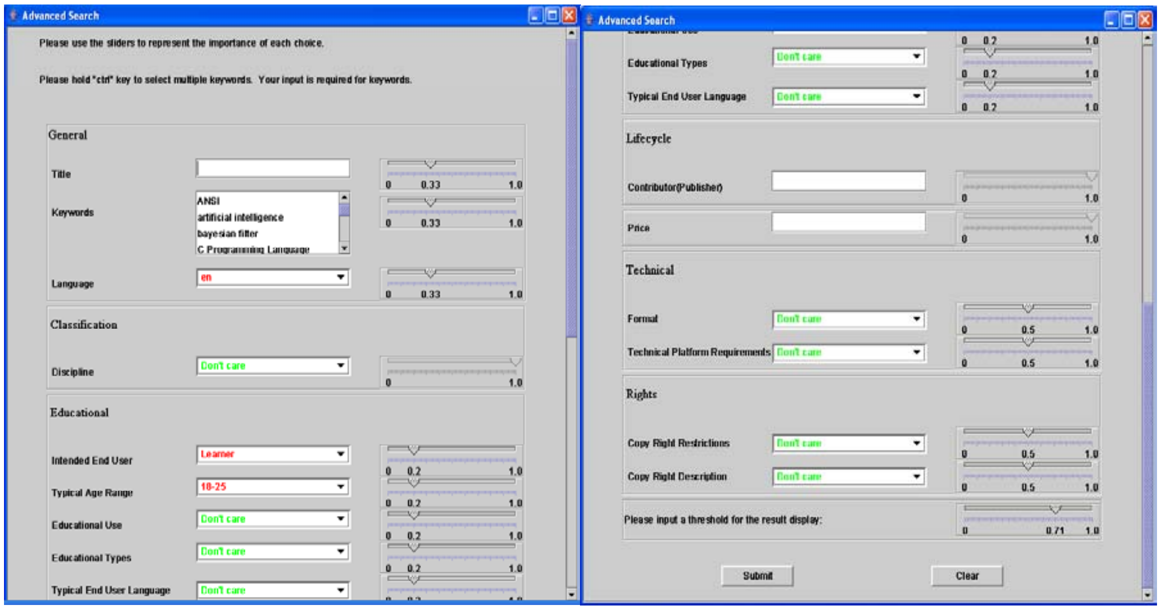


Figure 1. The AgentMatcher architecture.

3. User Interface

The user interface is employed for interaction between the end user and all other components. It permits a user to enter search parameters and retrieve ranked search results in a new browser window.



Please use the sliders to represent the importance of each choice.

Please hold "ctrl" key to select multiple keywords. Your input is required for keywords.

General		
Title		0 0.33 1.0
Keywords	ANSI, artificial intelligence, bayesian filter, C Programming Language	0 0.33 1.0
Language	en	0 0.33 1.0
Classification		
Discipline	Don't care	0 1.0
Educational		
Intended End User	Learner	0 0.2 1.0
Typical Age Range	18-25	0 0.2 1.0
Educational Use	Don't care	0 0.2 1.0
Educational Types	Don't care	0 0.2 1.0
Typical End User Language	Don't care	0 0.2 1.0
Lifecycle		
Contributor(Publisher)		0 1.0
Price		0 1.0
Technical		
Format	Don't care	0 0.5 1.0
Technical Platform Requirements	Don't care	0 0.5 1.0
Rights		
Copy Right Restrictions	Don't care	0 0.5 1.0
Copy Right Description	Don't care	0 0.5 1.0
Please input a threshold for the result display:		0 0.71 1.0

Figure 2. The user interface.

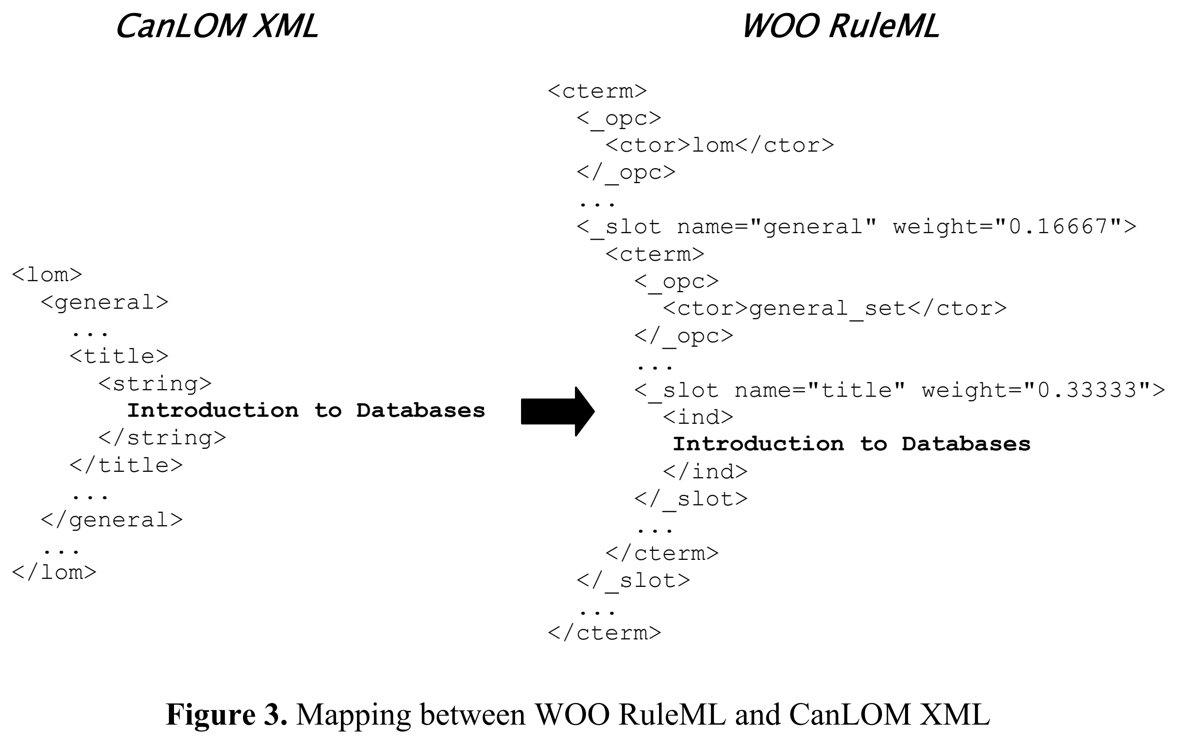
As shown in Fig. 2, the user interface is split into multiple boxes, each of which contains one or more search parameters chosen from the same category in the CanCore schema. Accompanying each search parameter is a slider, permitting the user to input not only the value for the parameter but also a corresponding weight. This weight indicates the importance of a parameter to the user relative to other parameters within the same category. All the weights within one box add up to 1.0. The user is also able to input a threshold for the search result recommendations, causing all the LOs with a similarity value above the threshold to be considered as the recommendations.

After the user submits the advanced search request, the internal functions will be invoked according to the dataflow in Fig. 1. First of all, a Weighted Object-Oriented RuleML (WOO RuleML) parameter file (hereafter referred to as `user.xml`) is generated by the user interface. WOO RuleML is the native format required by the Similarity Engine. Then selected search parameters are sent to the KnowledgeAgora database server for pre-filtering, using the conventional database query functionality to select relevant LOs by examining their Learning Object Metadata (LOM). The response from KnowledgeAgora is parsed into multiple XML files. These files are translated by the Translator into WOO RuleML files and passed to the Similarity Engine. At this point, `user.xml` is compared with each of the LOM files translated into WOO RuleML. The final result of the similarity computations is then displayed as a list of LOs ranked according to their similarity to the original search parameters entered by the user. Only those LOs with similarities above the threshold are recommended to the user.

4. Translator

The translator is responsible for translating the pre-filtered LOM files from the CanLOM repository into Weighted Object-Oriented RuleML, required by the Similarity Engine. It defaults LOM weights to equal values (up to rounding) on all tree levels, since this e-Learning application of AgentMatcher uses proper weights only for the query trees. The translator uses the Extensible Stylesheet Language Transformations (XSLT), a W3C Recommended language for transforming XML documents into other XML documents.

The (abbreviated) sample illustrated in Figure 3 demonstrates the mapping between the two formats. Additional information about this translation process is available in a separate report [4]. When translation is complete, the resulting WOO RuleML files are passed to the Similarity Engine for comparison to the WOO RuleML representation of the search parameters specified by the user.



CanLOM XML

```
<lom>
  <general>
    ...
    <title>
      <string>
        Introduction to Databases
      </string>
    </title>
    ...
  </general>
  ...
</lom>
```

WOO RuleML

```
<cterm>
  <_opc>
    <ctor>lom</ctor>
  </_opc>
  ...
  <_slot name="general" weight="0.16667">
    <cterm>
      <_opc>
        <ctor>general_set</ctor>
      </_opc>
      ...
      <_slot name="title" weight="0.33333">
        <ind>
         Introduction to Databases
        </ind>
      </_slot>
      ...
    </cterm>
  </_slot>
  ...
</cterm>
```

Figure 3. Mapping between WOO RuleML and CanLOM XML

5. Similarity Engine

The Similarity Engine is responsible for computing the similarity of the query and LOM files using our tree similarity algorithm [1]. It also displays the ranked list of search results in a browser window. The inputs of the Similarity Engine are the query file `user.xml` generated from the user interface and translated LOMs, as shown in Fig. 4. Similarity values fall into the real interval [0.0, 1.0]. The user will find on the top of the list the LOM that has the highest similarity value with his/her query.

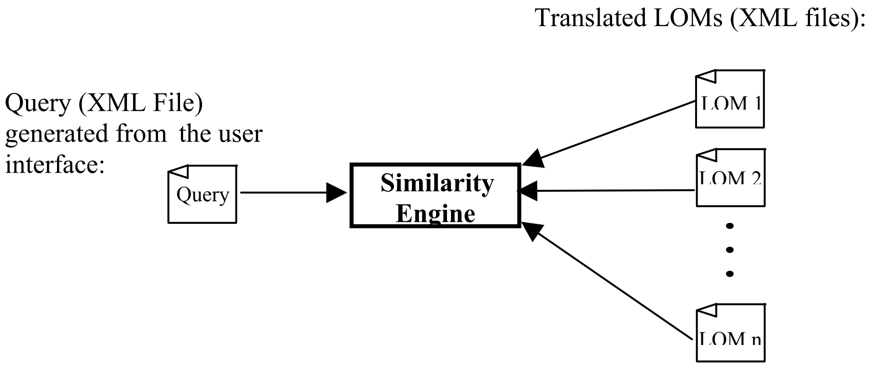
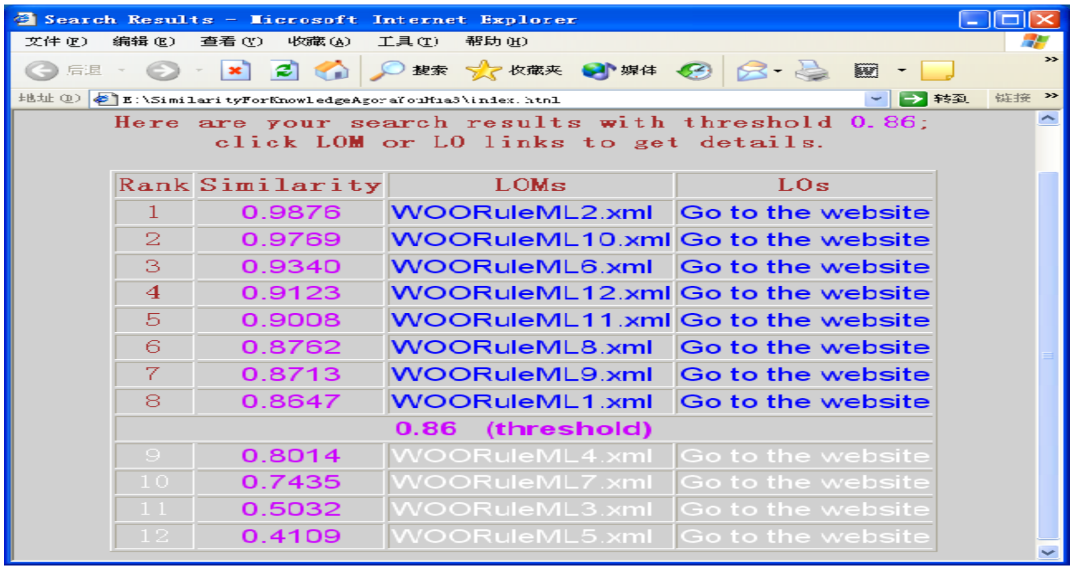

Translated LOMs (XML files):
Query (XML File) generated from the user interface:
Query
Similarity Engine
LOM 1
LOM 2
LOM n

Figure 4. Inputs of the Similarity Engine.

After computing the similarity between the query and LOMs, the Similarity Engine ranks all of the LOs in descending order of similarity, graphically separating those results whose similarity values fall below the threshold.


Search Results - Microsoft Internet Explorer
E:\SimilarityForKnowledgeAgora\oMia3\index.html

Here are your search results with threshold 0.86;
click LOM or LO links to get details.

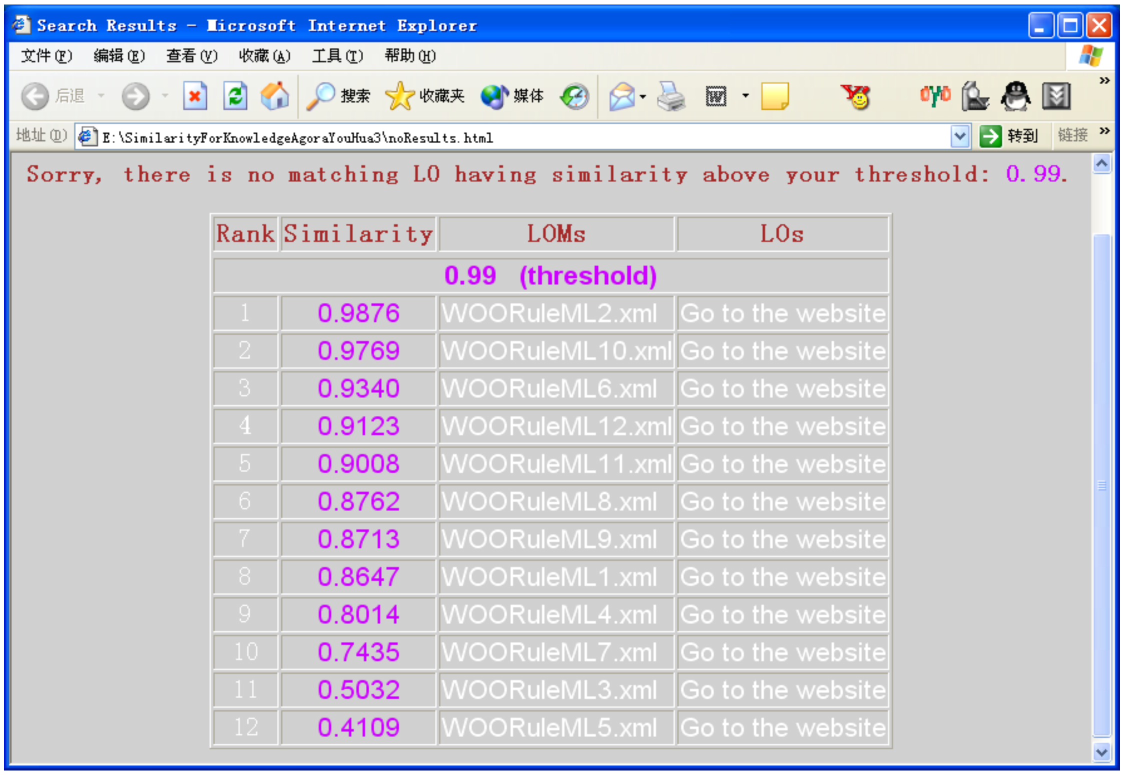
Rank	Similarity	LOMs	LOs
1	0.9876	WOORuleML2.xml	Go to the website
2	0.9769	WOORuleML10.xml	Go to the website
3	0.9340	WOORuleML6.xml	Go to the website
4	0.9123	WOORuleML12.xml	Go to the website
5	0.9008	WOORuleML11.xml	Go to the website
6	0.8762	WOORuleML8.xml	Go to the website
7	0.8713	WOORuleML9.xml	Go to the website
8	0.8647	WOORuleML1.xml	Go to the website
		0.86 (threshold)	
9	0.8014	WOORuleML4.xml	Go to the website
10	0.7435	WOORuleML7.xml	Go to the website
11	0.5032	WOORuleML3.xml	Go to the website
12	0.4109	WOORuleML5.xml	Go to the website

Figure 5. Snapshot of search results (low threshold).

Fig. 5 shows the ranking and HTML output for a relatively low threshold. There are four columns in the result table: Rank, Similarity, LOMs and LOs. The rank represents the relevance of the LOs to the user, where higher rank indicates higher similarity between the query and the LOM. The actual similarity values are displayed in the second column. All the LOMs and LOs shown in the final two columns are clickable; clicking the link of an LOM (e.g. `WOORuleML10.xml`) displays the metadata (in XML format) corresponding to the LO.

The “Go to the website” links in the final column point to organizations’ websites that display the contents of LOs.

Besides showing the search results above the threshold, we also show those that are below the threshold in case some users want to see more LOMs and LOs. Links for these results are displayed in white color.


Search Results - Microsoft Internet Explorer
文件(F) 编辑(E) 查看(V) 收藏(A) 工具(T) 帮助(H)
后退
搜索
收藏夹
媒体
地址(D)
转到
链接

E:\SimilarityForKnowledgeAgoraYouHua3\noResults.html

Sorry, there is no matching LO having similarity above your threshold: 0.99.

Rank	Similarity	LOMs	LOs
	0.99 (threshold)		
1	0.9876	WOORuleML2.xml	Go to the website
2	0.9769	WOORuleML10.xml	Go to the website
3	0.9340	WOORuleML6.xml	Go to the website
4	0.9123	WOORuleML12.xml	Go to the website
5	0.9008	WOORuleML11.xml	Go to the website
6	0.8762	WOORuleML8.xml	Go to the website
7	0.8713	WOORuleML9.xml	Go to the website
8	0.8647	WOORuleML1.xml	Go to the website
9	0.8014	WOORuleML4.xml	Go to the website
10	0.7435	WOORuleML7.xml	Go to the website
11	0.5032	WOORuleML3.xml	Go to the website
12	0.4109	WOORuleML5.xml	Go to the website

Figure 6. Snapshot of search results (high threshold).

Sometimes a user may input a too high similarity threshold that may result in a failed search. In this case, we do not ask users to go back to the user interface to input a lower threshold, but give users warning that their threshold is too high and still show all the search results that are below the threshold. Fig. 6 shows the search results in this situation. If users want to change other inputs (e.g., keywords), they have to go back to the interface to input again.

6. LOM Generator (LOMGen)

The process of manually entering metadata to describe an LO is a time-consuming process. The process requires the metadata administrator/author to be familiar with the LO content to a great extent. A semi-automated process which extracts information from the LO can alleviate the difficulties associated with this time-consuming process. The Learning Object Metadata Generator (LOMGen) aims at automating the metadata extraction process with minimal user intervention.

LOMGen works with LOs in Hypertext Markup Language (HTML) format. LOMGen uses the Free Online Dictionary of Computing (FOLDOC) to generate keywords and key phrases from an LO. It obtains the most frequent words and phrases from the contents of the LO. In order to get relevant results, frequently occurring stop words like is, are, the, in, etc. are ignored. This data combined with the Meta information found in the HTML file is passed on to a database module, which interfaces with the dictionary. Additional key phrases that may not be present in the LO but are relevant, are generated with the help of the dictionary. The metadata administrator is then presented with a Graphical User Interface (GUI) for key phrase selection, synonym and term addition (see Fig. 8). The updates made by the administrator are stored in a database. Subsequently, while parsing another LO, the newly added terms are considered to provide better choices to the administrator.

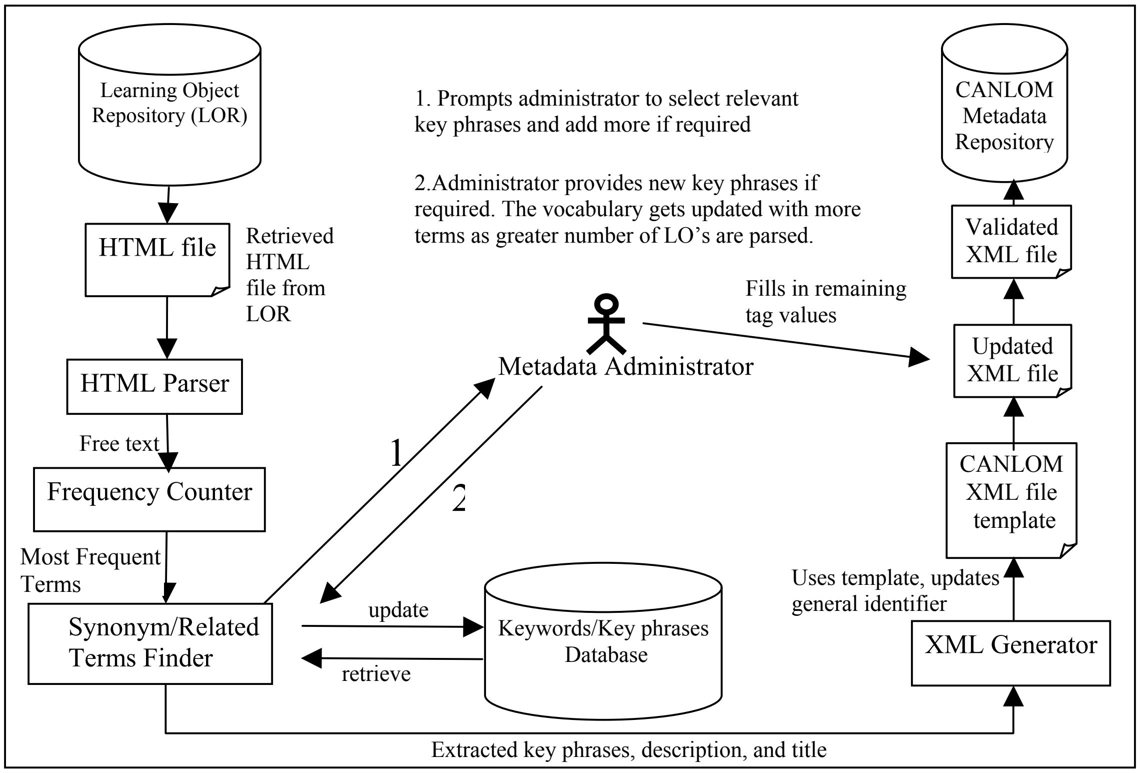

Learning Object Repository (LOR)
HTML file
Retrieved HTML file from LOR
HTML Parser
Free text
Frequency Counter
Most Frequent Terms
Synonym/Related Terms Finder
update
retrieve
Keywords/Key phrases Database
1. Prompts administrator to select relevant key phrases and add more if required
2.Administrator provides new key phrases if required. The vocabulary gets updated with more terms as greater number of LO's are parsed.
Metadata Administrator
1
2
Fills in remaining tag values
CANLOM Metadata Repository
Validated XML file
Updated XML file
CANLOM XML file template
Uses template, updates general identifier
XML Generator
Extracted key phrases, description, and title

Figure 7. LOMGen architecture.

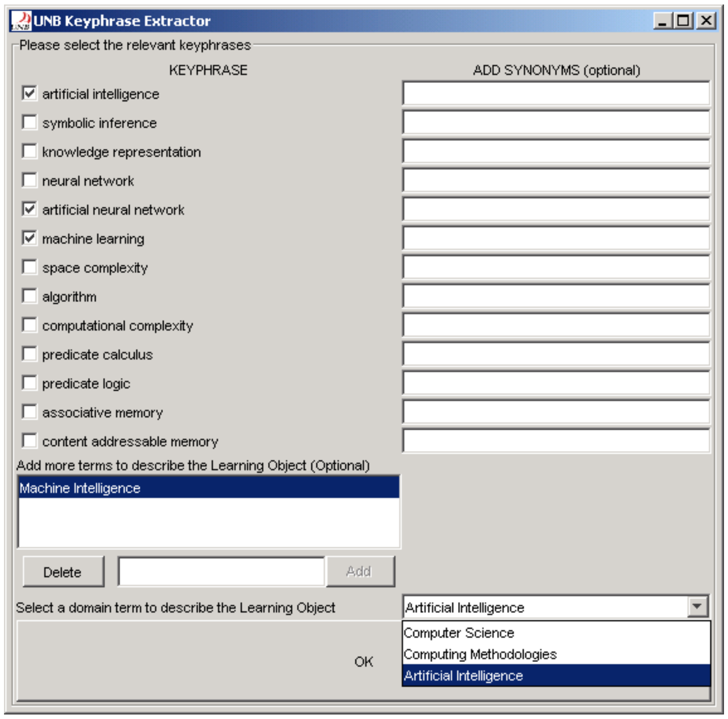
As shown in Fig. 7, the LOMGen architecture consists of an *HTML file reader module* which can read an LO file from a Uniform Resource Identifier (URI) or the disk, an *HTML parser*, a *word frequency counter*, a *database interface module*, and an *XML file writer* which updates the metadata repository with a newly generated LOM file.

A snapshot of the GUI presented to the metadata administrator is shown below in Fig. 7.

The GUI presents a list of keywords and keyphrases which were extracted or derived from the LO. The checkboxes present under the title "KEYPHRASE" allow the metadata administrator to select the most important keywords or keyphrases. The textboxes under "ADD SYNONYMS" allow the administrator to add a term which acts as a synonym for the corresponding keyphrase on the left. These synonyms, if added by the administrator, are also added to the database. The choices made by the administrator populate the domain term dropdown listbox. A domain term gives a hierarchy for classifying the LO.

If an LO lacks sufficient information in the text and HTML metatags, the quality of the keywords or keyphrases extracted by LOMGen may not be satisfactory. In such a scenario, the GUI enables the administrator to add more terms explicitly to describe the LO.

Finally, clicking the "OK" button generates an LOM file with the metadata and posts it to the LO repository.



UNB Keyphrase Extractor

Please select the relevant keyphrases

KEYPHRASE	ADD SYNONYMS (optional)
✓ artificial intelligence	
symbolic inference	
knowledge representation	
neural network	
✓ artificial neural network	
✓ machine learning	
space complexity	
algorithm	
computational complexity	
predicate calculus	
predicate logic	
associative memory	
content addressable memory	

Add more terms to describe the Learning Object (Optional)

Machine Intelligence

Select a domain term to describe the Learning Object: Artificial Intelligence

Computer Science
Computing Methodologies
Artificial Intelligence

Figure 8. GUI for key phrase selection.

The LOMGen component can be used as a training module to a text summarizer which uses machine learning techniques, with the intention of eliminating administrator inputs over a period of time.

7. Conclusion

The AgentMatcher match-making system is applied to e-Learning scenarios where learners are in search of procurable LOs. The resulting Java-based architecture takes advantage of the added expressiveness possible with tree-based matching and user-assigned weights. CanLOM metadata is extracted from HTML LOs by the LOMGen indexer, greatly speeding the task of metadata generation. The metadata is first prefiltered via a query URI, and then transformed to Weighted Object-Oriented RuleML via an XSLT translator. The results are then compared to another tree representation of the learner query as generated by the user interface. Finally, a list of learning objects is presented to the learner in descending order of similarity, computed by the weighted tree similarity algorithm.

This application of AgentMatcher, restricted to the computer science domain for the purposes of this project, demonstrates enhanced precision relative to standard keyword-based searches. AgentMatcher is easily adapted to match-making in other domains; in fact, an application currently in development involves technology transfer.

Acknowledgements

We thank the CANARIE eduSource Project and NSERC as well as the New Brunswick Department of Education for their support.

References

[1] Bhavsar, V.C., H. Boley, L. Yang, “A Weighted-Tree Similarity Algorithm for Multi-Agent Systems in E-Business Environments”, *In* Proceedings of 2003 Workshop on Business Agents and the Semantic Web, Halifax, June 14, 2003, National Research Council of Canada, Institute for Information Technology, Fredericton, pp. 53-72, 2003. (Revised version to appear in *Computational Intelligence*, November 2004.)

[2] Sarno, R., L. Yang, V.C. Bhavsar and H. Boley, “The AgentMatcher architecture applied to power grid transactions”, *In* Proceedings of the First International Workshop on Knowledge Grid and Grid Intelligence, Halifax, 2003, pp. 92-99.

[3] Singh, A., H. Boley and V.C. Bhavsar, “A Learning Object Metadata Generator Applied to Computer Science Terminology,” Presented at Learning Objects Summit, Fredericton, March 29-30, 2004.

[4] Hirtle, D. and Z. Sun, “CanCore ⇔ WOO RuleML”, Internal Report, Faculty of Computer Science, University of New Brunswick, December 2003.

[5] eduSource Canada: Canadian Network of Learning Object Repositories, http://www.edusource.ca/english/home_eng.html, June 13, 2004.

[6] Boley, H. 2003. Object-Oriented RuleML: User-level roles, URI-grounded clauses and order-sorted terms. Springer-Verlag, Heidelberg, LNCS-2876, pp. 1-16.

[7] Sycara, K., M. Paolucci, M. van Velsen, and J. A. Giampapa. 2001. The RETSINA MAS infrastructure. Robotics Institute, Carnegie Mellon University, CMU-RI-TR-01-05.