A Hybrid Recommender System Guided by Semantic User Profiles for Search in the E-learning Domain

Leyla Zhuhadar and Olfa Nasraoui
Knowledge Discovery and Web Mining Lab
Department of Computer Engineering and Computer Science
University of Louisville, Louisville, KY 40292, USA
Emails: leyla.zhuhadar@wku.edu, olfa.nasraoui@louisville.edu

Abstract—**Various concepts, methods, and technical architectures of recommender systems have been integrated into E-commerce storefronts, such as Amazon.com, Netflix, etc. Thereby, recently, Web users have become more familiar with the notion of recommendations. Nevertheless, little work has been done to integrate recommender systems into scientific information retrieval repositories, such as libraries, content management systems, online learning platforms, etc. This paper presents an implementation of a hybrid recommender system to personal the user's experience on a real online learning repository and vertical search engine named *HyperManyMedia*. This repository contains educational content of courses, lectures, multimedia resources, etc. The main objective of this paper is to illustrate the methods, concepts, and architecture that we used to integrate a hybrid recommender system into the *HyperManyMedia* repository. This recommender system is driven by two types of recommendations: *content-based* (*domain ontology model*) *and rule-based* (learner's interest-based and cluster-based). Finally, combining *the content-based* and the *rule-based* models provides the user with hybrid recommendations that influence the ranking of the retrieved documents with different weights. Our experiments were carried out on the *HyperManyMedia* semantic search engine at Western Kentucky University. We used *Top-n-Recall* and *Top-n-Precision* to measure the effectiveness of re-ranking based on the learner's semantic profile. Overall, the results demonstrate the effectiveness of the re-ranking based on personalization.**

Index Terms— **recommender system, search engine, clustering, personalization, semantic profile**

I. Introduction

The work presented in this paper describes a hybrid recommendation based retrieval model that can filter information based on user needs. We believe that the methodology for designing an efficient recommender system, regardless of the approach used, i.e., content-based, collaborative, or hybrid, is to incorporate the following essential elements: contextual information, user interaction with the system, flexibility of receiving recommendations in a less intrusive manner, detecting the user's change of interest and responding accordingly, supporting user feedback, and finally the simplicity of the user interface. We noticed, by tracking user behavior in our applied personalized vertical search engine, *HyperManyMedia*, that using general recommendation methods was not sufficient to make users interested in using the recommendations provided by the system. However, if the recommender system was tailored to the user's specific needs via personalization, the user got more interested and engaged into the recommendation process. Our finding resulted in generalizing the personalization aspect. We considered personalization as the main building block of the recommender system architecture. This conclusion is noticeable in most of the recommender systems that succeeded. Their success was a result not of the complexity of the theoretical methodology that has been used to design the system, but rather of the usability and the simplicity of the recommender system interface which guides the user without interrupting his/her activities. In this paper, we present an implementation of a hybrid recommender system on a search engine frontend to a real online learning repository named *HyperManyMedia*. This repository contains educational content of courses, lectures, multimedia resources, etc. The main objective of this paper is to illustrate the methods, concepts, and architecture that we used to integrate the recommender system into the *HyperManyMedia* repository. This recommender system is driven by two types of recommendations: *content-based* (*domain ontology model*) *and rule-based* (learner's interest-based and cluster-based). The domain ontology model which is used to represent the learning materials, is composed of a hierarchy of concepts and subconcepts that represent colleges, courses, and lectures; whereas, the learner's ontology model represents a subset of the domain ontology (an ontology that contains only a personalized, pruned subset from the whole domain which consists only of the college/courses/lectures that the learner is interested in). Finally, combining *the content-based* and *rule-based* recommendations provides the user with hybrid recommendations that influence the ranking of the retrieved documents via different weights. However, before describing the design of our system, we first

doi:10.4304/jetwi.2.4.272-281

present a comprehensive background of the origin of recommender systems and other related work in Section II. Then, we present various methodologies that we used in Section III, followed with a detailed description of our implementation and the evaluation results in Section IV. Finally, we draw our conclusions.

II. Previous Work

The scope of literature review in this paper concerns recommender systems in academic repositories. More specifically, we are interested in answering the following question: **What is the current state-of-the-Art and the next generation of recommender systems in academic repositories and do scientific portals, digital libraries, and e-books repositories consider the value of embedding recommender systems into their system?**

To answer this question, we reviewed the most popular scientific digital libraries. In addition, we investigated some of the promising Web 2.0 digital libraries that use recommender systems.

We believe that a few of the main services that can benefit from the usage of recommender systems are digital libraries. In particular, when we compare the usability of search engines with digital libraries, we notice that the design of search engines has changed dramatically over the last decade. Web users can easily search for resources using search engines. This flexibility is provided by the simplicity of the search engines' user interface. However, digital libraries did not adapt to those changes. The complexity of using a combined methodology of Boolean operators with Metadata fields to retrieve resources from databases is considered to be a tedious process, especially for the new generation of Web users who are not used to spending a long time to search for resources. For example, many Web users now prefer using Google Scholar to search for journal articles, research papers, and e-books, regardless of its limitation to provide the user with a complete access to the resource (unless the user already set his/her digital libraries' access inside the advanced feature in Google Scholar), than the digital libraries, such as ACM, IEEE Xplore or CiteSeer. Thus it appears that the simplicity of the Google Scholar interface surpasses the accuracy that major digital libraries provide. However, ACM, IEEE Xplore and CiteSeer incorporated some techniques that could be considered as a form of recommendations (with little success). For example, ACM Portal provides two types of recommendations: (1) a content-based research tool known as "find similar articles". The mechanism used to find similar papers involves three techniques: cluster analysis, dictionary and thesauri. The retrieved documents are ranked based on date, publisher, or relevance, but there is no reference to the type of measure used in the ACM Portal, as cited in [14], (2) behavior-based recommendations presented as "Peer-to-Peer readers of this article also read". According to [14], this recommendation is built using simple frequency counts, and therefore fails to provide accurate recommendations.

According to [2], IEEE Xplore announced the implementation of content-based recommendations on their portal. Nevertheless, to date, no such recommender system is embedded into the IEEE Xplore libraries. However, CiteSeer1 showed a promising venue for the usage of recommender systems. The first prototype provided the users with three different types of recommendations: (1) link structure-based recommendation: those recommendations are based on link citations and they can be distinguished into four types of recommendations (recommend documents that are cited inside the searched document, recommend documents that cite the document, the Co-citation and the active bibliography), (2) content-based recommendations using (TF-IDF) similarity metrics and (3) explicit recommendations, where the user can rank the retrieved documents on a scale of 1 to 5. In addition, the user can write a review or a comment about the paper. However, the progress of this portal apparently stopped since 2006. The success of Google Scholar is evident even though it provides limited recommendations, e.g., finding similar documents based on content and the ranking of those documents may be inherited from Google's page ranking algorithm. Another limitation of Google Scholar is that it does not retrieve documents that are cited inside a specific document, but rather only the documents that cite this specific document. As we noticed, a variety of recommender systems portals have been implemented in the domain of digital libraries and scientific repositories, some of which succeeded while others failed to survive. In the following paragraph, we discuss two significant implementations of scientific recommender systems. The first is the Melvyl recommender system, which has been implemented by the California digital library2. This system uses a simple technique to provide recommendations to users. First, it generates a graph of all the purchased documents in the library, then each document is considered as a weighted node (with the weight representing the number of purchases). Therefore, the recommendation for a given document is based on the neighboring nodes (documents) which are sorted according to their edge weights. The second is TechLens3, which is specialized for the domain of scientific papers, it uses hybrid recommendations combining a collaborative filtering and a content-based approach. The system uses graph theory where each research paper is considered as a node and the citations inside each paper are considered as recommended nodes. Also, the system uses a more complex collaborative filtering (CF) technique that considers each cited paper as an input, therefore also considering all citation papers as recommendations. This technique is referred to as Dense CF. Finally, the system applies a content-based recommendation technique (TF-IDF) on the list of all

[1] http://citeseer.ist.psu.edu
[2] http://www.dlib.org/Architext/AT-dlib2query.html
[3] http://techlens.cs.umn.edu/tl3

recommended papers. Thus, the most similar papers are recommended to the user. The system provides two options: (1) Pure content-based CF (the similarity measure is only based on two entities, the title of the paper and the abstract, and (2) Content-based Separated-CF, where the whole text in the papers is considered as the final recommendations provided to the user would be a list of sorted recommendations that combine multiple factors based on the type that the user chose. Recently, with the increased popularity of social tagging systems, portals such as CiteULike4 and BibSonomy5, are considered promising projects that use social bookmarking to derive recommendations. [1], [5], [6], [4] used a different approach to recommend documents based on the user profiles, In this case by learning from implicit feedback or past click history. Other ways to form a user model include using data mining, such as by mining association rules [11], or by partitioning a set of user sessions into clusters or groups of similar sessions. The latter groups are called session clusters [12], [10], or user profiles [12], [10]. More recently, [13] presented a Semantic Web usage mining methodology for mining evolving user profiles on dynamic Websites by clustering the user sessions in each period and relating the user profiles of one period with those discovered in previous periods to detect profile evolution, and also to understand what type of profile evolutions have occurred. This latter branch of using data mining techniques to discover user models from Web usage data is referred to as Web Usage Mining. A previous work that used Web mining for developing smart E-learning systems [16] integrated Web usage mining, where patterns were automatically discovered from users' actions, and then fed into a recommender system that could assist learners in their online learning activities by suggesting actions or resources to a user. Another type of data mining in E-learning was performed on documents rather than on the students' actions. This type of data mining is more akin to text mining (i.e., knowledge discovery from text data) than Web usage mining [3]. This approach helps alleviate some of the problems in E-learning that are due to the volume of data that can be overwhelming for a learner. It works by organizing the articles and documents based on the topics and also providing summaries for documents. [7] combines Web usage mining and text-based indexing and search in the content to provide hybrid recommendations. [8] uses a learning algorithm to select sequential articles based on context and user-click feedback to recommend news articles to users. Our approach shares some similarity with the above techniques. It is a Hybrid recommender system which combines Content-based recommendations with two types of Rule-based recommendations. In Section III, we explain our methodology, followed by the implementation section. Finally, we present our evaluations and we conclude with our key findings.

III. Methodology

The methodology of designing our hybrid recommender system is divided into two parts: (1) The first part centers around designing the domain ontology: First, we relied on a fine grained taxonomy that encapsulates all the domain of Education in general, and E-learning in specific, by borrowing an already made taxonomy from WordNet. This attempt ended with a great disappointment since the terminology used in WordNet is far wider and different than what the *HyperManyMedia* domain contains. As a result, two major problems occur, the overloading of the fine grained taxonomy during the searching process and the ambiguity. Therefore, a decision was made to create a hand-made ontology using a coarse-grained taxonomy. In Section IV, we describe in detail the design of the domain ontology. (2) The second part centers around designing the learner's ontology: Each learner has his or her own ontology based on his/her preferences. The learner's ontology is extracted from the domain ontology and presented as a pruned subset ontology. In Section IV, we describe in detail the design of the learner's ontology. In the following sections, we describe the methodology used to provide the learner with hybrid recommendations: (1) Ontology Content-based, (2) Cluster-based, and (3) Interest-based.

A. Building the HyperManyMedia Domain Ontology

Recently, a variety of knowledge-based framework applications became available that support modeling ontologies. The best known applications are Protégé6 and Altova7. We used Protégé as a framework application. Figure 1 shows the design of the *HyperManyMedia* ontology in Protégé. Since our approach is based on a search engine recommender system, the content of each lecture is considered as a document and the recommendation of pages is related to the degree of matching between a learner's query and the reverse-indexing of the lecture (Webpage). The *HyperManyMedia* search engine uses the Vector Space Model (VSM) and the score of a query q for a document d is computed based on the cosine similarity between the document and the query vector. The implementation can be described as follows: (1) Preliminary crawling and indexing (offline): crawling and indexing the E-learning platform that contributes to the content of the recommendation; (2) We start by representing each of the N documents as a term vector $d = < w_1, w_2, ...w_n >$, where w_i is the term weight for term (i), combining the term frequency, tf_i, and the Term's Inverse Document Frequency $IDF_i = \log\frac{N}{n_i}$ if this term occurs in n_i documents, as $w_i = tf_i * \log\frac{N}{n_i}$, and (3) Building the E-learning Domain Ontology: Let R represent the root of

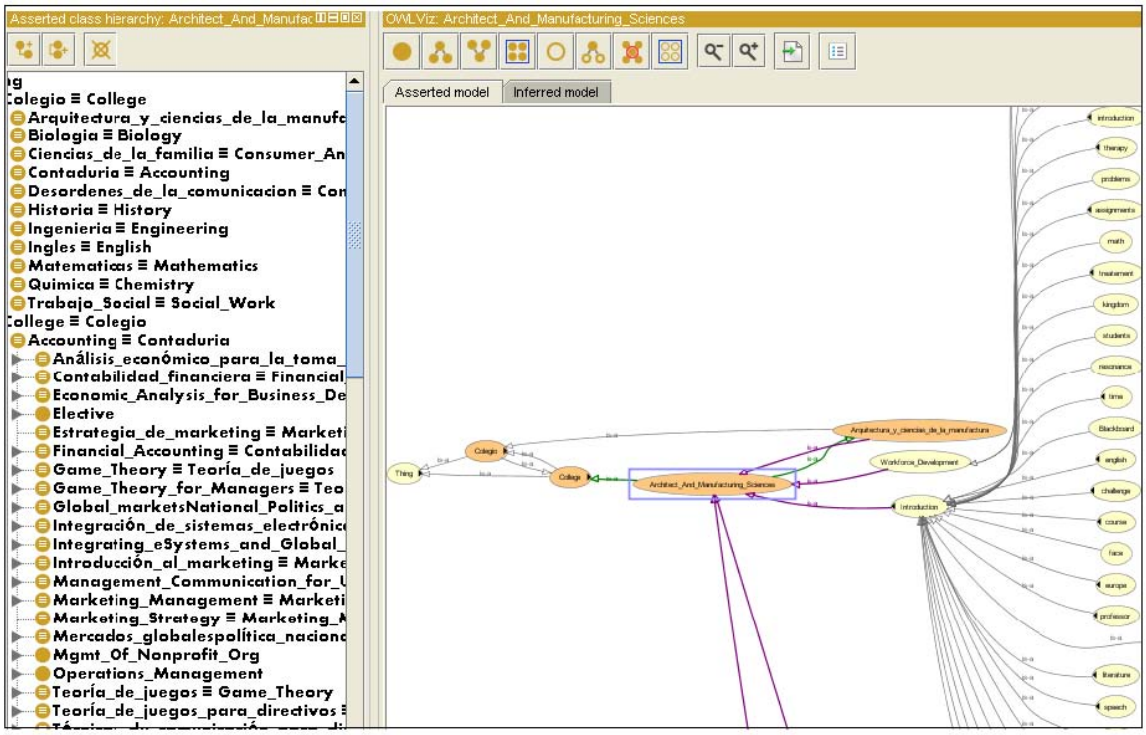


Figure 1. Hierarchical Structure of the *HyperManyMedia* Ontology.

the domain which is represented as a tree, and C_i represent a concept under R, as $R = \cup_{i=1}^{n} C_i$, where n is the number of concepts in the domain. Each concept Ci consists either of subconcepts ($C_i = \cup_{j=1}^{m} SC_{ji}$) or of leaves which are the actual documents ($C_i = \cup_{k=1}^{l} d_{ki}$). We encoded the above semantic information into a tree-structured domain ontology in OWL, based on the hierarchy of the E-learning resources. The root concepts are the colleges, while the sub-concepts are the courses, and the leaves are the resources of the domain (lectures).

IV Implementation

A. Ontology Content-based Recommendations

The idea of a Content-based recommender system in an E-learning platform can be summarized as follows: Given the lectures that the learner has visited, the platform recommends other lectures with *content* that is similar to the *content* of the viewed lectures. We build the learner's ontology profile by extracting the learner interests from the user's profile. Let $docs(U_i) = \cup_{k=1}^{l} d_{ki}$ be the documents visited by the i^{th} learner, U_i. The learner's ontology is considered as a subset of the E-learning domain ontology from Section III.A. Since the activity log of the user's activities records the visited documents (which are the leaves), a bottom-up pruning algorithm is used to extract the semantic concepts that the learner is interested in. Each learner U_i R has a dynamic semantic representation. First, we collect the learner's activities over a period of time to form an initial learner profile, as follows, let $docs(U_i) = \cup_{k=1}^{l} d_{ki}$ be the documents visited by the i^{th} learner U_i, then starting from the leaves, the bottom-up pruning algorithm searches for each document visited by the learner in the "domain semantic structure", and then increments the visit count (initialized with 0) of each visited node along with its ancestors all the way up to the root. After back-propagating the counts

$$S_{Cosine} = \frac{d^T q}{||d||^2 \cdot ||q||^2} = \sum_{j=1}^{m} d_j q_j / \sqrt{\sum_{j=1}^{m} (d_j)^2 \cdot \sum_{j=1}^{m} (q_j)^2} \quad (1)$$

of all the documents in this way in the domain structure, the pruning algorithm keeps only the concepts (colleges) and sub-concepts (courses) related to the learner's interests along with their weighted interests (which are the number of visits). When a learner searches for a lecture using a specific query q, the cosine similarity measure is used to retrieve the most similar documents d that contain the terms in the query, as shown in equation (1). As we mentioned in Section III, the *HyperManyMedia* search engine's scoring algorithm is based on the VSM. For each field, the score is computed as follows,

$$score(q,d) = coord(q,d) \times queryNorm(q) \times \sum (tf\ (t\ in\ d) \times idf\ (t)^2 \times t.getBoost() \times norm(t,d)) \quad (2)$$

Lucene[8] (Apache) defines each term for equation (2) as follows [9], where *tf* (*t in d*) is the number of times term *t* appears in the currently scored document *d*, defined as

tf (*t in d*)= *frequency*½, *id f* (*t*) is the inverse document frequency (related to the number of documents in which the term *t* appears), defined as $idf(t) = 1 + \log(numDocs/docFreq + 1)$ and *coord*(*q*,*d*) is a score factor based on how many of the query terms are found in the specified document, and *queryNorm*(*q*) is a normalizing factor used to make scores between queries comparable,

$$queryNorm(sumOfSquaredWeights) = 1/sumOfSquaredWeights^{1/2} \quad (3)$$

The sum of squared weights (of the query terms) is computed by the query weight object. For example, for a Boolean query, we compute this value using,

$$sumOfSquaredWeights = q.getBoost()^2 . \sum (idf(t) \cdot t.getBoost())^2 \text{ t in q} \quad (4)$$

Where *t.getBoost*() is a search time boost of term *t* in the query *q* as specified in the query text, or as set by application calls to *setBoost*(), there is only multi-terms boost access, and so the boost of a term in the query is accessible by calling the sub-query *getBoost*(), and *norm*(*t*,*d*) encapsulates a few (indexing time) boost and length factors: (1) document boost, is set by calling *doc.setBoost*() before adding the document to the index, (2)field boost, is set by calling *field.setBoost*() before adding the field to a document, and (3) *lengthNorm* (*field*), is computed when the document is added to the index in accordance with the number of tokens of this field in the document, so that shorter fields contribute more to the score, and LengthNorm is computed by the Similarity class in effect at indexing. When a document is added to the index, all the above factors are multiplied. If the document has multiple fields with the same name, all their boosts are multiplied together,

$$norm(t,d) = doc.getBoost() . lengthNorm(field) \cdot \prod_{field\ f\ in\ d\ named\ as\ t} f.getBoost() \quad (5)$$

When a learner searches for lectures using a specific query *q*, the cosine similarity measure is used to retrieve the most similar documents that contain the terms in the query. In our approach, these results have been re-ranked based on two main factors: (1) the semantic relation between these documents and the learner's semantic profile, and (2) the most similar cluster to the learner's semantic profile (recommended cluster). Algorithm 1 maps the ranked documents to the learner semantic profile (Category 1), where each document d_i, belonging to a learner's semantic profile, is assigned a priority ranking (α = 5.0), and each document d_i belonging to the recommended cluster (Category 2) is assigned a priority ranking (β = 3.0), while the rest of the documents (Category 3) have the lowest priority (γ = 1.0). The threshold of each parameter was decided heuristically after several trials (α = 5.0, β = 3.0, and γ = 1.0). All the documents, in each category, are then re-ranked based on cosine similarity to the query *q*. Our search engine (based on Nutch) uses optional boosting scores to determine the importance of each term in an indexed document, when adding up the document-to-query term matches in the cosine similarity. Thus a higher boosting factor for a term will force a larger contribution from that term in the sum. We modified the boosting score as follows: *field.setBoost*() = α, in case of Category1, *field.setBoost*() = β, in case of Category2, and *field.setBoost*() = γ, in case of Category3. Accordingly, all documents have been boosted and re-ranked based on two factors. Here, we are going to introduce the first factor and in the following section, the second factor. Algorithm 1 maps the ranked documents to the learner's semantic profile (learner's previous visited lectures) as Category 1, where each document d_i, belonging to a learner's semantic profile, is assigned a priority ranking (α = 5.0). This boosting score has been implemented using *field.setBoost*(), the weight is only added to the documents that the learner is interested in, based on his/her previous activities (sessions). Since we used the ontology to generate the user profile, we named this type of recommendation, ***Ontology Content-based Recommendations***.

Algorithm 1 Re-ranking a learner's search results

Input: q://keyword search
Input: //user index
Input: α, β, γ://threshold
Output: $Rank = \{d_1, d_2, ..., d_n\}$://re-rank
$Rank = \{d_1, d_2, ..., d_n\}$://*default search results for query q*
$UR_i = \cup_{j=1}^{n} SC_{ji} + \cup_{k=1}^{l} d_{ki}$
$RC = \cup_{c=1}^{c} d_c$;
foreach $d_j \in Rank$
  if $d_j \in UR_i$ then //*document is in user profile*
    $d_j.boost = \alpha$;
  end
  else
  if $d_j \in RC$ then //*document is in recommended cluster*
    $d_j.boost = \beta$;
  end
  else
  $d_j.boost = \gamma$;
  end
end
Sort Ranks based on the document boost field $d_j.boost$

B. Cluster-based Recommendations

A total corpus consisting of around 7,424 documents (lectures), was divided into 4,888 English documents and 2,536 Spanish documents. In both cases, we experimented with partitional algorithms, direct K-way clustering (similar to K-means), and repeated bisection or Bisecting K-Means with all criterion functions. We also experimented with graph-partitioning-based clustering algorithms [15]. First, for clustering English documents, we compared different hierarchical algorithms for the English corpus consisting of 4,888 documents using the clustering package Cluto [15]. The best clustering method

[8] http://lucene.apache.org/java/2_4_0/api/org/apache/lucene/search/Similarity.html

TABLE I.
ENGLISH CLUSTERS DESCRIPTIVE FEATURES

Cluster 0	angle	4.20%	prime	3.10%	line	2.60%	distance	2.60%
Cluster 1	terms	2.60%	child	2.40%	means	2.10%	stuttering	1.60%
Cluster 2	called	1.50%	war	1.40%	sort	1.20%	people	1.00%
Cluster 3	flood	5.80%	water	1.40%	building	1.40%	elevation	1.40%
Cluster 4	audit	4.40%	board	3.40%	internal	2.90%	management	2.40%
Cluster 5	zero	3.80%	grams	3.00%	fraction	2.80%	hundred	2.50%
Cluster 6	material	4.30%	materials	1.80%	process	1.50%	type	1.40%
Cluster 7	time	2.50%	times	1.90%	rainfall	1.80%	storm	1.50%
Cluster 8	voice	2.10%	vocal	1.90%	speech	1.40%	pitch	1.30%
Cluster 9	class	5.50%	java	4.20%	method	4.00%	methods	3.30%
Cluster 10	price	7.30%	market	4.40%	cost	2.60%	product	2.50%
Cluster 11	mean	2.10%	basically	2.10%	five	1.90%	data	1.80%
Cluster 12	income	4.70%	accounting	3.80%	balance	3.60%	statement	2.90%
Cluster 13	data	7.10%	system	2.80%	database	2.80%	server	2.60%
Cluster 14	children	2.60%	child	2.00%	program	1.70%	time	1.50%
Cluster 15	course	6.60%	assignments	5.80%	class	1.90%	topic	1.90%
Cluster 16	equal	4.10%	zero	3.30%	look	2.80%	*negative*	*2.60%*
Cluster 18	game	9.30%	theorem	7.30%	muhamet	5.20%	ergin	5.20%
Cluster 19	transport	4.60%	waves	3.70%	environment	3.30%	concentration	3.10%
Cluster 20	poem	2.20%	read	1.60%	look	1.30%	little	1.20%
Cluster 21	information	6.00%	systems	5.70%	technology	5.10%	organizational	3.60%
Cluster 22	test	2.40%	child	1.60%	score	1.60%	words	1.40%
Cluster 23	five	4.30%	times	4.00%	example	2.90%	nine	2.80%
Cluster 24	deviance	7.10%	social	6.60%	deviant	3.60%	identity	2.90%
Cluster 25	square	9.50%	squared	6.80%	equal	4.20%	times	3.00%
Cluster 26	western	1.50%	online	1.50%	literature	1.50%	course	1.40%
Cluster 27	times	5.00%	equal	3.60%	minus	3.40%	zero	2.70%
Cluster 28	game	5.70%	player	2.50%	strategic	2.30%	strategy	2.10%
Cluster 29	time	1.50%	product	1.30%	look	1.20%	example	1.10%
Cluster 30	angle	8.60%	equal	5.80%	triangle	3.80%	proposition	3.60%
Cluster 31	lecture	11.60%	global	2.40%	population	1.90%	species	1.80%
Cluster 32	metal	2.90%	formula	2.70%	name	2.60%	minus	2.30%
Cluster 33	market	11.70%	markets	8.90%	competition	8.80%	strategy	7.60%
Cluster 34	transportation	6.70%	land	3.10%	planning	2.80%	transit	2.50%
Cluster 35	time	3.40%	value	2.60%	markets	2.10%	resources	1.70%
Cluster 36	transportation	6.70%	land	3.10%	planning	2.80%	transit	2.50%
Cluster 37	time	3.40%	value	2.60%	markets	2.10%	resources	1.70%

TABLE II.
SPANISH CLUSTERS DESCRIPTIVE FEATURES

Cluster 0	desagradables	33.30%	aborrecible	33.30%	repugnancia	33.30%	accionistas	0.00%
Cluster 1	ciclo	7.40%	dep	4.00%	global	3.40%	azufre	3.00%
Cluster 2	contabilidad	4.20%	balance	4.10%	pasivo	3.20%	contable	1.90%
Cluster 3	precios	5.70%	producci	2.60%	fijaci	2.20%	discriminaci	2.00%
Cluster 4	product	8.60%	design	7.10%	hill	7.10%	mcgraw	7.10%
Cluster 5	programa	2.00%	coordenadas	1.80%	gui	1.60%	pdb	1.60%
Cluster 6	teorema	11.60%	conocimiento	11.60%	espesamiento	11.10%	trade	9.40%
Cluster 7	conservaci	7.40%	masa	7.30%	difusi	6.20%	volumen	5.60%
Cluster 8	ajuste	20.70%	ruido	17.30%	persistente	7.80%	stico	6.00%
Cluster 9	patente	2.20%	stephen	1.80%	patentes	1.40%	invenciones	1.30%
Cluster 10	juego	6.20%	juegos	5.90%	nash	4.90%	prueba	2.40%
Cluster 11	subastas	10.10%	equivalencia	8.20%	subasta	4.60%	licitaci	4.60%
Cluster 12	colas	16.20%	nacimiento	6.50%	muerte	6.50%	sistemas	5.20%
Cluster 13	arrays	3.50%	lista	2.60%	array	1.40%	elemento	1.40%
Cluster 14	interpretaci	83.60%	hoy	9.20%	objetivos	6.50%	los	0.60%
Cluster 15	software	3.40%	ide	1.90%	requisitos	1.80%	desarrollo	1.70%
Cluster 16	red	2.70%	fibra	2.40%	paquetes	2.40%	redes	2.30%
Cluster 17	navegador	4.40%	html	3.50%	server	3.30%	mime	2.90%
Cluster 20	kang	11.50%	arnold	9.80%	james	9.80%	barnett	9.80%
Cluster 21	reacciones	10.90%	reacci	4.30%	concentraciones	4.20%	concentraci	3.20%
Cluster 22	xml	4.20%	web	2.50%	corba	1.60%	servidor	1.10%
Cluster 23	transporte	10.30%	suelo	3.40%	planificaci	3.20%	teor	2.90%
Cluster 24	nike	1.70%	reputaci	1.60%	industria	1.40%	empresas	1.30%
Cluster 25	pasajeros	9.10%	mortalidad	8.90%	desarrollados	6.80%	vuelos	5.30%
Cluster 26	hilo	6.30%	hilos	4.20%	eventos	2.00%	deeventos	1.60%
Cluster 27	desplazamiento	7.90%	colas	7.10%	servidores	6.50%	ciudad	3.80%
Cluster 28	productividad	19.20%	primaria	11.80%	lecturas	4.20%	ecolog	3.60%
Cluster 29	amortizaci	13.00%	fiscal	5.00%	gasto	4.30%	impuestos	4.30%
Cluster 30	replicador	22.00%	ess	10.30%	din	6.90%	evolutiva	6.80%

TABLE III.
SPANISH CLUSTERS DESCRIPTIVE FEATURES CONT.

Cluster 31	mercados	5.10%	poder	4.50%	segmentos	4.10%	marketing	4.00%
Cluster 32	lotka	10.70%	nichos	9.20%	competencia	7.50%	xplique	5.40%
Cluster 33	integraci	2.30%	organizativos	1.60%	negocio	1.60%	tecnolog	1.50%
Cluster 34	ondas	17.00%	onda	7.50%	fluido	1.90%	dispersi	1.80%
Cluster 35	etiqueta	6.70%	desviado	3.70%	desviaci	3.20%	negar	2.30%
Cluster 36	juego	5.20%	juegos	4.00%	estrat	2.80%	jugador	2.00%
Cluster 37	gen	7.30%	mendel	7.10%	mutantes	4.20%	genes	3.70%
Cluster 38	aritm	11.50%	operadores	7.30%	estructuras	5.70%	control	3.50%
Cluster 39	duraderas	5.30%	recurso	5.00%	ventajas	3.60%	podemos	2.40%
Cluster 40	consultas	1.80%	bases	1.70%	filas	1.60%	datos	1.60%
Cluster 41	memoria	2.10%	java	1.90%	clases	1.90%	clase	1.80%
Cluster 42	cognitivo	6.30%	decisi	5.90%	aprobarla	5.60%	conocimientos	5.60%
Cluster 43	nodo	9.60%	nodos	5.60%	sub	3.00%	rboles	2.70%
Cluster 44	aparcamiento	5.50%	transporte	4.50%	viajes	3.00%	mit	2.60%
Cluster 45	lagos	2.60%	especie	2.60%	norte	2.50%	avi	2.10%
Cluster 46	dise	3.10%	especificaciones	2.90%	necesidades	2.70%	piz	1.80%
Cluster 47	contestar	3.30%	redacte	1.70%	feedback	1.60%	quejaslea	1.40%
Cluster 48	poblaci	3.80%	densidad	3.60%	fecundidad	3.40%	edades	2.60%
Cluster 49	huella	12.00%	ecol	10.40%	demogr	10.00%	poblaci	5.10%

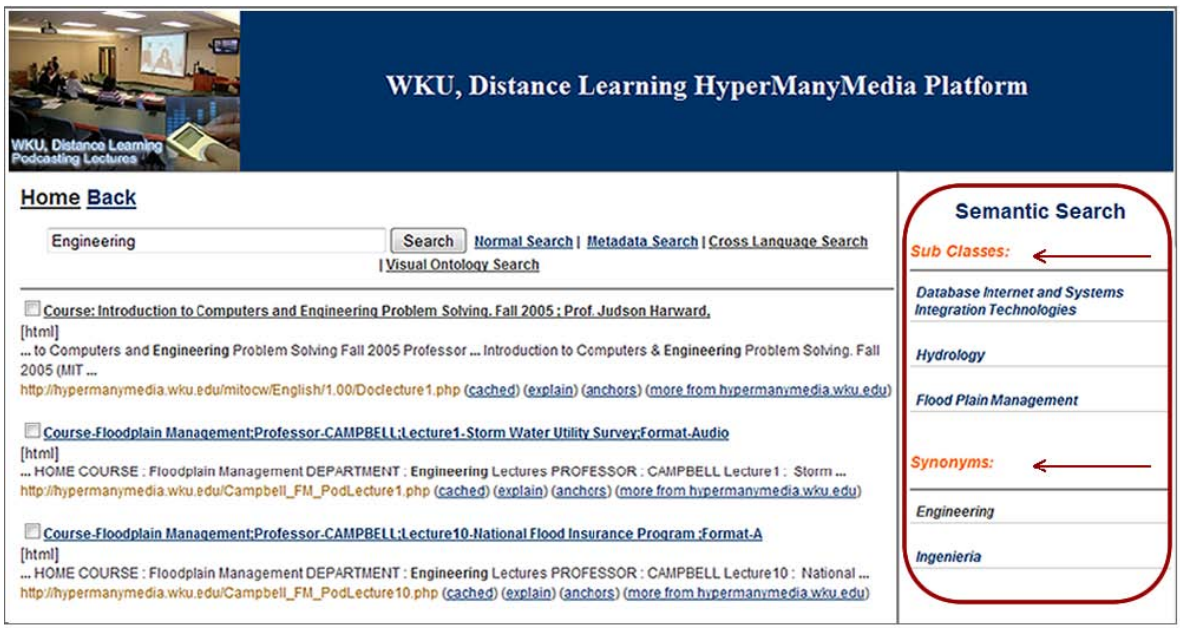


Figure 2. Semantic Terms Recommendations.

for the English corpus, which produced the highest *Purity* = 0.959 and with the lowest *Entropy* = 0.05, was the *Agglomerative Method*, with *Number of Clusters*= 38, using *Clustering Criterion Function* and *Cosine Similarity Measures* as *inter-object similarity measure*, as shown in equation (1). Table I shows the descriptive features in each cluster (those features that we added to the ontology). Second, for clustering Spanish documents we also compared different hierarchical algorithms for the Spanish corpus consisting of 2,536 documents. The best clustering method for this corpus, which produced the highest *Purity* = 0.927 and with the lowest *Entropy* = 0.140, was the *Agglomerative Method*, with *Number of Clusters* = 50, using *Clustering Criterion Function* and *Cosine Similarity Measures* as *inter-object similarity measure*. Table II and Table III show the descriptive features in each cluster (those features that we added to the ontology). We consider extracting the most similar (recommended) cluster C_i = BestCluster, which is summarized by the Top n keywords (significant or frequent terms) to modify the learner's semantic ontology and adding the cluster's terms as semantic terms under the concepts (parent nodes) that these documents belong to, as a Rule-based recommendation. In Algorithm 1, we defined this rule as Category 2, where each document d_i belonging to the recommended cluster is assigned a priority ranking ($\beta = 3.0$). This boosting score has been implemented using *field.setBoost*(). When a learner searches for lectures using a specific query *q*, the cosine similarity measure is used to retrieve the most similar documents that contain the terms in the query. Those documents are re-ranked based on the weighting factor β. Also, we name this type of recommendation, ***Cluster-based Recommendations***.

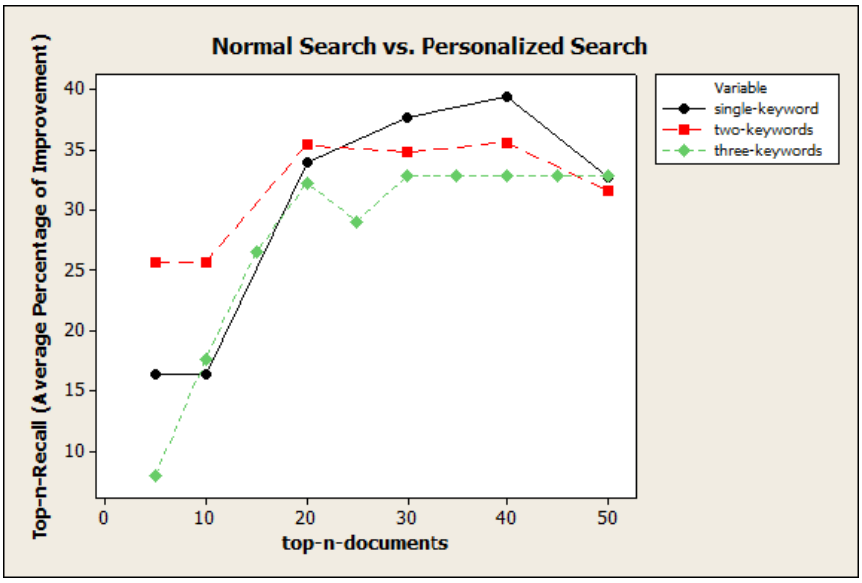


Figure 4. Average Percentage of Improvement in Top-n Recall.

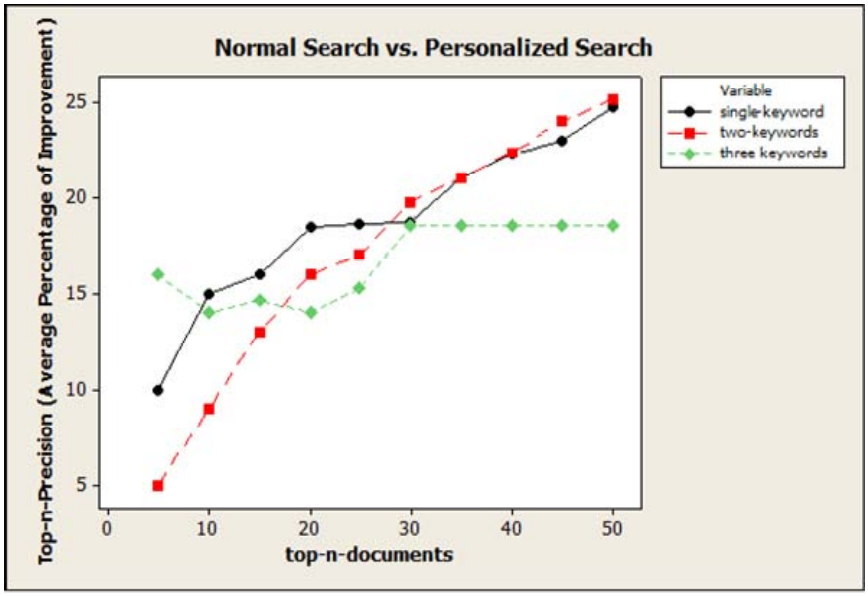


Figure 3. Average Percentage of Improvement in Top-n Precision.

Our current ontology consisting of ~40,000 lines of code, and it can be downloaded from URL[9].

B. *Interest-based Recommendations*

We provide the learner with semantic term recommendations based on his/her visited concepts. We consider this type of recommendation as *Rule-based.* Since the ontology represents *concepts* and *relationships*, *properties*, *functions* and *rules* among these concepts. For each query *q* submitted by a learner, a semantic mapping between the query and the learner's semantic profile brings all the concepts/subconcepts/cluster-based-recommended terms. This framework allows the learner to navigate through the semantic structure of his/her query, as shown in Figure 2 by clicking on one of the recommended terms. For more detail on modeling a learner's interests, refer to our previous work [18], [17]. The effect of this action is to add the selected term to the query and repeat the search. Therefore the search is finally personalized via a query expansion using the recommended term that is selected. We name this type of recommendation, ***Interest-based Recommendations***.

The difference between Content-based recommendations and Interest-based recommendations (Rules-based recommendations) is that in the latter, the user is provided with recommendations not only based on his/her profile, as in Content-based recommendations, but

[9] http//lucene.apache.org/java/2_4_0/api/org/apache/lucene/search/Similarity.html
[10] http//:://hypermanymedia.wku.edu

in addition, the user is provided with recommendations based on an extended ontology by using the First Order Logic. In this case, we defined the following entities: has_College, has_Course, has_Language, has_Lecture, has_Professor, sub_Class_Of.

In addition, each entity has different characteristics (*Functional, Description*), for example, we illustrate the characteristics of *Entity = has_College*: *(Description:* College, *Equivalent classes*: Colegio, *Superclasses:* Thing, *Members:* Accounting, Architecture_and Manufacturing, Biology, etc., *Disjoint:* Sub classes). Two most important definitions used in our ontology design are the following: (1) Equivalent classes: Equivalent classes equal to ≡ relation, to mention some of these entities (*College* ≡ *Colegio*, *Engineering* ≡ *Ingenieria*, *English* ≡ *Ingles*,..., *Social Work* ≡ *Trabajo Social*, *Chemistry* ≡ *Quimica*, etc.) and (2) Sub_Class_Of: Related to the hierarchy design of our domain: <Cluster_descriptive features *is−a* sub_Class_Of Lecture>, etc.

V. Experimental Analysis

Several evaluation metrics have been introduced in the literature, such as Recall, Precision, F-measure, Harmonic Mean, E-Measure, User-Oriented Measure (coverage, novelty), expected search length, satisfaction, frustration, etc. The most widely used ones in evaluating search engines have been Top-n Recall and Top-n Precision. **Top-n Recall** is the number of relevant retrieved documents among the top *n* retrieved documents divided by the total number of relevant documents, and **Top-n Precision** is the number of relevant retrieved documents within the top *n* divided by *n*. For example, starting with the top 50 results and going down to the top 10 search results: *n* = 50, 40, 30,...,10, e.g., at *n* = 50, the *top*-50 search results are used for recall computing the precision. Therefore, we used *Top-n-Recall* and *Top-n-Precision* to measure the effectiveness of re-ranking based on the learner's semantic profile (testing set). For the evaluation, we used our own semantic search engine[10] to evaluate each query, and compute the Top-n-Precision and Top-n-Recall for normal search and for personalized semantic search for each learner. The problem with evaluating a real search engine is that you cannot compare results obtained with different datasets. First, using a different dataset will return results not related to the content of the repository and in this case, our own search engine evaluation's results will definitely be better on our dataset. Second, from an architectural stands point, we cannot compare our search engine results with another search engine because the only search engine that we are aware of that has an architecture that supports the integration of semantics is the one we used, *Nutch*.

Figure 3 shows the *Average Percentage of Improvement in Top-n Precision*, whereas, Figure 4 shows the *Average Percentage of Improvement in Top-n Recall* for the *personalized* search over the *normal* search, with three sizes of queries (1, 2, and 3 keywords). We used keyword queries extracted from the logs that users typed the most for searching content. For each length of query, we used the Top-100 most used queries. The personalized semantic search shows an improvement in precision that varies between 5-25 %. This improvement is noticeable between the top-30 and top-50 search results for single-keyword and two-keywords queries. The recall results show a noticeable improvement in recall between top-20 and top-40. Also, we can summarize the impact of the query size by noticing that Precision is better when the size of a query was 1 or 2; whereas, Recall starts with a better results for queries of size 2 till Top-20, then both queries of size 1, 2 converge almost to the same results. Overall, these results show the effectiveness of the re-ranking based on the learner's semantic profile.

VI Conclusion

In this paper, we presented a hybrid recommender engine to personalize search in the E-learning domain. This engine is driven by multi-ontology models: ontology content-based recommendations (domain ontology model), and ontology rule-based recommendations (cluster-based and interest-based). We illustrated the methods, concepts, and architecture to integrate a recommendation engine into an E-learning search system. We demonstrated the design of the *HyperManyMedia* ontology using the Protégé framework. In this context, this ontology is composed of a hierarchy of concepts and sub-concepts that represents colleges, courses, and lectures. Also, we described the implementation of Rule-based recommendations by using clustering techniques to extract descriptive features from clusters, those features have been added to the domain ontology under the related concepts using the Protégé framework. In addition, we implemented a semantic mapping between the query and the learner's semantic profile to present the user's interest. Finally, each type of these recommendations influenced the re-ranking of the retrieved documents with different factors. Our experiments were carried out on the *HyperManyMedia* semantic search engine at Western Kentucky University. We used Top-n-Recall and Top-n-Precision to measure the effectiveness of re-ranking based on the learner's semantic profile. Overall, the search results showed the effectiveness of the re-ranking based on personalization.

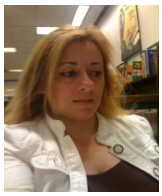
References

[1] M. de Gemmis, G. Semeraro, P. Lops, and P. Basile. A Retrieval Model for Personalized Searching Relying on Content-based User Profiles.

[2] G. Grenier. Path to document recommendation services: Technologies that enabled the development of on-line information systems. In *Presentation held at the ACS National Meeting*, volume 230, 2005.

[3] K. Hammouda and M. Kamel. Data mining in e-learning. *E-Learning Networked Environments and Architectures: A Knowledge Processing perspective", series: Advanced*

Information and Knowledge Processing, Springer Book Series, 2007.
[4] T. Joachims. Optimizing search engines using clickthrough data. *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142, 2002.
[5] T. Joachims, L. Granka, B. Pan, H. Hembrooke, F. Radlinski, and G. Gay. Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. 2007.
[6] T. Joachims and F. Radlinski. Search engines that learn from implicit feedback. *Computer*, 40(8):34–40, 2007.
[7] M.K. Khribi, M. Jemni, and O. Nasraoui. Automatic Recommendations for E-Learning Personalization Based on Web Usage Mining Techniques and Information Retrieval. *Educational Technology and Society*, 2009.
[8] Lihong Li, Wei Chut, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *WWW2010*. Yahoo! Labs, 2010.
[9] [10] B. Mobasher, R. Cooley, and J. Srivastava. Automatic personalization based on web usage mining. *Communications of the ACM*, 43(8):142–151, 2000.
[10] B. Mobasher, R. Cooley, and J. Srivastava. Automatic personalization based on web usage mining. *Communications of the ACM*, 43(8):142–151, 2000.
[11] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa. Effective personalization based on association rule discovery from web usage data. In *Proceedings of the 3rd international workshop on Web information and data management*, page 15. ACM, 2001.
[12] O. Nasraoui, R. Krishnapuram, and A. Joshi. Mining web access logs using a fuzzy relational clustering algorithm based on a robust estimator. *Eighth International World Wide Web Conference, Toronto, Canada*, 1999.
[13] O. Nasraoui, M. Soliman, E. Saka, A. Badia, and R. Germain. A Web Usage Mining Framework for Mining Evolving User Profiles in Dynamic Web Sites. *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, pages 202–215, 2008.
[14] A.W. Neumann. *Recommender Systems for Information Providers: Designing Customer Centric Paths to Information*. Springer Verlag, 2009.
[15] M. Rasmussen and G. Karypis. gcluto: An interactive clustering, visualization, and analysis system. *CSE/UMN Technical Report: TR# 04*, 21, 2008.
[16] O.R. Zaiane. Building a recommender agent for e-learning systems. *Computers in Education, 2002. Proceedings. International Conference on*, pages 55–59 vol.1, 3-6 Dec. 2002.
[17] Leyla Zhuhadar, Olfa Nasraoui, and Robert Wyatt. Automated discovery, categorization and retrieval of personalized semantically enriched e-learning resources. *International Conference on Semantic Computing*, 0:414–419, 2009.
[18] Leyla Zhuhadar, Olfa Nasraoui, and Robert Wyatt. Dual representation of the semantic user profile for personalized web search in an evolving domain. In *Proceedings of the AAAI 2009 Spring Symposium on Social Semantic Web, Where Web 2.0 meets Web 3.0*, pages 84–89, 2009.



Leyla Zhuhadar received the Ph.D. degree in Computer Engineering and Computer Science from the University of Louisville, Louisville, in 2009. Currently, she is a Research Scientist at Western Kentucky University and an Adjunct Assistant Professor at the Department of Computer Engineering and Computer Science (CECS), at the University of Louisville. Her research interests are in Knowledge Acquisition from the Web, Information Retrieval, Ontology Engineering, Semantic Web, Metadata for Accessible Learning Objects. She designed and implemented two working research platforms in the e-learning domain, *HyperManyMedia* and the Semantic Repository. She is a member of IEEE, IEEE Women in Engineering, IEEE Computer Society, IEEE Education Society, the ACM, SIGKDD, SIGACCESS, SIGIR, Web Intelligence Consortium (WIC), and the AIED.

Olfa Nasraoui is the founding Director of the Knowledge Discovery and Web Mining Laboratory, at the University of Louisville, where she is also an Associate Professor of Computer Engineering and Computer Science and the Endowed Chair of e-Commerce. She received the Ph.D. degree in Computer Engineering and Computer Science from the University of Missouri, Columbia, in 1999. From 2000 to 2004, she was an Assistant Professor at the University of Memphis. Her research interests include data mining, Web mining, stream data mining, and computational intelligence. She is a member of IEEE, IEEE Women in Engineering, and in the last 10 years, has been active in the SIGKDD community, notably by organizing the WebKDD workshop on Web Mining and by serving as Vice-Chair on Data Mining conferences, including KDD 2009, ICDM 2009, and WI 2009. She is a recipient of a US National Science Foundation Faculty Early Career Development (CAREER) Award, and a Best Paper Award in the Artificial Neural Networks in Engineering Conference. She has published more than 100 publications, and acquired close to $2M in funding for research from NSF, NASA and other agencies.