Recommendation Based on Deduced Social Networks in an Educational Digital Library

Monika Akbar  
Cyber-ShARE & Dept. of CS  
UTEP, El Paso, TX  
makbar@utep.edu

Clifford A. Shaffer  
Dept. of Computer Science  
Virginia Tech, Blacksburg, VA  
shaffer@vt.edu

Weiguo Fan  
Dept. of Computer Science  
Virginia Tech, Blacksburg, VA  
wfan@vt.edu

Edward A. Fox  
Dept. of Computer Science  
Virginia Tech, Blacksburg, VA  
fox@vt.edu

ABSTRACT

Discovering useful resources can be difficult in digital libraries with large content collections. Many educational digital libraries (edu-DLs) host thousands of resources. One approach to avoiding information overload involves modeling user behavior. But this often depends on user feedback, along with the demographic information found in user account profiles, in order to model and predict user interests. However, edu-DLs often host collections with open public access, allowing users to navigate through the system without needing to provide identification. With few identifiable users, building models linked to user accounts provides insufficient data to recommend useful resources. Analyzing user activity on a per-session basis, to deduce a latent user network, can take place even without user profiles or prior use history. The resulting Deduced Social Network (DSN) can be used to improve DL services. An example of a DSN is a graph whose nodes are sessions and whose edges connect two sessions that view the same resource. In this paper we present a recommendation framework for edu-DLs that depends on deduced connections between users. Results show that a recommendation system built from DSN-dependent parameters can improve performance compared to when only text similarity between resources is used. Our approach can potentially improve recommendation for DL resources when implicit user activities (e.g., view, click, search) are abundant but explicit user activities (e.g., account creation, rating, comment) are unavailable.

Categories and Subject Descriptors

H.3.7 [Information Storage and Retrieval]: Digital Libraries

General Terms

Design, Measurement

Keywords

Deduced social network, recommendation

1. INTRODUCTION

Educational digital libraries connect content with users. The vast amount of resources and activities within an edu-DL lead to the problem of information overload — how to find useful resources in a reasonable amount of time when there is too much information to manage. The abundance of educational resources in an educational DL (edu-DL) provides opportunities but also creates problems for users when searching for high quality material. Without information on the quality of resources it becomes difficult to locate usable resources from hundreds, if not thousands, of choices. Ideally the user community provides feedback in various forms such as ratings, reviews, comments, etc. that can be helpful to gauge the quality of the resources. However, edu-DLs often host collections with public access that users can navigate through without needing to create an account and provide their feedback on the resources. Lack of user accounts poses a difficulty when building user models since these depend on attributes derived from user accounts.

Using activity to deduce latent user networks can help direct users to relevant content. Such deduced user networks can be used to improve DL services or introduce personalization. Capturing user activities within an edu-DL can shed light on the potential usefulness of a resource. Making practices visible not only can show users what others are doing but also can motivate users to explore content and share their experience. We proposed the concept of deduced social network (DSN) [1] to capture user activities in the absence of user accounts. Prominent applications of such information include refining existing services and providing recommendations on contents in the absence of profiles and ratings. While most current recommendation systems rely on active user data (e.g., feedback, reviews, ratings, buying history, etc.), DSN-based recommendation solely depends on passive user data (e.g., clicks, pageviews, times in pages, etc.) generated by anonymous users. In this paper we propose a DSN-based recommendation framework that uses lo-
gistic regression to model page viewing trends of anonymous users. The model takes into account a user’s page viewing history, page viewing trends captured using DSNs, and text similarity between page titles. As our testbed we use the AlgoViz Portal\(^1\) which collects metadata on Algorithm Visualizations and provides community support. Experimental results show that DSN-based recommendation performs better compared to when only text similarity is used. This framework can be particularly useful for recommending resources in DLs that have an active user base but few user accounts.

The rest of the paper is organized as follows: Section 2 provides a review of related literature. We describe the concept of deduced social network in Section 3. Section 4 provides the methodology for DSN-based recommendation, while Section 5 presents the overall recommendation framework. We present experimental results in Section 6 and discussion in Section 7.

2. RELATED WORK

Social navigation methods are used to guide users in an unfamiliar information space, but these methods largely depend on previous user feedback/ratings [2]. In cases where user feedback is scarce, rating-based systems can prove insufficient to derive useful usage information. Recommender systems depend on various user attributes to recommend resources. Such attributes can come from user profiles [3]. Explicit activities such as ratings or comments also act as features for building models for recommendation [4]. Lack of user profiles leads to the use of navigational information in recommendation systems. Chen [5] converts access patterns to two measures, Web access graph (WAG) and page interest estimator (PIE), to predict a user’s interest in a certain page. Networks built using navigational information are used in many areas to model and predict user behavior [6,7,8]. However, many of these networks depend on extensive user activities, such as building a graph of pages that the user viewed [5], to build the model. Others depend on the page attributes, such as link structure [7], to provide recommendation. In contrast, our approach works with sessions only and relies on page titles and URLs.

Personalization in a digital library can take many forms. Personalization can be done based on an individual user profile, group membership, resource category, or perceived outcome [9]. Often digital libraries contain metadata for resources. Retrieving useful information from metadata can be difficult since the quality of metadata varies from data provider to data provider. In cases where the resources are present in the library, algorithms have been proposed to extract metadata that can be used in recommendation systems [10]. Although our proposed system is intended for digital libraries, it does not depend on any library-specific information, thus making it easily adaptable to other domains. Since we do not rely on profiles of registered users, our approach is particularly suitable for DLs with mostly anonymous users.

3. DEDUCING NETWORKS IN EDU-DLs

Content in educational DLs mostly is free to use. The expectation, experience, and opinion related to educational resources vary from user to user. Even after a resource is used there is little motivation to provide feedback since evaluation activities take time and seem not to produce any tangible benefit for the user. This problem is made harder by the fact that the user base of most edu-DLs is small compared to the large user base of e-commerce sites. In most cases, the users of an edu-DL are not required to create an account. All of this makes it difficult to identify the users, their preferences, or usage trends within an edu-DL.

While usage trends are extensively used by other online communities, those communities show certain traits that are missing or subtle in edu-DLs. Those include a large and diverse user base, and mandatory user account/profile creation to access the service. Many successful online communities are e-commerce sites that actively encourage the buyer to share their experience and opinion of a product purchased.

A user’s activity within a DL can be grouped into two broad categories: explicit and implicit. Explicit user activity includes the tasks that generate visible outcomes such as comments, ratings, feedback, etc. Implicit user activity comprises the tasks that are not visible to other users, such as pageviews, searching, browsing, downloads, etc. Repeated user activities that are similar in nature generate behavioral patterns. These patterns have the potential to lead users to resources that other users have explored. Often, implicit user activities are recorded in user logs that can allow us to construct a network of users. Such networks, which can connect one user with another based on their navigation history, can be interpreted as deduced social networks.

A user base that generates explicit user activities makes it easier to identify user groups and usage patterns. However, while educational DLs experience a significant amount of traffic, they often lack explicit user activities in the form of reviews, ratings, comments, etc. In the absence of explicit user activity, we can depend on implicit user actions to identify such trends. We proposed the concept of deduced social network (DSN) [11,1], a social network of users and objects, which is independent of user profiles and entirely depends on log data to connect users based on their activities. DSN is a flexible concept such that it can be used to connect users as well as objects within educational DLs. Analysis of the DSN can lead to findings that have the potential to guide users through the information space of educational DLs.

Usage logs often show key pieces of information like average time spent on a certain page, bounce rate, and exit percentage for a webpage. Such information can be used to deduce connections between users resulting in deduced social networks (DSNs). Figure 1 shows examples of deduced social networks and their potential to reveal interesting information. The top of the figure shows a possible usage log with information such as user ID, user’s topics of interest (TOIs) and URLs visited. We can construct various DSNs based on this information. Figure 1 shows two such networks built based on TOIs and URLs visited. An overlap of the two networks shows a group of users who share a topic of interest and have visited the same URLs. As seen from this example, the deduced social network is a graph that imposes thresholds on the edges. A DSN is defined as:

**Definition 1.** A Deduced Social Network (DSN) is a graph with tuple \((V, A, k)\), where:

1. \(V\) is a set of vertices,
2. \(A\) is the set of attributes of \(V\) which are used to create edges, and

\(^{1}\text{http://algoviz.org/}\)
3. $k$ is a constant or a function that returns the minimum number of elements of $A$ that must be common between two vertices to create a connection (i.e., edge) between them.

In Figure 1, users are $V$, topics of interest and URLs visited are the attributes $A$ that are used to create the two networks, and $k \geq 1$.

### 4. DSN-BASED RECOMMENDATION

Recommendation systems are widely used for easing information overload [12, 13]. These systems can be grouped into three broad categories: content-based recommendation, collaborative recommendation, and hybrid recommendation. In content-based recommendation, the similarity between content is the leading factor for recommending similar resources [14]. Collaborative recommendation, however, depends on similarity of user profiles and activities [15, 13]. There should be a considerable number of users with accounts and activities (e.g., rating, review) in the system for collaborative recommendation to perform well. The user accounts and activities are used to build user models which are later used to recommend resources to a new or existing user. The hybrid recommendation approach combines content-based recommendation and collaborative recommendation.

In the absence of explicit user activities, DSNs can provide insight into existing user trends. DSN-based recommendation, when used with text-based recommendation, would result in a hybrid that uses the DSN derived information to generate a model to recommend content. Figure 2 shows the workflows in DSN-based recommendation. We begin by constructing DSNs based on AlgoViz log data from Fall 2009 (August 1 to December 31) and Spring 2010 (January 1 to May 31). Although it is a continuous timeline, we split it into two segments to follow the traffic trends seen in Fall and Spring semesters. The DSN-building step takes two parameters: the time range when the network is built (e.g., Fall 2010, January 2013) and the connection threshold $k$. These parameters control the size and connectivity of the resultant network. A longer time range provides more log data and produces a larger network. A smaller connection threshold results in more connectivity and creates a denser network.

The constructed DSN becomes an input for a network partitioning module that uses modularity clustering [16] to detect groups in networks. Modularity clustering is dependent on edge betweenness — a measure that assigns weight to an edge based on the number of shortest paths between pairs of vertices containing this edge. If a network contains multiple groups then the number of edges connecting the groups will be less than the number of edges within the group, and all shortest paths between those groups will contain one of those edges that connect the groups. Thus, the edges that connect the groups will have relatively higher edge betweenness values. The groups detected using modularity clustering, $g_1, g_2, \ldots, g_n$, are used in the recommendation phase. We build a model using logistic regression to estimate a score for a pair of pages that might be seen in a particular user session. The model takes content based similarity, user group information, and user activity (both within his/her group and across other groups). Finally, we utilize the outcome of this model to recommend resources to the user.

#### 4.1 Building models using logistic regression

Our objective is to recommend content to a user based on which group in the DSN is most similar to that user. Our observation is that group-based recommendation performs better for identifying user interest than text-based recommendation. The groups identified from the DSN bring users with similar interest together in the same partition. Resources used by peers in the same group are considered candidates for the recommendation. We construct a model with logistic regression to predict the probability of co-occurrence for a pair of resources used by a user.

Consider a situation where each group $g_i$ contains a set of users $u_{i1}, u_{i2}, \ldots, u_{ij}$. Each user has at least one but possibly multiple sessions. Each session contains pages $P = p_1, p_2, \ldots, p_k$ visited during that session. The session information can be used to create objective data, also known as ground truth, to train the model. We model the binary ground truth, whether a pair of pages $p_i$ and $p_j$ exists (or not) for a user $u_j$ in the log data, in one equation (see Equation 1) using content similarity, the frequency of $p_i$ and $p_j$, and information collected from the user’s group $g_k$. This is based on our assumption that two pages are likely to be related if they appear in the same session. Note that, for each user, each session is considered separately. That is we do not break or merge the sessions associated with a user. The question we try to answer is: Given that page $p_i$ was seen by user $u_j$ who belongs to group $g_k$, how likely is it that $u_j$ will view page $p_j$?

The use of content information alone in the modeling cannot capture the community dynamics in the recommendation. Dependence on user trends entirely, on the other hand, can be risky, especially when user activities are scarce. The proposed recommendation model uses two types of information that reflect its hybrid nature: text similarity and user activity. The log table we use contains the title of the page and its URL — both of which are used as text information for a resource. The DSNs on the other hand will provide us with collaborative information.

To determine whether a user $u_j$ of group $g_k$ has seen pages
\[ g(x) = \frac{\pi(x)}{1 - \pi(x)} = e^{\beta_0 + \beta_1 x_1 + \cdots + \beta_4 x_4} \]

where \( \beta_0 \) is the intercept of the underlying linear regression model and \( \beta_i \) is the regression coefficient for predictor variable \( x_i \). Note that the value of \( \pi(x) \) will range from zero to one. The inverse of the logistic function is called logit, which is defined as:

\[ \pi(x) = \frac{1}{1 + e^{\beta_0 + \beta_1 x_1 + \cdots + \beta_4 x_4}} \]

The first two terms in this equation capture the text similarity and the last two terms represent group information.

If a certain page pair does not appear in a DSN, this model will use text-based information to find similarity between the pages. When the page pair is present in the DSN (that is, at least two users viewed these pages together in some sessions), the model includes the group information through \( l \) and \( m \). While \( l \) provides an idea about the general user interest in a page pair across all the groups in the DSN, \( m \) shows the popularity of a page pair in a certain group. In particular, \( m \) represents the average number of times each user of the group viewed this page pair.

Probabilistic modeling and optimization algorithms are techniques for building a robust model [17, 18]. We use logistic regression [19], which is a probabilistic classification method that performs well in situations where the dependent variable is binary. A logistic regression uses a logistic function which has a outcome between zero and one. Logistic regression is particularly suitable for our case because it can take values which may be discrete, continuous, or categorical, and do not need to be correlated. Furthermore, \( \pi(x) \) always returns a value between zero and one, which can be converted to conditional probability of class membership (i.e., pair exists/does not exist). To rank the recommendations for a particular page \( p_i \) seen by user \( u_j \), we can directly order pages \( p_n \) based on the conditional probability of existence of the pairs \( (p_i, p_n) \) where \( p_n \) can be any other page than \( p_i \). Logistic regression requires both positive examples (existence of a page pair in a session) and negative examples (absence of a page pair in any session). We generated pairwise positive examples from the log data. We created the negative examples by altering the positive data set, randomly generating pairs, and validating their absence in the log data. We used the LingPipe API \(^2\) to implement logistic regression.

\subsection*{4.2 Resource pair proximity model}

We use two AlgoViz DSNs created from log data captured in Fall 2009 and Spring 2010. The sessions are the nodes and an edge between two sessions indicate they share \( k \) common pages. The connection threshold \( k \), which affects the density of the DSN, is set at 10. Density [20] for a network with edges \( E \) and vertices \( V \) is defined as:

\[ \text{density} = \frac{2 \times |E|}{|V| \times (|V| - 1)} \]

Lowering the connection threshold adds more edges into the network, thus making the network more susceptible to outliers. On the other hand, with a higher threshold, as

\(^2\)http://alias-i.com/lingpipe/
the network starts to get sparse it may lose important information. As seen from Figure 3, the density of the DSN changes rapidly until \( k = 8 \) and starts to stabilize around \( k = 14 \). Our observation is that \( 8 < k < 14 \) is a good range for a threshold that will generate informative DSNs without much noise.

Partitioning the networks results in the detection of six groups in the Fall 2009 DSN and 12 groups in the Spring 2010 DSN. Figure 4 shows the number of users for each cluster detected in these DSNs. Both DSNs have three groups of users with more than 100 users. However, the number of groups with fewer users are more in the Spring 2010 DSN, indicating the presence of more users with diverse interests. Note that the number of users in the Spring 2010 DSN (i.e., 837) is twice the number of users in the Fall 2009 DSN (i.e., 413). The increase in users could be a reason for having more groups in the Spring 2010 DSN.

Figure 5 provides the details of model building. Logistic regression needs both positive and negative examples to build a model. We use the log data to generate all pairs of pages, \((p_x, p_y)\), that appear in any session. For each pair \((p_x, p_y)\), we compute several values. Parameter \( l \) stores the number of sessions that contain \((p_x, p_y)\). It acts as a global indicator of how frequently a pair of pages appear together in sessions regardless of any group. Group information is considered in parameter \( m \) which is computed by dividing the number of times a pair of pages appear together in any session in a given group \( g_k \) by the number of users in \( g_k \) (see Equation 2). For a group, this parameter provides information on the average times a user viewed this page pair. In most cases, the denominator helps offset the effect of large groups or the most active users within a large group.

Page titles are represented using TF-IDF [21] and we compute the similarity between titles of pages in pair \((p_x, p_y)\) using the cosine similarity measure [22]. The log also contains the page URL along with the page title. We include this information in the model equation under parameter \( LCP \). We use a different measure to identify if two pages are of the same type, based on the URLs. Instead of measuring the similarity between URLs, in this case we compute the longest common prefix (LCP) of the URLs. Usually, the greater the LCP, the closer the page type.

The page pairs and computed information such as \( l \) and \( m \) form the positive examples for the logistic regression. In order to build a good model we also need negative examples. Since the log only contains pages visited in any given session, we generate negative examples, where a pair of pages was not visited in a session. The Negative Generator module of Figure 5 attempts to create an equal number of negative examples of two types.

1. **All negative examples**: Select a pair of pages \((p_x, p_y)\) such that \((p_x, p_y)\) and \((p_x, p_y)\) appear in some sessions but \((p_x, p_y)\) does not. Having a common page \( p_1 \) in two pairs of pages indicates that pages \( p_x \) and \( p_y \) may share something in common. We use \((p_x, p_y)\) as a negative example. Although the pair of pages \((p_x, p_y)\) does not appear in any session in any group, the values of parameter \( l \) and \( m \) are both one which points to the negative example itself.

2. **Inter-group negative examples**: For a given group \( g_k \), select a pair of pages \((p_x, p_y)\) such that it appears in session \( sess_{i,g_k} \) but does not appear in any session \( sess_{j,m} \) in group \( g_m \). This page pair from group \( g_k \) is then used as a negative example for group \( g_m \). Only the value of parameter \( l \) computed for \((p_x, p_y)\) in group \( g_k \) remains the same. The value of \( m \) is one for \( g_m \).

Once we have computed the positive and negative examples, we use logistic regression to build a model for page pairs using both example sets following Equation 1. We call the resultant model a resource pair proximity model. This model predicts whether a pair of pages will appear together in a session for a user belonging to a particular group.

5. **RECOMMENDING RESOURCES**

Figure 6 shows the framework used to recommend content based on the classifiers. We begin with the resource pair proximity model and a list of pages \( p_1, p_2, \ldots, p_m \), which were used to train this model. The resource pair proximity model depends on four terms, \( l, m, LCP, \) and similarity and corresponding coefficients \( \beta_1, \beta_2, \ldots, \beta_4 \), to estimate the probability of viewing two pages in a session. In the absence of group information (i.e., \( m \)) this model depends on the other DSN-derived parameter \( l \). When both of these parameters are null it indicates that the given pair of pages was not visited by any user in any session. In this case, the resource pair proximity model depends on the title and
URL similarity. When a user starts viewing pages in AlgoViz, a session is automatically created that logs the pages visited. To recommend pages to this user using the resource pair proximity model, we assign this user to a group, as described next. This assignment will allow us to compute the values for group-dependent parameter $m$.

Clustering algorithms are often used to assign a user to a group [23]. Approaches such as point-based measures or cluster centroids are often used to assign newly arriving points to an existing cluster. We use the centroid-based approach [23] since it is a popular scheme for compact clusters which are similar to the clusters we see in the AlgoViz DSN. In order to assign a user to a group, we first compute group centroids. Group centroids are computed by taking the average of all the weights of the various terms present in a page $pg$ (i.e., title and URL) visited by the group members. Given a set of pages and their corresponding vector representations $pv$, the centroid vector $gc$ is defined as

$$gc = \frac{1}{pv} \sum_{pg \in PV} pg$$

(7)

A similar scheme is used to find a centroid for a user. We select the pages visited by the user and use the page vectors to compute user centroid $uc$. After the centroids are computed we use Euclidean distance to measure the distance between the user and the groups. Euclidean distance, $diss(gc, uc)$, between two vectors is computed as:

$$\sqrt{(gc_1 - uc_1)^2 + (gc_2 - uc_2)^2 + \cdots + (gc_n - uc_n)^2}$$

(8)

where $n$ is the length of the vectors.

The user is then assigned to the group that has the least distance from the user centroid. Once a user $u_j$ is assigned to a group, we take pages $p_1, p_2, \ldots, p_t$ from his session and pages viewed by other users $p_{o1}, p_{o2}, \ldots, p_{o_v}$ to create all possible page pairs $(p_x, p_{o_y})$. One page of $(p_x, p_{o_y})$ comes from the session of $u_j$ and the other page comes from the sessions of other users (it also may appear in $u_j$’s session). We compute the estimated probability of viewing these pages together in a session for all these page pairs with Equation 1. Since the values range from zero to one, they can be used to rank the pages. This ranked list of pages then can be used to recommend potentially useful content to the user.

6. EXPERIMENTAL RESULTS

The recommendation framework has a number of components. We used different measures to evaluate each component. The goodness of the model built using logistic regression is tested using NagelKerke-$R^2$. The accuracy of the classifier that uses the model is tested using measures including precision, recall, accuracy, etc. Lastly we test the accuracy of the recommendation provided by the recommendation system that uses the classifier.

While evaluating, we use the n-fold cross validation technique [24]. We built four models with different predictor variables: $S$, $PS$, $LPS$, and $MLPS$. We compare the performance of text-based models with models that include DSN-derived information. Of the four models we tested, the first two depend on text similarity, and the last two include DSN-derived information with text similarity. While the first two models are similar to content-based recommendation, the last two models represent the hybrid approach.

The $S$ model only contains the similarity variable in the model equation. The $PS$ model contains both $S$ and the longest common subsequence ($LCP$). The $LPS$ model contains the DSN-dependent parameter $l$ along with the previous parameters. Note that DSNs usually deduce connection between a small fraction of users. Many users remain disconnected from the DSN depending on the connection threshold. Information about these isolated users is included in the $LPS$ model. For example, if an isolated user viewed a page pair in a session, the value of $l$ for that particular page pair will include this information. The last model, $MLPS$, includes the DSN-based group parameter $m$ with the three other predictor variables. Note that since isolated users do not belong to any group, parameter $m$ does not include information about isolated users. However, since $l$ contains the session information for isolated users, both the $LPS$ and $MLPS$ models contain information about isolated users.

6.1 Model evaluation: goodness of fit

We began our evaluation with testing the model that was built using logistic regression. Although logistic regression is somewhat similar to linear regression, classic regression analysis evaluation methods (e.g., variance, chi-square test) are not suitable when it comes to testing the model and the goodness of its fit. Since logistic regression uses a log function, the resultant model usually lacks a typical linear trend. Hence commonly used model testing measures such as $R^2$, variance, etc. are unable to properly detect the goodness of a fit. As a result, for models built using logistic regression, a number of different evaluation methods such as deviance, $\text{Pseudo} - R^2$, NagelKerke-$R^2$[25], etc. are used to measure the fit of the model. We use NagelKerke-$R^2$ to test the goodness of fit of a model built using logistic regression. The NagelKerke-$R^2$ equation is defined as:

$$f(x) = \frac{1 - \exp\left(-\frac{2(LL_{\text{fitted}} - LL_{null})}{N}\right)}{1 - \exp\left(-\frac{2LL_{null}}{N}\right)}$$

(9)

where $LL_{\text{fitted}}$ is the log likelihood of the fitted model, $LL_{null}$ is the log likelihood of the null model, and $N$ is the sample size. According to this measure, an outcome closer to one indicates a good fit.

Figure 7 (left) shows the NagelKerke-$R^2$ values for different models at different folds. These models are built using the AlgoViz Fall 2009 dataset. The X axis shows the number of folds and the Y axis shows the average NagelKerke-$R^2$ value. Each line in this plot represents a different model with different numbers of predictor variables. This plot shows that according to the NagelKerke-$R^2$ measure, the
complete model (MLPS), represented by the blue line, performs better compared to the other three models. The models with S, PS, and LPS have the worst performance, which is close to zero. All three models without the M variable have NagelKerke-$R^2$ values that are closer to zero, meaning that there is no significant improvement of the model when it is compared with the null model - the model without any predictor variable. A good model will result in a NagelKerke-$R^2$ value closer to 1. Out of the four models, only MLPS results in values much greater than zero (and close to 1). Thus it shows that models built with group information derived from the DSN perform better compared to content-based models in predicting the likelihood of viewing two pages together in a session.

Figure 7 (right) presents the NagelKerke-$R^2$ values for AlgoViz Spring 2010 data. This figure also indicates that the MLPS model performs better compared to the other three. Models S, PS, and LPS are all close to zero indicating that none of them can provide a good model.

As we see in these figures, the model with all four predictors performs better than the other models. While the inclusion of $l$ (one of the two DSN-dependent variables) does not provide significant improvement, including the other DSN-dependent variable, $m$, provides a better measure for the goodness of fit. From this analysis we can conclude that the group-specific variable $m$ is important in building a model following Equation 1.

### 6.2 Classifier accuracy

The resource pair proximity model provides a probability for a pair of pages to appear together in a session. This model thus can act as a classifier. Experimental results of the accuracy, precision, recall, and specificity show that the classifier with both parameters $m$ and $l$ performs better compared to the others. The accuracy of a classifier [26] is computed as:

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (10)$$

where Table 1 describes TP, TN, FP, and FN.

While accuracy shows the fraction of correct predictions, precision [26] shows the fraction of correct positive predictions. Precision is defined as:

$$\text{Precision} = \frac{TP}{TP + FP} \quad (11)$$

Two other widely used measures of classifier accuracy are recall and specificity. Recall or sensitivity [26, 27] provides the fraction of positive cases that are predicted as positive. It is defined as:

$$\text{Recall} = \frac{TP}{TP + FN} \quad (12)$$

Specificity [27] on the contrary provides the fraction of negative cases that are predicted negative and is defined as:

$$\text{Specificity} = \frac{TN}{TN + FP} \quad (13)$$

Figure 8 and Figure 9 show the experimental results for the two DSNs. Figure 8 (top left) shows the accuracy of the classifier for the AlgoViz Fall 2009 dataset. As we see when only similarity (i.e., cosine similarity in this instance) between a pair of pages is used to create the model (S), the average accuracy is around 60%. The accuracy of the model with PS is also similar to S, hence the lines overlap. Model LPS (the green line) shows an accuracy slightly over 70%. Considering longest common prefix ($P$) and $l$ along with the similarity increases the accuracy to around 70-80%. Again, $l$ is a global indicator of how likely two pages are to appear together in a session. Once we include the group parameter $m$ into the model, the average accuracy jumps to around 99%.

We also check the precision of the classifier (see Equation 11) which is shown in Figure 8 (top right). Here again the MLPS model (the blue line) exhibits the highest precision (around 100%) compared to the other models. LPS (the green line) shows better precision compared to S and PS.
Both $S$ and $PS$ models have approximately the same precision which is 60%. Both accuracy and precision for all of the models remain fairly similar for different folds. This indicates that the amount of data used to train and build a model has small impact beyond 6 folds.

Figure 8 (bottom left) shows the recall for the classifier. Recall is computed based on Equation 12. All of the models except $LPS$ show a recall of 100% and overlap with each other. Only $LPS$ shows varying recall, which ranges from 87% to 98%. Lastly, Figure 8 (bottom right) shows the specificity of the classifier. Specificity indicates the fraction of negative prediction for true negative cases (see Equation 13). In this plot, the $S$ and $PS$ models overlap at zero. The model with $LPS$ generates values between 30% to 50%, showing an average performance. The best performance is achieved by the $MLPS$ model with a specificity of 100%.

We also conducted these experiments with AlgoViz Spring 2010 data as seen in Figure 9. The accuracy for this dataset is given in Figure 9 (top left). Models $S$ and $PS$ both show an average accuracy of 52% for all the folds. The average accuracy of model $LPS$ ranges from 74% to 86%. The highest average accuracy of $LPS$ is 86% which is achieved at Fold 7 and the lowest average accuracy is 76% with Fold 10. Model $MLPS$ gives the best accuracy of all the models, which is around 99% for any of the folds.

Figure 9 (top right) depicts the precision of the classifier. This plot shows a trend similar to the accuracy plot. When we start with only text similarity in the model the average accuracy is 52%. This increases as we add $l$, $p$, and $m$ into the model. Model $LPS$ shows an average precision between 72% and 84% while model $MLPS$ shows an average precision close to 100%.

Figure 9 (bottom left) shows the recall of the classifier. According to this measure, models $S$ and $PS$ achieve the highest recall which is close to 100%. Next best recall is achieved by model $MLPS$ which is around 90%. Model $LPS$ exhibits the worst performance among all the models by producing a minimum recall of 90% with Fold 6 and maximum recall of 94% with Fold 10. Figure 9 (bottom right) shows the specificity of the models. Here the $MLPS$ model outperforms the other models. Models $S$ and $PS$ both show a specificity of zero. Model $LPS$ exhibits a specificity between 60% and 70%, showing a better performance than $S$ and $PS$.

As we see from these two datasets, for the majority of cases, the classifier that has the best performance is the $MLPS$ model. The $MLPS$ model based classifier has an accuracy of 99% to 100% on average. The model also has around 99% precision in most cases. The average recall and specificity of this model is close to 100%. Compared to $MLPS$, the performance of $LPS$ is low for all four of the evaluation measures. Models $S$ and $PS$ show similar performance, indicating that the inclusion of URLs in the model may not improve performance significantly.

One interesting point to note here is that although in most cases models $S$ and $PS$ perform poorly, they do exhibit very high recall, close to 100%. This is because these models produce an outcome of one for most cases. While training and testing the classifier we attempted to use similar numbers of positive and negative examples. Thus the models rightly predict the true outcomes in half the cases. Since recall does not consider false positives (see Equation 12) these models show a better recall performance. They do however show a poor performance in specificity — a measure that considers the false positives (see Equation 13).

We also test the performance of the classifiers using the F1 score, which depends on precision and recall [28]. The value of F1 score ranges from 0 to 1 where 1 shows the best performance and 0 indicates worst performance. F1 score is computed as:

$$F1 = \frac{2TP}{2TP + FP + FN}$$  \hspace{1cm} (14)

Figure 10 shows the F1 score for the Fall 2009 and Spring 2010 AlgoViz DSNs. The X axis shows the number of folds while the Y axis lists the average F1 score for that fold. In the Fall 2009 (Figure 10 (left)) the lowest F1 score is achieved by the $S$ and $PS$ models which is 0.74. The $LPS$ model shows improved F1 score (close to 0.8). The highest F1 score, which is 1, is achieved by the $MLPS$ model. Similar trends are visible in the Spring 2010 DSN. Models $S$ and $PS$ exhibit similar F1 scores (approximately 0.74) while the $LPS$ model shows improved performance — ranging from 0.82 to 0.85. The best F1 score is generated by the $MLPS$ model. At 6-fold the F1 score for the $MLPS$ model is 0.98.
which increases to 0.99 for the subsequent folds. For all the DSNs, models $S$ and $PS$ have the worst F1 score and $MLPS$ shows the best score. $LPS$ performs in between these three models. Also, all models except $LPS$ show a steady F1 score through all the folds. With higher number of folds, $LPS$ shows decreasing F1 score. This indicates that even with more data the $LPS$ fails to model user behavior successfully.

6.3 Recommendation performance

The classifiers described in the previous subsection are later used to recommend content. The approach towards recommending resources based on DSN is described in Figure 6. We follow a similar approach while evaluating the recommendation framework.

For any given user $u_i$ in the test fold, we have a list of pages that the user saw in a session. Based on the pages viewed by the user we assign him to a group. While building the DSN a number of users were not assigned to any groups for various reasons (e.g., did not view $x$ similar pages like any other user, did not view $y$ pages in a session). At this step we assign these users, who appear in the test fold, to a particular group based on their pageviews. The centroid method (see Section 5) is used at this step.

Once a user is assigned to a group, we select one page $ps_i$ from his session $ps_1, ps_2, \ldots, ps_n$ and one page $pt_{j}$ from the set of pages used in the training phase, to build a test page pair, $(ps_i, pt_{j})$. This test page pair then passes through the resource pair proximity model which provides the probability of these two pages being viewed in a session. The estimated score is used to rank all the pages. The page pairs containing only the pages that appear in the session of user $u_i$ are called user page pairs. We then compute the percentile of the user page pairs in the ranked list. The average percentile for all the user page pairs is reported for each fold.

We claim that a good recommendation places the user page pairs at the top of the ranking, at a higher percentile.

Figure 11 (left) shows the performance of the recommendation for the AlgoViz Fall 2009 dataset. When we use only similarity between the page titles to build the model, the recommendation framework does not perform well. The average percentile for user page pairs with model $S$ is zero for all the folds. A slightly better but varying performance is seen by the next model, $PS$. According to this model, the average percentile for user page pairs is either zero or close to zero at Fold 6, Fold 7, and Fold 9. Folds 8 and 10 place those page pairs in around the 10th percentile. This means that the page pairs seen by the user appear at the bottom of the ranked list. Model $LPS$ shows a varying but better performance than $PS$. At Fold 6 it performs poorly where the user page pair appears in the 10th percentile. At Fold 7 the user page pairs appear in the 35th percentile. For the rest of the folds the percentiles remain close to 25 for this model. With this dataset, $MLPS$ achieved the best performance by placing the user page pairs in the 99th percentile.

A similar trend is seen in the AlgoViz Spring 2010 dataset in Figure 11 (right). Model $S$ performs poorly by placing the user page pairs low in the ranked list for all the folds. Model $PS$ improves the recommendation performance slightly, which places the user page pairs in the 5th percentile. The performance of model $LPS$ peaks at Fold 6 with user page pairs appearing in the 80th percentile. However, as the number of folds increases the model starts to show decreasing performance. Model $MLPS$ shows a steady and improved performance by placing the user page pairs at around the 80th percentile for each fold.

For both datasets, model $MLPS$ performed better than the other models. While this model achieved a good recommendation performance for the Fall 2009 dataset (around 99th percentile) it showed a decreased performance for the Spring 2010 dataset (around 80th percentile). However, in both cases, it outperforms the other three models. Having more isolated users in the Spring 2010 DSN could be one of the reasons for the decreased performance. Another reason could be the large number of smaller groups. As we see in Figure 4, Spring 2010 has 12 groups of users. Many of these groups have two to six users. Having large numbers of groups with few users increases the risk of inaccurate group assignment, eventually leading to recommendation of non-relevant content. One way to mitigate this problem is to merge smaller groups until they meet a certain user threshold. Another way could be to depend on the topic of the groups and to merge the groups as long as the topic distance is smaller than a certain distance.

7. DISCUSSION

Based on the research discussed above, and some related work with the Ensemble digital library for computing educational resources [29], we can conclude that effective resource recommendation services can be added to edu-DLs, that make use of logged data from the sessions of anonymous users. This is made possible by building a deduced social network based on common page visits, identifying groups by modularity clustering, and using logistic regression to predict if an unseen resource page will be viewed by a user. Our approach to training, including a tailored generator for negative training examples, and our $MLPS$ resource pair proximity model for prediction, shows promise when the model is evaluated as to goodness of fit, classifier accuracy, and recommendation performance.

However, DSN-based recommendation can become computationally expensive for any digital libraries with heavy traffic. One way to mitigate this problem is to break down the log data in smaller segments (e.g., weeks instead of months). Another way is to change the connection threshold $k$. Yet another approach is to build the models periodically instead of dynamically generating them. Quality of log data is another critical aspect of DSN-based recommendation. Our approach depends heavily on session information. The performance of DSN-based recommendation would degrade if the sessions have fewer pages in them. One option could be to expand the sessions artificially by including sim-
ilar resources. Smaller sessions also can produce large numbers of isolated users resulting in failure to detect useful user groups. Future work will focus on scalability, deployment in the AlgoViz and Ensemble digital libraries, and consideration of other types of DSNs.

8. ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant Numbers DUE-0836940, DUE-0937863, and DUE-0840719.

9. REFERENCES


