Announcements Recommendation System
Arlo White
awhite@calpoly.edu
Annie Beug
asbeug@calpoly.edu
Department of Computer Science, California Polytechnic State University

Abstract
This project investigates possible solutions for the recommendations component of a web-based announcements system at Cal Poly. The aim is to consider how rich user and announcement metadata can be used to enhance recommendations and alleviate the cold-start problem. The large number of user and announcement attributes and quantity of possible attribute values poses significant challenges to machine learning systems. This project considers various approaches and evaluates their applicability to the problem and highlights implementation concerns and limitations.

Introduction
Cal Poly ITS staff will soon be developing a new announcements system for the campus community. This system will include rich announcement metadata and have access to a wealth of information about system users. Eventually this system may contain hundreds if not thousands of announcements that will be of interest to different users. The problem of automatically determining which items are relevant to specific users is known as a recommender problem. The general data used by the system will be user attributes, announcement item attributes, and user-generated item feedback, described in fuller detail in the Data Set section.

The aim of our project was to experiment with using different classifiers in a recommender system. The recommender system is architected as a black box into which the announcement system will pass a user, expecting back an ordered subset of announcements. Some announcements may be required to be displayed to all users; others may be filtered to be shown only to specific users, while the rest will be displayed based on the results of the recommender system. All announcements will be time-sensitive, having a time frame in which they are considered candidates for display.
The entire set of announcements cannot usably be displayed at one time; therefore, a mechanism is needed to suitably determine which announcements should be displayed to a particular user and in what order. We chose to investigate using classifiers to build a recommender system through a set of classifiers. We used the Weka machine learning API to test as many classifiers as we could. Specifics on which classification algorithms we looked at are detailed in the Experiments section below.

**Background / Related Work**

Recommender systems predict the level of interest a user will have for an item based on previous feedback data. There are two general approaches to recommender systems: collaborative filtering and content-based filtering. Collaborative filtering operates by discovering similarities between user feedback patterns and is generally more accurate than content-based filtering; however, it suffers from the cold-start problem. Content-based filtering uses the item attributes thus can overcome the cold-start problem caused by items lacking feedback data. [12][14]

Collaborative filtering is based on the assumption that users that have had similar feedback in the past will have similar feedback in the future. Feedback can be predicted for a given user by evaluating the similarity of the given user’s past feedback to the feedback of other users. For example, if users $U_1$, $U_2$, and $U_3$ all like announcement $A_1$, and users $U_1$ and $U_2$ both like announcement $A_2$, it may be predicted that user $U_3$ will like $A_2$ as well. [2][3]

Content-based filtering uses information about the items being recommended. One option is to use the explicit content of the item, which in the case of an announcement would be the text of the announcement itself. A second option is to use metadata about the item, such as the announcement provider or tags. The system then recommends items to a given user determined to be similar to other items they have liked in the past. [12][14][15]

Content-based filtering in many cases may not produce as accurate results as collaborative filtering because the associations built between items may be too shallow. As an example, suppose a user likes *Snow Crash* and *Cryptonomicon* by Neil Stephenson, but does not like the book *Diamond Age* by the same author but has not provided that feedback to the system. A content-based filtering approach may recognize
that the first two books are by the same author and recommend *Diamond Age* to the user. A collaborative filtering approach, on the other hand, may see that multiple users have the same feedback pattern of liking *Snow Crash* and *Cryptonomicon*, but not liking the third novel. Using this information, the system can refrain from recommending *Diamond Age* to the user. Unfortunately, collaborative filtering does not work when users or items have sparse or absent feedback data, this issue is referred to as the cold start problem.

**Solution**

Our cold medication is content-based filtering techniques. Such solutions can supplement the recommendations provided by the traditional recommender system. When a new announcement is posted, the system will lack feedback data to use in recommending the announcement to users based on collaborative filtering techniques. In addition to new announcements lacking any feedback, many users may not provide feedback. Collaborative filtering techniques will not be able to generate feedback predictions for users who do not have feedback patterns. Content-based filtering relying solely on announcements will also fail to generate feedback predictions, as there is no feedback linking users to announcements. Since the user has not rated any items, the system cannot find items similar to the ones they have rated in the past.

However, in our problem we already have attributes about the user and the announcement. Using both sets of attributes in combination can help to alleviate the cold start problem. With the user attributes, we find similar users who have provided feedback for users who have not. Then, based on the feedback the similar users have provided, we can generate recommendations for a cold start user.

**Data Set**

Data can be separated into three different sets: information about users, information about announcements, and user-provided feedback about announcements. The users in our experimental data were pulled from a development data set, but are representative of actual users in the production portal environment. We chose not to use sensitive data, such as GPA, date of birth, social security number, and race. We did this to simplify the project, as well as because the attributes we chose were from the generic view of user data exposed to the portal.
Not all of these attributes may be relevant. Many machine-learning methods suffer under an increased number of attributes, so we wanted to only select those that seem relevant to the problem. For example, major is relevant since we would expect the model to discover rules such as Mechanical Engineers like announcements tagged with robotics. Another user attribute we included was gender because we would expect correlations to announcements about events such as the Vagina Monologues or the Society of Women Engineers.

Item attributes will be static information provided at item creation. User attributes will be a sub-set of semi-static portal user data. Some attributes, such as primary person type, major or class year will change, but seldom. Feedback is binary: like versus dislike.

We chose to model only explicit feedback to minimize the scope our project. Use of implicit feedback is discussed in the Future Work section below. We chose to use binary classes since many of the classifier implementations we were using could only deal with such. We did choose to filter classifiers by what could deal with missing values for attributes (symbolized in the schema in Table 1 by a question mark: "?"). Students will have attributes for degree, major, college and class level. Faculty and staff will have attributes for department and subdivision. Some users who are, for instance, both a student as well as a staff person will have values for degree, major, and college as well for department and subdivision.

Users
- Gender (M, F, ?)
- Primary person type (ASI, Student, CPC, Faculty, Staff)
- Degree (Bachelors, Masters, MBA, Doctorate, ?)
- Major (CSC, MIS, ENGL, etc. and ?)
- College (CENG, CAED, etc. and ?)
- Class Level (Freshman, Sophomore, Junior, Senior, Post-Bacc, Graduate, ?)
- Department (Computer Science, Vending, Information Management, etc. and ?)
- Subdivision (University Advancement, Academic Affairs, etc. and ?)

Announcements
- Type (Announcement, Event)
- Provider (Ex.: computer science department, PAC, Rec Center)
- Tags (Ex.: soccer, play, womens')

Feedback
- Rating (Like, Dislike)

Figure 1: Data set definition

Testing Data Generation

In order to emulate feedback data, a system to generate such data was developed. The generation model is based on a set of rules. Each rule has criteria to match a given user
and announcement pair based on the attributes of each. Given a match, a rule has a percent chance to be used and a percent chance to like the announcement. The rules are then combined in the fashion illustrated in Figure 2. This gives a somewhat realistic and random data set with appropriate biases that we can test. After generating test data with the rules generator, the distributions were verified by statistical analysis.

```plaintext
// U is a list of users from the data set
// A is a list of announcements from the data set
// R is a list of rules determining the feedback probabilities
// that a given user will rate and like the given announcement;
// chanceToRate is the probability that the user will rate the
// announcement as either like or dislike; chanceToLike is the
// probability that given a rating, it is like
Algorithm rules-generator(R, U, A)
For Users in U
  For Announcements in A
    For Rules in R
      If Rule matches Announcement and User
        If randNum <= R.chanceToRate
          If R.chanceToLike >= 1.0
            feedback = LIKE
          Else if R.chanceToLike <= 0.0
            Feedback = DISLIKE
        Else
          If randNum2 <= R.chanceToLike
            Feedback = LIKE
          Else
            Feedback = DISLIKE

Figure 2: Pseudo code for generating rule-based feedback

Data Transformations

Transforming the problem into a classification problem is fairly straightforward. The user and announcement attributes are simply combined into a single data instance and the class label represents whether the user feedback.

One limitation of classifiers is that many algorithms require that the attributes be nominal. This means that the data needs to be scanned to determine the nominal values for every attribute. This also prevents the class label from being a numeric value. One hope was to express different degrees of interest. For example, if a user clicked an announcement and viewed it, this might be considered a small degree of interest. However, expressing this as another class label confuses the problem. Classifiers consider class labels independently and not as a continuum such as hate, dislike, neutral, and like.

Another challenge in forming the classification problem is attributes which may have multiple values per record. For example, an announcement may have multiple tags. However, we cannot simply add more attributes, i.e. TAG1, TAG2, TAG3, as classifiers
consider each attribute separate and independent, and we want correlations to a single set of attribute values. Instead each data instance with multi-value attributes is split into multiple instances.

A user that likes an announcement with two tags will be represented in the classification problem as that user liking an announcement with “TAG1” and a separate instance of the same announcement with liking “TAG2”. Of course this has a multiplicative effect. A user that likes an announcement with two tags will be split into two rows.

Another requirement is that we can obtain a score from our model given the degree of liking. Many classifiers reveal a probability distribution that gives the chance a user will like or dislike an announcement. This can be used to generate the score.

One consideration is in the reversing of the multi-attribute split. When requesting a score for an announcement with multiple tags you will need to request a score with the announcement for each tag.

Another consideration is that there is a “theoretical” maximum performance that a classifier should have. For example, if we have a single rule that 80% of Mechanical Engineers like Robotics announcements, then our generated feedback data will only have Mechanical Engineers and Robotics announcements with 80% of the class labels being liked. Any classifier should theoretically only be able to “achieve” an accuracy of 80% due to the random nature of the data.

Another issue is that some users may defy their attributes. A male user may be a strong feminist and be very interested in seeing the Vagina Monologues, yet the content-based
system may not recommend the announcement because most men are not interested in announcements with the same tag. One approach is to simply fall back to a traditional recommender system if a user has enough feedback data for collaborative filtering.

The challenge is to create a recommender system that uses user and announcement attributes as part of its recommendations in addition to computing similarity between users, without the attribute-based recommendations overshadowing feedback-based recommendations. For example, given a female Mechanical Engineer and using something similar to the C4.5 classification algorithm, it may be that gender is chosen as the first differentiating attribute and the major (Mechanical Engineering) is the second most influential attribute for the item attribute we are looking at. We do not want the item to be ranked low purely on the basis of gender; the fact that the student is a Mechanical Engineer should have a stronger influence on the ranking than gender.

By taking into account not only the attributes of the users and announcements as well as feedback from users on displayed announcements we hope to create a more robust and accurate recommendation system.

**Experiments**

In our experiments with classifier algorithms we attempted to generate data with a similar complexity to the data we anticipate to see in the future application.

We selected classifiers available in the Weka project that are capable of processing our data and that provide a probability distribution for the class label predictions. Using our feedback rules generation system, we created multiple training sets and ran these classifiers on each of the training sets against the testing data listed below.

```
M, STUDENT, BACHELORS, KINE, CSM, JUNIOR, ?, ?, EVENT, Athletics, basketball
F, STUDENT, BACHELORS, KINE, CSM, JUNIOR, ?, ?, EVENT, Athletics, basketball
M, STUDENT, BACHELORS, KINE, CSM, JUNIOR, ?, ?, EVENT, Athletics, football
F, STUDENT, BACHELORS, KINE, CSM, JUNIOR, ?, ?, EVENT, Athletics, football
M, STUDENT, BACHELORS, KINE, CSM, JUNIOR, ?, ?, EVENT, Athletics, soccer
F, STUDENT, BACHELORS, KINE, CSM, JUNIOR, ?, ?, EVENT, Athletics, soccer
M, STUDENT, BACHELORS, BUS, COB, JUNIOR, ?, ?, EVENT, Athletics, basketball
F, STUDENT, BACHELORS, BUS, COB, JUNIOR, ?, ?, EVENT, Athletics, basketball
M, STUDENT, BACHELORS, BUS, COB, JUNIOR, ?, ?, EVENT, Athletics, football
F, STUDENT, BACHELORS, BUS, COB, JUNIOR, ?, ?, EVENT, Athletics, football
M, STUDENT, BACHELORS, BUS, COB, JUNIOR, ?, ?, EVENT, Athletics, soccer
F, STUDENT, BACHELORS, BUS, COB, JUNIOR, ?, ?, EVENT, Athletics, soccer
M, STUDENT, BACHELORS, MU, CENG, JUNIOR, ?, ?, EVENT, Athletics, basketball
F, STUDENT, BACHELORS, MU, CENG, JUNIOR, ?, ?, EVENT, Athletics, basketball
M, STUDENT, BACHELORS, MU, CENG, JUNIOR, ?, ?, EVENT, Athletics, football
F, STUDENT, BACHELORS, MU, CENG, JUNIOR, ?, ?, EVENT, Athletics, football
M, STUDENT, BACHELORS, MU, CENG, JUNIOR, ?, ?, EVENT, Athletics, soccer
```
Figure 4: Test data set

All attribute values are the same except for gender, major, and announcement tag. Our different training sets are designed to create variations related to these attributes in order to reveal differences in classifier behaviors.

The chance for each testing instance to be liked was extracted from each classifier using the probability distribution provided by the classifier. These values are output to a table for each testing set. The first column contains the classifier name and the second contains the training runtime in milliseconds. The remaining columns correspond to the testing data instances above.

A simple Python script was developed to verify statistics about the training set distribution. This script was used to verify the output of our feedback generation system and assure a good data distribution. In each table, the "actual" row denotes the actual data distribution measured by this script. The "theoretical" row contains the theoretical values based on the model that generated the training data.

Our training data is based on a statistically realistic set of user data from the Cal Poly campus. These users are combined with example announcements using the feedback generation system described earlier. The feedback rules we use take a percentage of the user data using a random selection process. This results in poorly distributed attribute values for training instances with few rule matches. This is the reason we contrast business (BUS) and music (MU) majors. Business majors consist of a large percentage of the campus population while there are only a handful of music majors. We were curious to see how this would play out in our experiments.

Tests

Test 1

Our first test establishes the background noise against which our later test sets will contrast. Theoretically, we expect each of the values to be 0.50.

Though our feedback generation model creates data with a 50% chance to like overall, narrow slices will stray from this value. For example, there are only 4 training instances of male, kinesiology (KINE) majors who rated football. The random generator resulted
with 3 liking and 1 disliking, resulting in the 0.75 actual values in column 3. Despite the fact that we have 29094 training instances, the number of attributes and quantity of attribute values inevitably results in these anomalies.

This first experiment of random data shows how hard some classifiers try to overfit the data. LBR, Kstar, DTNB, DecisionTable, and REPTree have high values in their rows that suggest they tried to develop correlations that do not really exist.

<table>
<thead>
<tr>
<th></th>
<th>KINE</th>
<th>KINE</th>
<th>KINE</th>
<th>KINE</th>
<th>KINE</th>
<th>BUS</th>
<th>BUS</th>
<th>BUS</th>
<th>BUS</th>
<th>BUS</th>
<th>BUS</th>
<th>MU</th>
<th>MU</th>
<th>MU</th>
<th>MU</th>
<th>MU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Theoretical</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
</tr>
<tr>
<td>Actual</td>
<td>0.50</td>
<td>0.48</td>
<td>0.44</td>
<td>0.44</td>
<td>0.60</td>
<td>0.59</td>
<td>0.48</td>
<td>0.48</td>
<td>0.61</td>
<td>0.47</td>
<td>0.56</td>
<td>0.54</td>
<td>1.00</td>
<td>0.68</td>
<td>0.50</td>
<td>0.50</td>
</tr>
<tr>
<td>bayes.AODE</td>
<td>88</td>
<td>0.51</td>
<td>0.55</td>
<td>0.51</td>
<td>0.46</td>
<td>0.50</td>
<td>0.55</td>
<td>0.54</td>
<td>0.56</td>
<td>0.56</td>
<td>0.54</td>
<td>0.48</td>
<td>0.45</td>
<td>0.46</td>
<td>0.46</td>
<td></td>
</tr>
<tr>
<td>bayes.AODEsr</td>
<td>116</td>
<td>0.52</td>
<td>0.52</td>
<td>0.52</td>
<td>0.52</td>
<td>0.52</td>
<td>0.52</td>
<td>0.48</td>
<td>0.48</td>
<td>0.51</td>
<td>0.51</td>
<td>0.50</td>
<td>0.49</td>
<td>0.43</td>
<td>0.45</td>
<td>0.46</td>
</tr>
<tr>
<td>bayes.NaiveBayes</td>
<td>45</td>
<td>0.53</td>
<td>0.53</td>
<td>0.53</td>
<td>0.53</td>
<td>0.53</td>
<td>0.53</td>
<td>0.48</td>
<td>0.48</td>
<td>0.51</td>
<td>0.51</td>
<td>0.50</td>
<td>0.49</td>
<td>0.43</td>
<td>0.45</td>
<td>0.46</td>
</tr>
<tr>
<td>bayes.BayesNet</td>
<td>32</td>
<td>0.52</td>
<td>0.52</td>
<td>0.52</td>
<td>0.52</td>
<td>0.52</td>
<td>0.52</td>
<td>0.48</td>
<td>0.48</td>
<td>0.51</td>
<td>0.51</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
</tr>
<tr>
<td>Functions.Logistic</td>
<td>4352</td>
<td>0.51</td>
<td>0.51</td>
<td>0.51</td>
<td>0.51</td>
<td>0.51</td>
<td>0.51</td>
<td>0.48</td>
<td>0.48</td>
<td>0.51</td>
<td>0.51</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
</tr>
<tr>
<td>lazy.LBR</td>
<td>101</td>
<td>0.51</td>
<td>0.51</td>
<td>0.51</td>
<td>0.51</td>
<td>0.51</td>
<td>0.51</td>
<td>0.48</td>
<td>0.48</td>
<td>0.51</td>
<td>0.51</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
</tr>
<tr>
<td>lazy.KStar</td>
<td>6</td>
<td>0.51</td>
<td>0.51</td>
<td>0.51</td>
<td>0.51</td>
<td>0.51</td>
<td>0.51</td>
<td>0.48</td>
<td>0.48</td>
<td>0.51</td>
<td>0.51</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
</tr>
<tr>
<td>rules.JRip</td>
<td>48825</td>
<td>0.52</td>
<td>0.52</td>
<td>0.52</td>
<td>0.52</td>
<td>0.52</td>
<td>0.52</td>
<td>0.48</td>
<td>0.48</td>
<td>0.51</td>
<td>0.51</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
</tr>
<tr>
<td>rules.ConjRule</td>
<td>411</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.48</td>
<td>0.48</td>
<td>0.51</td>
<td>0.51</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
</tr>
<tr>
<td>rules.DTNB</td>
<td>22856</td>
<td>0.51</td>
<td>0.51</td>
<td>0.51</td>
<td>0.51</td>
<td>0.51</td>
<td>0.51</td>
<td>0.48</td>
<td>0.48</td>
<td>0.51</td>
<td>0.51</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
</tr>
<tr>
<td>rules.DecisionTable</td>
<td>7045</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.48</td>
<td>0.48</td>
<td>0.51</td>
<td>0.51</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
</tr>
<tr>
<td>rules.ZeroR</td>
<td>27</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.48</td>
<td>0.48</td>
<td>0.51</td>
<td>0.51</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
</tr>
<tr>
<td>rules.PART</td>
<td>120774</td>
<td>0.52</td>
<td>0.52</td>
<td>0.52</td>
<td>0.52</td>
<td>0.52</td>
<td>0.52</td>
<td>0.48</td>
<td>0.48</td>
<td>0.51</td>
<td>0.51</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
</tr>
<tr>
<td>trees.J48</td>
<td>5003</td>
<td>0.54</td>
<td>0.54</td>
<td>0.54</td>
<td>0.54</td>
<td>0.54</td>
<td>0.54</td>
<td>0.48</td>
<td>0.48</td>
<td>0.51</td>
<td>0.51</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
</tr>
<tr>
<td>trees.ADTree</td>
<td>1212</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.48</td>
<td>0.48</td>
<td>0.51</td>
<td>0.51</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
</tr>
<tr>
<td>trees.REPTree</td>
<td>1211</td>
<td>0.76</td>
<td>0.74</td>
<td>1.00</td>
<td>1.00</td>
<td>0.62</td>
<td>0.41</td>
<td>0.44</td>
<td>0.76</td>
<td>0.59</td>
<td>0.45</td>
<td>0.48</td>
<td>1.00</td>
<td>1.00</td>
<td>0.75</td>
<td>0.75</td>
</tr>
</tbody>
</table>

Table 1: Test 1 results

Test 2

In the second test set, we add the rule that all KINE majors like the events from the Athletics provider. We expect all of the KINE columns to be close to 1.0 and the rest to be 0.50. The actual values match this for KINE and BUS majors, but as before, the low number of MU majors result in spiky data.

Most of the algorithms did fairly well, recognizing the KINE rule while not giving false predictions for the other majors.

LBR, VFI, DTNB, DecisionTable, ZeroR, PART did particularly poor with strange variations in the values for BUS and MU majors.
Some algorithms maintain a high value for the KINE prediction, such as AODE, Kstar, and ADTree, while predicting around 0.50 for the other majors. Other algorithms show a weaker prediction for the KINE major.

<table>
<thead>
<tr>
<th></th>
<th>BB</th>
<th>BUS</th>
<th>M</th>
<th>MU</th>
<th>M</th>
<th>F</th>
<th>M</th>
<th>F</th>
<th>M</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Theoretical</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1.00</td>
<td>1.00</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
</tr>
<tr>
<td>Actual</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1.00</td>
<td>1.00</td>
<td>0.44</td>
<td>0.49</td>
<td>0.46</td>
<td>0.47</td>
<td>0.50</td>
</tr>
<tr>
<td>bayes.AODE</td>
<td>50</td>
<td>0.96</td>
<td>0.95</td>
<td>0.92</td>
<td>0.98</td>
<td>0.50</td>
<td>0.49</td>
<td>0.48</td>
<td>0.50</td>
<td>0.55</td>
</tr>
<tr>
<td>bayes.AODEsr</td>
<td>46</td>
<td>0.82</td>
<td>0.82</td>
<td>0.84</td>
<td>0.82</td>
<td>0.51</td>
<td>0.49</td>
<td>0.48</td>
<td>0.55</td>
<td>0.55</td>
</tr>
<tr>
<td>bayes.NaiveBayes</td>
<td>46</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.47</td>
<td>0.46</td>
<td>0.46</td>
<td>0.46</td>
<td>0.46</td>
</tr>
<tr>
<td>bayes.BayesNet</td>
<td>168</td>
<td>0.71</td>
<td>0.71</td>
<td>0.71</td>
<td>0.70</td>
<td>0.47</td>
<td>0.46</td>
<td>0.46</td>
<td>0.46</td>
<td>0.46</td>
</tr>
<tr>
<td>functions.Logistic</td>
<td>4682</td>
<td>0.70</td>
<td>0.69</td>
<td>0.69</td>
<td>0.69</td>
<td>0.50</td>
<td>0.49</td>
<td>0.47</td>
<td>0.47</td>
<td>0.47</td>
</tr>
<tr>
<td>lazy.LBR</td>
<td>97</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>0.55</td>
<td>0.86</td>
<td>0.75</td>
<td>0.50</td>
<td>0.25</td>
</tr>
<tr>
<td>lazy.KStar</td>
<td>5</td>
<td>0.98</td>
<td>0.98</td>
<td>0.98 0.98</td>
<td>0.50</td>
<td>0.44</td>
<td>0.42</td>
<td>0.39</td>
<td>0.50</td>
<td>0.50</td>
</tr>
<tr>
<td>Misc.VFI</td>
<td>46</td>
<td>0.53</td>
<td>0.52</td>
<td>0.52</td>
<td>0.52</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
</tr>
<tr>
<td>rules.JRip</td>
<td>46460</td>
<td>0.70</td>
<td>0.69</td>
<td>0.70</td>
<td>0.70</td>
<td>0.47</td>
<td>0.46</td>
<td>0.46</td>
<td>0.46</td>
<td>0.46</td>
</tr>
<tr>
<td>rules.ConjRule</td>
<td>300</td>
<td>0.69</td>
<td>0.69</td>
<td>0.69</td>
<td>0.69</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
</tr>
<tr>
<td>rules.DTNB</td>
<td>6706</td>
<td>0.97</td>
<td>0.97</td>
<td>0.97</td>
<td>0.97</td>
<td>0.54</td>
<td>0.54</td>
<td>0.54</td>
<td>0.54</td>
<td>0.54</td>
</tr>
<tr>
<td>rules.DecisionTable</td>
<td>7060</td>
<td>0.97</td>
<td>0.97</td>
<td>0.97</td>
<td>0.97</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
</tr>
<tr>
<td>rules.Sarah</td>
<td>14</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
</tr>
<tr>
<td>rules.PART</td>
<td>32551</td>
<td>0.69</td>
<td>0.68</td>
<td>0.68</td>
<td>0.68</td>
<td>0.49</td>
<td>0.48</td>
<td>0.48</td>
<td>0.48</td>
<td>0.48</td>
</tr>
<tr>
<td>trees.J48</td>
<td>5362</td>
<td>0.70</td>
<td>0.70</td>
<td>0.70</td>
<td>0.70</td>
<td>0.54</td>
<td>0.54</td>
<td>0.54</td>
<td>0.54</td>
<td>0.54</td>
</tr>
<tr>
<td>trees.ADTree</td>
<td>567</td>
<td>0.96</td>
<td>0.96</td>
<td>0.96</td>
<td>0.96</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
</tr>
<tr>
<td>trees.REPTree</td>
<td>1351</td>
<td>0.90</td>
<td>0.90</td>
<td>0.90</td>
<td>0.90</td>
<td>0.47</td>
<td>0.47</td>
<td>0.47</td>
<td>0.47</td>
<td>0.47</td>
</tr>
</tbody>
</table>

Table: Test 2 results

**Test 3**

In the third experiment, we add some the rules that MU majors like concerts and dislike robotics. This should keep their overall chance to like at 0.5 but create some spikes in the data. We want to verify that these spikes toward unrelated attributes (non-Athletic announcements) do not result in changes to the MU majors.

While the Bayesian algorithms did well in the previous test, they now create false positives and abnormally high predictions for the MU majors. The Logistic algorithm also suffers from this.

Strangely, LBR does a little better here as does DecisionTable and DTNB. REPTree had a strange drop in the two rightmost cells.
Test 4

This experiment adds the rules that men like football and women like soccer while maintaining that all KINE majors like all Athletics events. The aim is to see how much this affects the KINE scores or BUS & MU basketball scores.

Our verification script revealed a few anomalies in the generated training set. While the bias of Men-Football, Female-Soccer bias is good compared to their standard Athletics statistics, there is an unwanted spike in MU majors liking announcements overall. So we cannot really trust the MU section.

Table 2: Test 3 results

<table>
<thead>
<tr>
<th>Method</th>
<th>Test 3 Results</th>
<th>Test 4 Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bayes.NaiveBayes</td>
<td>24</td>
<td>0.67</td>
</tr>
<tr>
<td>Bayes.BayesNet</td>
<td>112</td>
<td>0.67</td>
</tr>
<tr>
<td>Functions.Logistic</td>
<td>60542</td>
<td>0.68</td>
</tr>
<tr>
<td>Lazy.LBR</td>
<td>69</td>
<td>0.95</td>
</tr>
<tr>
<td>Lazy.KStar</td>
<td>6</td>
<td>0.99</td>
</tr>
<tr>
<td>Naiv.VFI</td>
<td>18</td>
<td>0.99</td>
</tr>
<tr>
<td>Rules.JRip</td>
<td>15093</td>
<td>0.54</td>
</tr>
<tr>
<td>Rules.ConjRule</td>
<td>334</td>
<td>0.51</td>
</tr>
<tr>
<td>Rules.DTNB</td>
<td>24069</td>
<td>0.96</td>
</tr>
<tr>
<td>Rules.DecisionTable</td>
<td>8823</td>
<td>0.95</td>
</tr>
<tr>
<td>Rules.Zar0R</td>
<td>9</td>
<td>0.51</td>
</tr>
<tr>
<td>Rules.PART</td>
<td>34372</td>
<td>0.69</td>
</tr>
<tr>
<td>Trees.J48</td>
<td>6025</td>
<td>0.00</td>
</tr>
<tr>
<td>Trees.ADTree</td>
<td>757</td>
<td>0.99</td>
</tr>
<tr>
<td>Trees.REPTree</td>
<td>1170</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Figure 5: Output from data distribution script

AODEsr was able to detect the trend a little in its BUS predictions. LBR also predicted the trend but decreased Females chance to like Football and Men's chance to like soccer. It also shifted the basketball scores toward men. Kstar seemed to well at predicting everything but F, Soc, BUS. DTNB and DecisionTable had limited success. REPTree did an extraordinary job picking out the rules in this test.
Table 4: Test 4 results

Test 5

The last four tests had a fairly brutal amount of background noise and attribute variety. We were curious if simplifying the background noise would result in some classifiers doing better. In this experiment, we remove the unrelated rules from Test 3 and only have the KINE-Athletics, Male-Football, Female-Soccer rules. The background noise consists only of KINE, MU, BUS majors and Athletics and just a few other tags.

With stronger correlations, AODEsr actually did a bit poorer, the strength of the male-female biases carried over into the KINE scores and the basketball scores decreased. LBR did fairly well for the KINE and BUS majors again but predicted 1.0 for all MU majors for some reason despite MU majors having a good data distribution according to our verification script. Kstar picked out the trends somewhat, but did not have a very strong Male-BUS-Football score. It is possible the male-female biases are not a strong enough proportion of the data.

Figure 6: Output from data distribution script
Conclusion

These experiments revealed some interesting information about the classifiers. Many of the classifiers failed to perform well in any of our tests: VFI, Jrip, ConjunctiveRule, ZeroR, PART, J48. It may be that the default Weka configuration for these classifiers needs to be tuned for our data or that their algorithms are just not suited to problems with so many attributes and attribute values.

The Bayesian classifiers do fairly well, and the AODE and AODEsr algorithms seem to have better precision than the others.

LBR, Kstar, and REPTree succeeded in finding some of the rules in Tests 4 and 5 but not all of them. Many of the other classifiers flat out failed to show significant understanding of biases we injected into the data.

In developing these experiments, we realized that classifiers have a few qualities that are not well suited to our problem. One of the major issues is that classifiers require a background of data against which to detect biases. If we simply had a bunch of data listing the instances that were liked, the classifier would short circuit and just guess that everyone likes everything. One concern is that in the actual application, users may not click dislike enough to create this background data against which to compare. The formation of the problem as a binary classification problem is somewhat flawed as well. We had hoped to express various degrees of interest, such as when a user views an announcement. It is not apparent how this can be done with classifiers requiring nominal class labels.

<table>
<thead>
<tr>
<th>bayes.NaiveBayes</th>
<th>67.67</th>
<th>0.67</th>
<th>0.82</th>
<th>0.79</th>
<th>0.81</th>
<th>0.50</th>
<th>0.50</th>
<th>0.69</th>
<th>0.71</th>
<th>0.65</th>
<th>0.68</th>
<th>0.55</th>
<th>0.99</th>
<th>0.99</th>
<th>1.00</th>
<th>1.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>bayes.BayesNet</td>
<td>221</td>
<td>0.66</td>
<td>0.69</td>
<td>0.81</td>
<td>0.82</td>
<td>0.50</td>
<td>0.50</td>
<td>0.69</td>
<td>0.71</td>
<td>0.67</td>
<td>0.67</td>
<td>0.00</td>
<td>0.00</td>
<td>0.10</td>
<td>0.01</td>
<td>0.00</td>
</tr>
<tr>
<td>Functions.Logistic</td>
<td>31972</td>
<td>0.57</td>
<td>0.64</td>
<td>0.61</td>
<td>0.69</td>
<td>0.70</td>
<td>0.50</td>
<td>0.50</td>
<td>0.56</td>
<td>0.57</td>
<td>0.55</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.01</td>
<td>0.01</td>
</tr>
<tr>
<td>Lazy.LBR</td>
<td>72</td>
<td>0.67</td>
<td>0.69</td>
<td>0.81</td>
<td>0.82</td>
<td>0.50</td>
<td>0.50</td>
<td>0.69</td>
<td>0.71</td>
<td>0.67</td>
<td>0.67</td>
<td>0.00</td>
<td>0.00</td>
<td>0.10</td>
<td>0.01</td>
<td>0.00</td>
</tr>
<tr>
<td>Lazy.KStar</td>
<td>3</td>
<td>0.97</td>
<td>0.97</td>
<td>0.97</td>
<td>0.97</td>
<td>0.50</td>
<td>0.50</td>
<td>0.51</td>
<td>0.51</td>
<td>0.49</td>
<td>0.66</td>
<td>0.70</td>
<td>0.99</td>
<td>0.99</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Misc.VFI</td>
<td>33</td>
<td>0.52</td>
<td>0.54</td>
<td>0.54</td>
<td>0.54</td>
<td>0.50</td>
<td>0.50</td>
<td>0.52</td>
<td>0.52</td>
<td>0.52</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Rules.JRip</td>
<td>3787</td>
<td>0.59</td>
<td>0.59</td>
<td>0.59</td>
<td>0.59</td>
<td>0.50</td>
<td>0.50</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
</tr>
<tr>
<td>Rules.ConjunctiveRule</td>
<td>319</td>
<td>0.53</td>
<td>0.53</td>
<td>0.53</td>
<td>0.53</td>
<td>0.50</td>
<td>0.50</td>
<td>0.51</td>
<td>0.51</td>
<td>0.51</td>
<td>0.51</td>
<td>0.51</td>
<td>0.51</td>
<td>0.51</td>
<td>0.51</td>
<td>0.51</td>
</tr>
<tr>
<td>Rules.DTNB</td>
<td>1367</td>
<td>0.96</td>
<td>0.96</td>
<td>0.96</td>
<td>0.96</td>
<td>0.50</td>
<td>0.50</td>
<td>0.96</td>
<td>0.96</td>
<td>0.96</td>
<td>0.96</td>
<td>0.96</td>
<td>0.96</td>
<td>0.96</td>
<td>0.96</td>
<td>0.96</td>
</tr>
<tr>
<td>Rules.DecisionTable</td>
<td>3521</td>
<td>0.96</td>
<td>0.96</td>
<td>0.96</td>
<td>0.96</td>
<td>0.50</td>
<td>0.50</td>
<td>0.51</td>
<td>0.51</td>
<td>0.51</td>
<td>0.51</td>
<td>0.51</td>
<td>0.51</td>
<td>0.51</td>
<td>0.51</td>
<td>0.51</td>
</tr>
<tr>
<td>Rules.ZeroR</td>
<td>27</td>
<td>0.53</td>
<td>0.53</td>
<td>0.53</td>
<td>0.53</td>
<td>0.50</td>
<td>0.50</td>
<td>0.51</td>
<td>0.51</td>
<td>0.51</td>
<td>0.51</td>
<td>0.51</td>
<td>0.51</td>
<td>0.51</td>
<td>0.51</td>
<td>0.51</td>
</tr>
<tr>
<td>Rules.PART</td>
<td>106861</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
</tr>
<tr>
<td>Trees.J48</td>
<td>12078</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
</tr>
<tr>
<td>Trees.ADTree</td>
<td>358</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
</tr>
<tr>
<td>Trees.REPTree</td>
<td>1163</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
</tr>
</tbody>
</table>

Table 5: Test 5 results
One fundamental problem is that our assumption that splitting multi-value attributes into multiple instances could work has an unintended side effect. This action strengthens the associations between the common attributes in the split, which introduces false bias into the data.

We anticipate that the actual data will have many small but significant rules—rules with low support but high confidence. These classifiers did no seem to find such rules in our experiments. Unless the classifiers can be tuned to our problem, these experiments seem to reveal that they are incapable of performing as we would hope.

**Future Work**

**Collaborative Filtering**

We focused on our own version of content-based filtering because of the lack of feedback data—the cold start problem. We would definitely need to add in collaborative filtering to flush out the recommender system. This specific instance of a recommender system is a prime candidate for a hybrid method of collaborative and content-based filtering. [14]

**HITS**

An interesting algorithm to adapt to the collaborative filtering need would be HITS. Traditionally used to rank web pages for search engines, HITS finds hubs and authorities in a bi-partite graph. One adaptation of HITS for the announcements system would be to use users as hubs, announcements as authorities, and feedback as the directed edges from the users to the announcements. [1]

The adaptation could start with a single user (for whom the system is predicting feedback). The graph would be grown by adding in the announcements to which the user points, adding in additional users providing the same feedback to the same announcements, and then adding in the announcements for the additional users. The adaptation could then find a community of users and announcements containing the original user, and then propagate the expected feedback ratings of the additional users back to the original user.
**Association Rule Mining**

Another data mining technique that could be adapted to collaborative filtering is association rules data mining. Apriori can handle more than one item in the consequent (right-hand side) of the association rule. However, there is no distinction between which items can appear in the antecedent (left-hand side) of the rule versus the consequent of the rule. The algorithm could be modified to only consider rules where one type of item (either user attributes or announcement attributes, but not both) appears together on the same side of the rule. We would not want to mine a rule CSC → Male, instead CSC → data mining. The support and confidence metrics of the frequent item sets could be used to generate scores for announcements. [1]

**Data Extensions**

**Non-binary Feedback**

Our experimental data set only allowed two values for the feedback class: like or dislike. The data set was in part determined by the classifier algorithms with which we wanted to experiment. Classifier algorithms could be adapted to use multi-valued classes such as a scale of 1-5 or like, dislike, neutral.

**Implicit Feedback**

Another interesting addition to the system would to use not just explicit feedback, but also implicit feedback. Explicit feedback will take the form of a user actually providing a rating for an announcement. Implicit feedback of a user following a link to the announcement could be used to alleviate a scarcity of explicit feedback in the system, potentially with a different weight than the explicit feedback.

**Training Set**

The recommender system models will have to be re-trained at some interval. The production data set retains information on all users indefinitely; similarly, the announcement system will retain all announcement and feedback data indefinitely. When the recommender system is re-trained, it may use, for instance, only users who are still current Cal Poly portal users or announcements with an effective end date within one year the re-training. Models could also be trained on subsets of the data, such as only students, only men, or only people within a certain department.
Integration with Announcement System

The recommender system will be integrated with the larger announcement system. The recommender system will essentially be treated as a black box—a user is passed in to the recommender, and an ordered list of announcements is returned. The internal workings of the recommender system will have to be altered for deployment in a production environment. The system will link directly to the warehouse to access user attributes and to the announcements database for announcement data.

References

Appendix A: Classifiers

**AODE / AODEsr**
http://weka.sourceforge.net/doc/weka/classifiers/bayes/AODE.html
http://en.wikipedia.org/wiki/AODE

**Naïve Bayes**
http://en.wikipedia.org/wiki/Naïve_Bayes
http://weka.sourceforge.net/doc/weka/classifiers/bayes/NaiveBayes.html

**BayesNet**
http://weka.sourceforge.net/doc/weka/classifiers/bayes/BayesNet.html
http://en.wikipedia.org/wiki/Bayes_net

**Logistic**
http://weka.sourceforge.net/doc/weka/classifiers/functions/Logistic.html
http://en.wikipedia.org/wiki/Linear_classifier

**LBR (Lazy Baysian Rules)**
http://weka.sourceforge.net/doc/weka/classifiers/lazy/LBR.html
http://weka.sourceforge.net/doc/weka/classifiers/lazy/LBR.html

**Kstar**

**Jrip**

**Conjuctive Rule**
DTNB (Decision Table Naïve Bayes)

Decision Table

http://weka.sourceforge.net/doc/weka/classifiers/rules/DecisionTable.html

ZeroR


http://en.wikipedia.org/wiki/Association_rule_learning

PART


J48


http://en.wikipedia.org/wiki/Treatment_learner

ADTree (Alternating Decision Tree)

http://weka.sourceforge.net/doc/weka/classifiers/trees/ADTree.html

http://en.wikipedia.org/wiki/ADTree

REPTree

http://weka.sourceforge.net/doc/weka/classifiers/trees/REPTree.html