A Cold Start Recommendation System Featuring Item Correlation

Suchismit Mahapatra
Department of CSE
University at Buffalo
Buffalo, NY 14226
suchismi@buffalo.edu

Alwin Tareen
Department of CSE
University at Buffalo
Buffalo, NY 14226
tareen@buffalo.edu

Ying Yang
Department of CSE
University at Buffalo
Buffalo, NY 14226
yyang25@buffalo.edu

ABSTRACT

Conventional recommendation systems tend to focus on variations of well-known information retrieval techniques. We took a fresh approach, rather than to follow the traditional, commonly applied recommendation methodology of creating a user-item matrix, and then using them to make recommendations. Instead, we established and examined three types of relationships: user-user similarity, wine-wine similarity and user preference relationships, in the form of adjacency lists. Using this approach, we did not encounter the usual problems associated with large dimension matrices, such as *sparsity*[5] and *synonymy*, as well as the basic problems of storing the large matrix, and having to perform a large number of computations every single time. We attempted to address the *synonymy* problem by using the wine-wine similarity index we formed. Also, we developed a model that took care of the *cold start*[6] problem which is fairly common in recommendation systems. We attempted to address the *grey sheep* problem as well, by minimizing the effect of any one outlying element, and taking the overall cumulative effect of all the elements. With our recommendation system, we wanted to establish a relationship of trust with the user, because just a few erroneous values would greatly discredit our models.

In this paper, we defined 25 attributes by which a wine could be classified, and determined the wine-wine and user-user similarities, as well as the user preferences for the wines he has tried so far using our own novel scoring mechanism. The wines as such have had no scores associated with them with regards to the 25 attributes, rather, the scores were meaningful only when the relationships between the two wines were considered. We defined our own two methodologies, *cold start* and *top k1, k2* models to address the issues associated with recommendation systems. Our experimental results showed that the newly introduced *cold start* model performed better than the traditional models of *pure content-based*[3], *pure collaborative-based* and *content + colloborative based*[10, 1] approaches with regards to the quality. Also, our *top k1, k2* model performed almost as well as the *content + colloborative* based model. Our resulting wine recommendations were consistent with the expectations of the user's particular wine tastes.

CSE 635 '11 Buffalo, NY USA

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*information filtering, retrieval models*; I.2.6 [**Artificial Intelligence**]: Learning—*knowledge acquisition*

General Terms

Algorithms, Experimentation, Theory

Keywords

Recommendation model, collaborative filtering, content filtering, cold start problem, grey sheep problem, sparsity, synonymy

1. INTRODUCTION

The objective of this project was to create a recommendation system for wines based on a series of descriptive attribute phrases and classifier tags. Such a system was composed of several distinct modules, which were responsible for the bulk of the decision making process. There were several different types of algorithms that contributed to the final decisions, and the recommendation system utilized a blend of these. We regarded user similarity as a basic comparison of terms. We defined item correlation as a concept that extended that of similarity by considering more factors, in our case, brand, region, price, etc. We extended a similar line of thinking in our project, since the item-item algorithms performed better than the user-user algorithms. We have strived to make the item-item correlation as accurate as possible, since it makes a larger contribution to good recommendations.

The first part of the project dealt with collecting the respective data for the input of the system. We used an iterative approach for this. This task was accomplished with a smart web crawler, directed toward three wine-based vendor websites, namely, `eBacchus.com`, `wine.com` and `tastings.com`. We selected these three websites because they had good quality wine tasting notes, and the reviewers had a high level of expertise. Once a significant amount of data had been retrieved, it was parsed into its constituent pieces. We used Stanford's part-of-speech tagging tool to extract attribute

phrases based on lexical patterns, such as *adjective-adjective*, or *adjective-noun*. We learnt from the verbose, poetic style of language that was usually employed by the various wine reviewers and experts. Then, a database schema was established, where each of the column headings corresponded to a type of wine, a vintage, a price, etc. Specifically, we derived all of the various attributes that a wine could have, and we populated our database with those attributes. Once we had a suitable table of data to work with, we were able to apply various information retrieval techniques.

With the wine data set in place, our next task was to analyze the various attributes that were associated with each wine. We applied a word frequency count to the entire data set, which produced a list of the most commonly expressed attributes. We selected the top 25 attributes from this list, for use as classification tags. By tagging the attributes[8] in the data set with one or more of these classifiers, we were able to create a level of normalization. Then, we investigated the levels of commonality that existed between various pairs of wines our data set. From this investigation, we were able to create a Wine-Wine Similarity Index, which expressed the level of similarity between a particular wine, and other wines on a top-N[2] ranked list. *Jaccard's coefficient* was used to calculate the degree of similarity. Higher scores were assigned to pairs of wines that exhibited close levels of attribute similarity, and some weight was also given to region and brand identifiers.

The next stage in our project was to determine the relationships between the types of wine-purchasing customers[3] in our data set. Specifically, we had to establish groups or clusters of customers, that is, people who had similar tastes or interests in wine. This was also known as *neighborhood formation*. We applied *Jaccard's coefficient* to compute the level of similarity between users. When the similarities between all pairs of users had been established, we created a User-User Similarity Index. For each user, we created a list of wines that the system believed they had a strong preference for. We wanted to avoid false positives in our system at all costs, because they could cause customers to lose faith in our recommendations. In this paper, we used a four-pronged approach to make our recommendations: *wine-wine similarity*, *user-user similarity*, *attribute-classification based* and *user preference*.

		User
	B: similar wine tastes + dissimilar wine prices	A: similar wine tastes + similar wine prices
User	C: dissimilar wine prices + dissimilar wine tastes	D: similar wine prices + dissimilar wine tastes

Table 1: Feature comparison. A > B > C > D. In our case, [similar wines] > [wines with similar prices]

2. RELATED WORK

In our recommendation system, we decided to take a different approach to the problem. Existing approaches tend to focus on user-item matrix techniques, and their models reflect this line of thinking. We still do similarity calculations, but in a different way. There are some concepts that we use, which are common to most currently existing recommendation systems. Namely, we use model-based collaborative filtering, content filtering, clustering and neighborhood formation.

Collaborative filtering-based recommender systems rely on information derived from the social activities of users, such as opinions or ratings, to form predictions, or produce recommendation lists. Existing collaborative filtering techniques involve generating a user-item matrix, from which recommendation results could be derived. Content-based filtering focuses on the selection of relevant items from a large data set, things that a particular user has a high probability of liking. This involves training the data set with maching learning techniques. Clustering involves sectioning the data set into particular sets, each of which correspond to certain preference criteria. Also, typical recommendation systems output their results either as *predictions*, a numerical ranking value corresponding to a particular item or *recommendations*, a list of relevant items.

The conventional approaches to computing similarity involve the use of two popular techniques: *pearson correlation similarity*, given by Equation 1, and *cosine similarity*, given by Equation 2. We used *Jaccard's coefficient* in our models, to compute similarity.

$$\begin{matrix}\text{Pearson}\\ \text{Correlation}\\ \text{Similarity}\end{matrix} = \frac{\sum_{j=1}^{l}(r_{ij}-\bar{r}_i)(r_{kj}-\bar{r}_k)}{\sqrt{\sum_{j=1}^{l}(r_{ij}-\bar{r}_i)^2\sum_{j=1}^{l}(r_{kj}-\bar{r}_k)^2}} \quad (1)$$

$$\text{Cosine Similarity} = \sum_j \frac{r_{ij}}{\sqrt{\sum_j r_{ij}^2}}\frac{r_{kj}}{\sqrt{\sum_j r_{kj}^2}} \quad (2)$$

Shi[9] has proposed a novel approach to the collaborative filtering technique, by generating a list of top-N recommendations, then applying a re-ranking algorithm to this list. In this manner, the recommendation list becomes more refined by using the information derivable from the original data set. We intend to build upon this technique in our future work.

3. DATA ACQUISITION

3.1 Crawling and Parsing of the Data

We surveyed many wine-specific websites to use as potential data crawling candidates, but we decided to use three particular ones: `eBacchus.com`, `wine.com` and `tastings.com`. We discovered that the various wine tasting notes that were listed on these websites were of a particularly high quality, and they were mostly composed of relevant content. Also, the types of reviewers that tended to post comments on these websites tended to be highly knowledgeable in the area of wine tasting expertise.

We attempted to use currently available website crawling tools to achieve our data collection, such as `80legs`, `nutch` and `ASPSeek`. However, we found that these web crawlers were unnecessarily complicated to use, and they simply did not meet our specific needs. Therefore, we wrote our own Java-based crawler, and used smart crawling techniques that satisfied certain patterns. We collected a total of 28000 individual wine review files, however, we discovered that the

majority of these did not contain any useful wine review information. So, we discarded those particular reviews, and we also merged the wines that were in common, into single wine files. After this, our data set contained 6000 wines.

We used the open-source tool `Apache Tika` to remove the HTML tags from each of these files. Then, we wrote multiple `bash` scripts to remove the various headers, footers and erroneous data from these files. Once the cleaning of the data was complete, we moved on to the part-of-speech tagging phase. In order to accomplish this task, we used the Stanford Log-Linear Part of Speech Tagger. This tool enabled us to accurately identify the particular types of words in our wine files, such as adjectives, adverbs and nouns. The information contained in these specific tags represented the most relevant types of information that could be used to deduce similarity (see Figure 1).

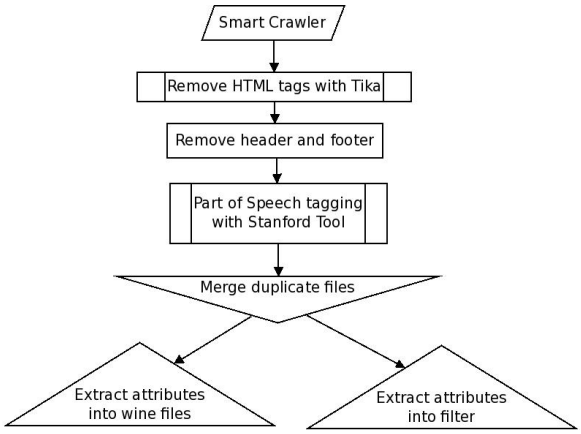

Figure 1: Crawling and cleaning

Our next task was to remove unintended and irrelevant groups of words from our data set, such as usernames or geographic locations. We extracted all of the attributes, and determined 27 lexical patterns that represented highly significant word groupings, such as *adjective-adjective* or *adjective-conjunction-noun*. We sorted through each of these word groupings, line by line, and retained the relevant ones into filter files. Then, we cross-referenced these filter files with our wine attribute files, and this resulted in clean, relevant data. Also, we performed a thorough spell check on each of the filter terms, to ensure that our resulting data set was of a high quality (see Figure 2).

3.2 Classification of Attributes

When we examined the wine data set, we discovered that the vast majority of attribute terms were of an ambiguous nature. That is, they tended to be somewhat subjective, and their intended meaning wasn't always expressly clear. As such, we decided to place each of these attributes into individual class categories.

We were then faced with the problem of defining the individual class categories, and we devised a novel solution for this. We ran a word frequency counter on the entire data set, and selected the top 25 relevant terms as our classifier tags. A sample set of the top 10 terms, along with their respective frequency counts, are given in Figure 3. The full set of the attribute classifier tags that we selected are listed

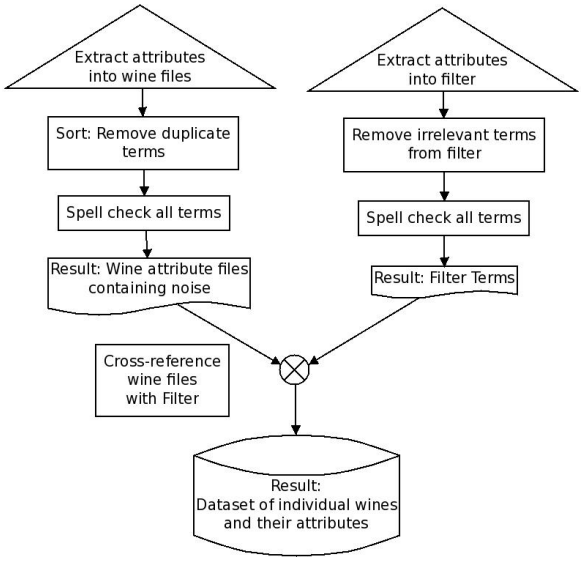

Figure 2: Parsing and filtering

in Table 2.

Our next task was to examine each of the attribute phrases in our data set, and tag them with one or more of our classifier tags. In the example in Figure 4, the phrase "flavorful clean fresh" was tagged with the classes *taste* and *freshness*. This proved to be a mundane, labor-intensive process, as there were roughly 30000 attribute phrases that had to be manually inspected and tagged. We managed to successfully tag about 20000 of the attributes, and those were the ones that contributed to our models.

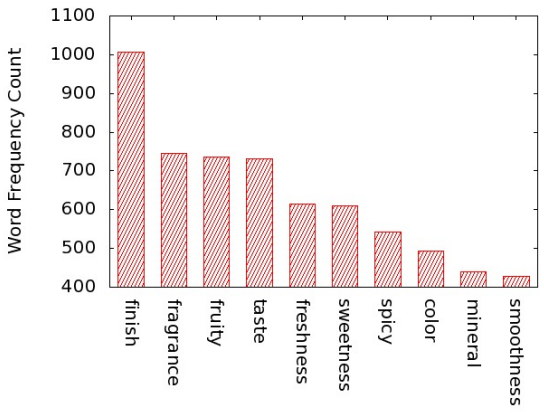

Figure 3: Relative frequency of classifier tags ($T = 10$)

3.3 Database Integration

We parsed each of the 6000 wine attribute files and inserted them into a MySQL database. Also, we included files that contained information that was specific to the indentification of each wine, such as name, brand, region and vintage. Since we used three different websites to collect the data, our parsing strategy had to be modified to suit

taste	fruity	spicy
sweetness	balance	fullness
texture	finish	depth
freshness	crispness	strength
dryness	tannin	smoothness
fragrance	color	complexity
mineral	acidity	vintage
category	subjective	transparency
priceopinion		

Table 2: Attribute classifier tags ($T = 25$)

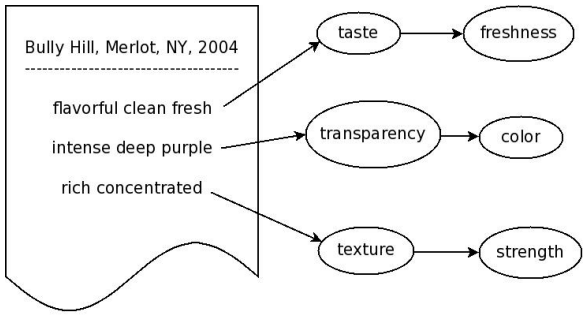

Figure 4: Attribute classification

each one. After all the data from the wine attribute files had been collected, we generated the User-Wine Preference, the Wine-Wine Similarity and User-User Similarity Indexes, using the equations discussed in the section 4.

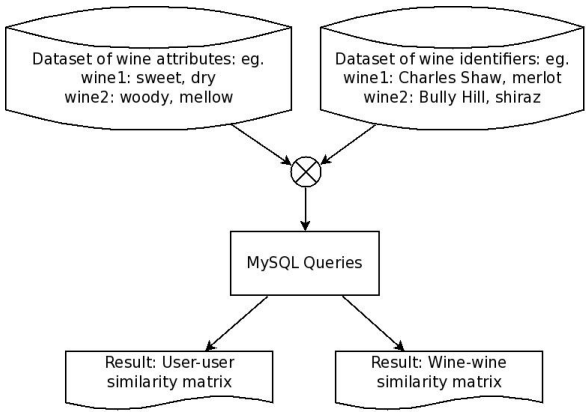

Figure 5: Database integration

4. WINE INDEXING THEORY

According to [7], the item-based algorithms worked better than the user-based algorithms. In our models, we have followed a similar line of reasoning, by giving more weight to the Wine-Wine Similarity values, than to the User-User Similarity values.

4.1 User-Wine Preference

We defined *wine frequency* as the number of times a particular user has purchased a certain wine, irrespective of the number of bottles in each individual purchase.

In our analysis, when a user bought a wine with a wine frequency of 1, it need not represent his fondness for the wine, but instead, it might represent an experimental or gift purchase. Such a purchase would have no relation to his particular tastes. Wines with a wine frequency of 2 or greater can not represent coincidence, in general circumstances. Also, the user does not like wines with a wine frequency of 2, twice as much as he likes the wines with a wine frequency of 1. Rather, it represents a more damped version of interest which can be represented by the formula $1 + \log(\text{freq})$.

We found that, by increasing wine frequencies, such as 3 from 2, represent a smaller increase of affinity. Thus, we found the above formula to appropriately represent the frequency relation. The number of bottles the user purchased can be a good estimate of his interest for that particular wine. We have used made extensive use of the log function to dampen the overall effect, and to maintain a uniform scoring profile. In Equation 3, the numerators for the different cases evaluated to numbers less than 1.7. Therefore, we used the value of 1.7 in the denominator, which along with the multiplier of 10, gave us scores in the range of 0 to 10.

$$\text{UPI Score} = \frac{\log(N_{bot} \times (1 + \log(\text{freq})))}{1.7} \times 10 \qquad (3)$$

4.2 Wine-Wine Similarity

Each wine had a certain number of attribute phrases associated with it. Some of these phrases were highly relevant, while other were less so. In order to deal with this sort of attribute variation, we devised a system of classification tagging. First, we applied a word frequency count to the entire wine data set, and selected the top 25 most popular terms for use as classification tags. Then, we applied one or more of these tags to each of the attribute phrases that were associated with every wine. In doing so, we were able to sufficiently categorize each of these attributes. When wine_1 and wine_2 were compared with each other, their respective attributes and classifier tags were matched up, and Jaccard's Coefficient (Equation 5) was applied to determine the degree of similarity between wine_1 and wine_2.

$$\text{score}_i = \text{JC}(\text{wine}_1\text{attr}_i, \text{wine}_2\text{attr}_i) \qquad (4)$$

$$= \frac{\text{wine}_1\text{attr}_i \cap \text{wine}_2\text{attr}_i}{\text{wine}_1\text{attr}_i \cup \text{wine}_2\text{attr}_i} \qquad (5)$$

Two other similarity aspects that we wanted to introduce into our calculations were the brand and region similarities. These were evaluated according to Equations 6 and 7.

$$\text{score(brand)} = \delta(x, y) \begin{cases} 0.5 & \text{if} \quad x = y \\ 0 & \text{if} \quad x \neq y \end{cases} \qquad (6)$$

$$\text{score(region)} = \delta(x, y) \begin{cases} 0.1 & \text{if} \quad x = y \\ 0 & \text{if} \quad x \neq y \end{cases} \qquad (7)$$

Once the above three terms had been determined, the Wine-Wine Similarity Index(WWSI) between wine_1 and wine_2 was the sum of these three components as given by Equation 8.

$$\text{WWSI Score} = \frac{1}{N}\sum_{i=1}^{N} \text{score}_i + \text{score(brand)} + \text{score(region)} \tag{8}$$

Where i is one of the attributes in each wine.

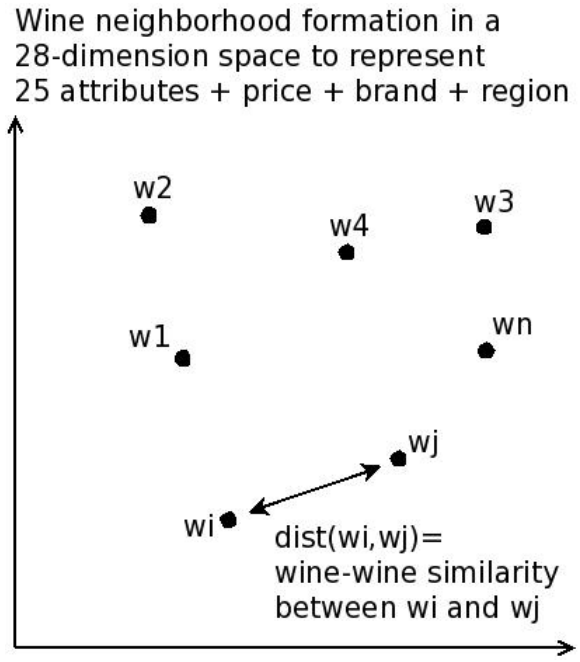

Figure 6: Wine-Wine Similarity dimension space. Lower distance($\mathbf{w}_i$,$\mathbf{w}_j$) $\implies$ $\mathbf{w}_i$ and $\mathbf{w}_j$ are similar. If user $\mathbf{U}_i$ likes $\mathbf{w}_i$ $\implies$ high probability that user $\mathbf{U}_i$ will like $\mathbf{w}_j$ to a similar degree.

4.3 User-User Similarity

Consider the following, where U_1 and U_2 refer to particular users from our user database. User U_1 has tried m wines, given by: wine_1,wine_2,..., wine_m, and User U_2 has tried n wines, given by: wine_{m+1},wine_{m+2},..., wine_{m+n}.

$$\begin{aligned} U_1 &= \{\text{wine}_1, \text{wine}_2, \ldots, \text{wine}_m\} \\ U_2 &= \{\text{wine}_{m+1}, \text{wine}_{m+2}, \ldots, \text{wine}_{m+n}\} \end{aligned}$$

We calculated the similarity between the two wines by considering both the price aspect and attribute list. We define the Maximum Price Difference (MPD) as difference between the maximum and minimum prices of the wines in our database. For two wines, wine_i and wine_j, we define f to be the difference between their prices divided by the MPD as given in Equation 9.

$$f = \frac{\text{Price Difference}_{ij}}{\text{Maximum Price Difference}} \tag{9}$$

For wines wine_i and wine_j, score_{ij} is defined by Equation 10.

$$\text{score}_{ij} = JC(\text{wine}_i, \text{wine}_j) = \frac{\text{wine}_i \cap \text{wine}_j}{\text{wine}_i \cup \text{wine}_j} \tag{10}$$

$\forall$ attributes $\in$ Attribute List

If the prices of wine_i and wine_j are very similar, then the value of f in this case is very small. However, the price similarity factor is high in this case, thus we have used a factor of $(1-f)$ to represent this relationship.

$$\begin{matrix}\text{wine similarity} \\ \text{component}\end{matrix} = \begin{matrix}\text{average of score}_{ij} \\ \text{for the 25 attributes}\end{matrix}$$

We have calculated the Wine-Wine Similarity (WWS) using weight components of price similarity factor and wine similarity component. We introduced a weight parameter of α, and used it in Equation 11, where $\alpha = 0.8$.

$$\text{WWS} = \alpha \times (1-f) + (1-\alpha) \times \text{similarity component} \tag{11}$$

We define the User-User Similarity Index (UUSI) as given by Equation 12.

$$\text{UUSI} = \frac{1}{U_1 + U_2}\sum_{i=1}^{U_1}\sum_{j=1}^{U_2} \text{score}_{ij} \times (\text{WWS}) \tag{12}$$

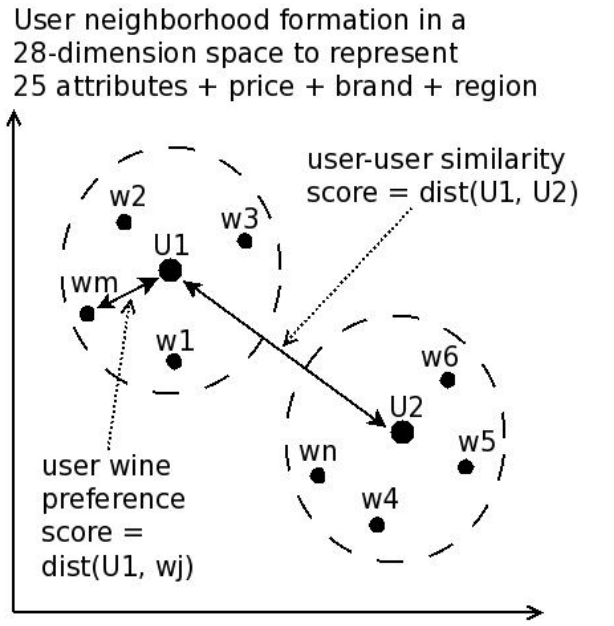

Figure 7: User-User Similarity dimension space. Points $\mathbf{U}_1$, $\mathbf{U}_2$ represent users who are defined by the wines they have tried thus far, namely wines $\mathbf{w}_1$, $\mathbf{w}_2$,...,$\mathbf{w}_m$ and wines $\mathbf{w}_4$, $\mathbf{w}_5$,...,$\mathbf{w}_n$ (defined by the points as shown) respectively. Lower distance($\mathbf{U}_i$,$\mathbf{w}_j$) $\implies$ user $\mathbf{U}_i$ likes wine $\mathbf{w}_j$ more $\implies$ user $\mathbf{U}_i$ will like wines closest to the point $\mathbf{w}_j$ to a similar degree.

5. RECOMMENDATION MODELS

We performed an analysis of the conventionally used methodologies that are popular in modern recommendation sys-

tems, namely, content and collaborative filtering. We examined a pure content based model, as well as a pure collaborative based model, based on user similarity. Then, we combined these two methodologies into a blended form, and analyzed its performance. In doing so, we wanted to establish a benchmark by which we could effectively measure our two newly established recommendation models, namely, the *cold start* model and the *top k1, k2* model.

5.1 Pure Content Based Model

In this model, we applied the User Preference Index to generate a list of recommended wines, for the particular user.

Algorithm 1: Pure content based

Input: user-wine preference list
Output: list of recommended wines
foreach *wine in user-wine preference list for user* **do**
    put wine into list of recommended wines
end

5.2 Pure Collaborative Based Model

We performed calculations with the User-User Similarity Index to produce the recommendation list. First we got the list of similar users to the particular user in question, along with the respective similarity scores. Subsequently, we got the user preference lists for each of these similar users. The wines in them formed the recommendation candidates. We have used a combination of the user similarity scores along with the user preference scores to get the most suitable list of recommendations. We made use of an intermediate list. This was where we scored wines using the User-User Similarity Index value (which was a fractional value) and the User Preference Index score, for that particular user. In case a wine occurred in multiple user preference indexes, for different users, we added this score to the running total. At the end of this, we took the wines with the highest scores as our recommendations.

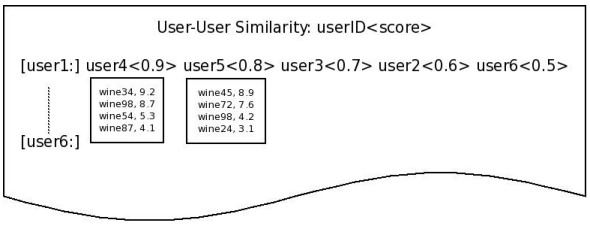

Figure 8: User-user similarity matrix

5.3 Content + Collaborative Model

We determined the candidates for recommendation by using a combination of the User Preference Index, and the Wine-Wine Similarity Index. First, we got the list of wines in the User Preference Index, for that particular user. Subsequently, we got the most similar wines to the previously mentioned list using the Wine-Wine Similarity Index. We made use of the same type of intermediate list strategy that we used in the collaborative model, however, in this case, we used the Wine-Wine Similarity Index and the User Preference Index as our inputs.

Algorithm 2: Pure collaborative based

Input: user-user similarity and user-wine preference lists
Output: list of recommended wines
`// intermed-list = intermediate list holding wines, scores associated with them`
foreach *item in user-user similarity list for user* **do**
    similar-user = item.user
    user-user-sim-score = item.score
    foreach *wine in user-wine preference list for similar-user* **do**
        user-wine-score = user-user-sim-score × wine.user-pref-score
        if *intermed-list contains wine* **then**
            `// update wine score for wine`
            intermed-list-wine-score = intermed-list-wine-score + user-wine-score
        end
        else
            put <wine, user-wine-score> into intermed-list
        end
    end
end
put top ranked wines from intermed-list into list of recommended wines

Algorithm 3: Content + Collaborative based

Input: wine-wine similarity and user-wine preference lists
Output: list of recommended wines
`// intermed-list = intermediate list holding wines, scores associated with them`
foreach *item in user-wine preference list for user* **do**
    wine = item.wine
    item-wine-score = item.wine-score
    foreach *sim-wine in wine-wine similarity list for wine* **do**
        wine-score = item-wine-score × sim-wine.wine-score
        if *intermed-list does not contain sim-wine* **then**
            put <sim-wine, wine-score> into intermed-list
        end
        else
            `// update wine score for sim-wine`
            intermed-list-sim-wine-score = intermed-list-sim-wine-score + wine-score
        end
    end
end
put top ranked from intermed-list into list of recommended wines

5.4 Cold Start Model

Typically, we encountered cases whree users have tried very few wines. In these cases, recommending items for them becomes pretty difficult, or biased. This is known as the cold start problem. As such, we can't make assumptions about what the user might like, because they might result

in false positives, which erodes the trust factor[10]. In the cold start model, we have attempted to tackle this problem. The user starts with an initial list of wine preferences, and an expanded list is formed from this, taking into account the Wine-Wine Similarity Index. We then applied our aforementioned collaborative techniques on this expanded list, making use of the intermediate list strategy with the expanded list of user preferences and the Wine-Wine Similarity Index as the inputs. The top ranked results from this procedure are returned as recommendations.

We would like to explain a few situations that can occur in this model. Referring to the example in Figure 9, the initial list contains `wine34, wine23, wine37, wine81`, with `wine34` on top. However, in the expanded list, `wine68` has the highest score. This happened due to the cumulative effects of `wine68` being in the Wine-Wine Similarity Indexes of multiple wines present in the initial list of user preferences. Another scenario which could be observed in Figure 9 was that `wine37` had a score of 4.7 in the initial list of user preferences, whereas it had a score of 5.1 in the expanded list. This was again due to the effect of `wine37` being present in the Wine-Wine Similarity Index of one of the wines in the initial list. This resulted in its final score being greater than its base score. There was also the case of wines being present in the initial list and absent in the expanded list. This happened because other wines with higher scores rose up in the ranking, and pushed them out. For example, this situation happened to `wine81` in Figure 9, in our example.

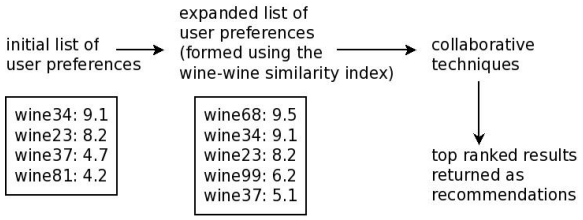

Figure 9: Cold start

5.5 Top k1, k2 Model

This model was a variant of Model 3, the simple content-collaboration based methodology. We considered the top k1 user preferences and the associated top k2 wines with the highest Wine-Wine Similarity Index scores. This results in a total of k1 × k2 candidates as recommendations. This model is based on the principle that the most similar wines(k2) to the wines the user likes the most(k1) are the most suitable recommendations. There can be another variation wherein we consider threshold scores as our boundary, both in the cases of user preferences, and wine-wine similarities.

6. RESULTS AND DISCUSSION

The results were generated from random user simulations, since we did not have a comprehensive user purchase data to work from. We had to create our own users and had to tag them with certain wines that were of a particular type. This was important because we didn't have a lot of wine expertise to be able to distinguish between subtle differences between two or more closely related wines. For example, if, for a particular user who had tried only light red wines our

Algorithm 4: Cold start model

Input: wine-wine similarity and user-wine preference lists
Output: list of recommended wines
`// expanded-user-prefs = intermediate list holding expanded user wine preferences`
`// intermed-list = intermediate list holding wines, scores associated with them`

```
1  foreach item in user-wine preference list for user do
2      put item into expanded-user-prefs
3      wine = item.wine
4      item-wine-score = item.wine-score
5      foreach sim-wine in wine-wine similarity list for
       wine do
6          wine-score = item-wine-score ×
           sim-wine.wine-score
7          if intermed-list does not contain sim-wine then
8              put <sim-wine, wine-score> into
               intermed-list
9          end
10         else
               // update wine score for sim-wine
11             intermed-list-sim-wine-score =
               intermed-list-sim-wine-score + wine-score
12         end
13     end
14 end
15 put top ranked from intermed-list into
   expanded-user-prefs
16 foreach item in expanded-user-prefs for user do
17     wine = item.wine
18     item-wine-score = item.wine-score
19     foreach sim-wine in wine-wine similarity list for
       wine do
20         wine-score = item-wine-score ×
           sim-wine.wine-score
21         if intermed-list does not contain sim-wine then
22             put <sim-wine, wine-score> into
               intermed-list
23         end
24         else
               // update wine score for sim-wine
25             intermed-list-sim-wine-score =
               intermed-list-sim-wine-score + wine-score
26         end
27     end
28 end
29 put top ranked from intermed-list into list of
   recommended wines
```

recommendation system returned a wine result which was white, or strong, then we would be able to regard it as incorrect. This meant that our tests were relatively simplistic and non-exhaustive, which made our testing methodology slightly biased. Situations in which the user had bought more wines produced better results.

We created each of our models in a `C++` software program. We used our indexes that we generated in the previous stages. We chose recommendation basket sizes of 10, 15, and 20. We ran simulations for each of the five models, for the user simulations and some users from the actual customer purchase data that was provided. The metric we used

Algorithm 5: Top k1, k2 model

Input: user-wine preference and wine-wine similarity lists
Output: list of recommended wines
// intermed-list = intermediate list holding wines, scores associated with them
// top-k1-user-prefs = list containing top k1 user prefs
// top-k2-wine-similarities = list containing top k2 most similar wines for a particular wine
1 top-k1-user-prefs = top-k1(user-wine preference-list)
2 top-k2-wine-similarities = top-k2(wine-wine similarity list)
3 **foreach** *item in top-k1-user-prefs for user* **do**
4 wine = item.wine
5 item-wine-score = item.wine-score
6 **foreach** *sim-wine in top-k2-wine-similarities for wine* **do**
7 wine-score = item-wine-score × sim-wine.wine-score
8 **if** *intermed-list does not contain sim-wine* **then**
9 put <sim-wine, wine-score> into intermed-list
10 **end**
11 **else**
// update wine score for sim-wine
12 intermed-list-sim-wine-score = intermed-list-sim-wine-score + wine-score
13 **end**
14 **end**
15 **end**
16 put top ranked from intermed-list into list of recommended wines

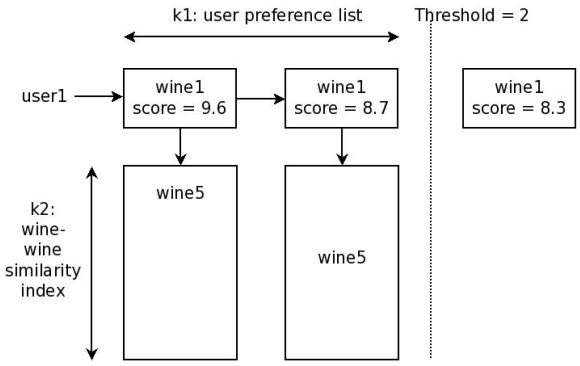

Figure 10: Top k1, k2 model. The threshold can also be score based, and a threshold of 8.5 in the above figure would give the same results.

to evaluate the results for each of the 5 models, for varying basket sizes, was Root Mean Square(RMS). For example, suppose $user_i$ likes subtle, sweet red wines. A simulation run of one of our models, for a basket size of 10, returned 9 results which matched the user's tastes and another result for a strong, dry white wine.

Precision is defined as the fraction of the results that match the user's needs. In our case, the user's particular

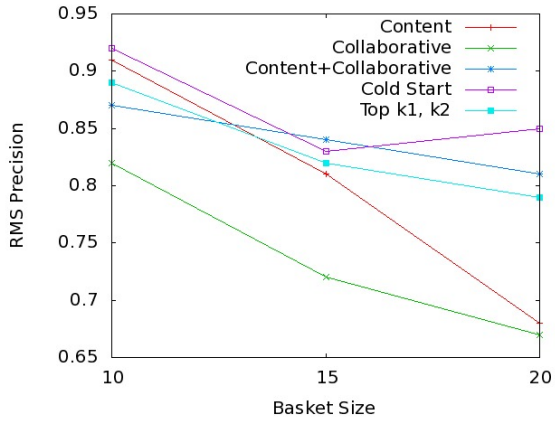

Figure 11: Evaluation of the 5 recommendation system models.

wine preferences. In the above example, the precision was calculated as 0.9. This was just one instance of the results, and by running simulations for each of our different models on varying basket sizes, for the different users, we got our set of precision values. We applied the RMS equation to obtain a generalized mean for each of the models, and for varying basket sizes. In this manner, we obtained the results of Figure 11. The RMS relation is given in Equation 13.

$$x_{rms} = \sqrt{\frac{x_1^2 + x_2^2 + \ldots + x_n^2}{n}} \tag{13}$$

7. CONCLUSIONS AND FUTURE WORK

We found that the pure content method precision graph drops, for increasing values of basket size. This is possibly due to the fact that there are only a fixed number of results that can be obtained from the content based method, and as the basket size increases, the degree of precision falls off. The pure collaboration based method also works in a similar way, but returns more far-fetched results which it gets from the users that are more similar to the current user. Since it tries to correlate wines from seemingly similar users, this has a negative effect on larger basket sizes. The content + collaborative takes the best points from each of the two previous models, and they cancel out the disadvantages. It never runs short of recommendations to make, because its results have a uniform scoring pattern in spite of increasing basket size. As such, the results fall off, but the rate of decrease is rather shallow, as demonstrated in Figure 11.

Cold start was the best performing model, possibly because it generated an expanded user preference list on which the collaboration technique was applied. The fact that it had a high number of candidate wines due to the expansion, and the subsequent correlation introduced through the collaboration technique ensured that its results increasingly improved with larger basket size. The results for the top k1, k2 model followed a similar pattern to the content + collaboration based model. This was expected because this model was just a variation of it, but, with a reduced number of candidate wines. This affected its precision value for the increasing basket size because it did not have enough wines to meet the user's needs.

We hope to improve the accuracy and performance of our models in future implementations of our recommendation system. One way of doing this, is to improve the quality of each wine's associated attributes. We could use an expanded list of attributes, by consulting a glossary of relevant wine tasting terminology[4]. We found that many foreign words were present, and if these were translated into english, then this would have greatly enhanced our term matching process. For example, consider the following word equivalents: *rouge* → red, *blanc* → white. Also, our models made extensive use of various weight parameters, such as α, as described in Section 4. We would like to perform analyses with different values of these weight parameters, and examine the resulting effects. Finally, we would like to use other types of recommendation system models. For example, we have considered a *Prestige*[9] or *PageRank* model, that re-ranks the intermediate list. They use multiple iterations, which re-sort the list each time, giving a higher level of refinement.

8. REFERENCES

[1] Y. Chuan, X. Jieping, and D. Xiaoyong. Recommendation algorithm combining the user-based classified regression and the item-based filtering. In *8th International Conference on Electronic Commerce*, Fredericton, Canada, 2006.

[2] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *ACM Recommender Systems*, Barcelona, Spain, 2010.

[3] E. H. Han and G. Karypis. Feature-based recommendation system. In *ACM Conference on Information and Knowledge Management*, Bremen, Germany, 2005.

[4] A. Hawkins. A glossary of wine-tasting terminology. `http://zebra.sc.edu/smell/wine_glossary.html`, 1995.

[5] H. Ma, I. King, and M. R. Lyu. Effective missing data prediction for collaborative filtering. In *30th International ACM SIGIR Conference*, Amsterdam, The Netherlands, 2007.

[6] S. Park, D. Pennock, O. Madani, N. Good, and D. DeCoste. Naive filterbots for robust cold-start recommendations. In *12th ACM International Conference on Knowledge Discovery and Data Mining*, Philadephia, PA, 2006.

[7] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *10th International World Wide Web Conference*, Hong Kong, China, 2001.

[8] S. Sen, J. Vig, and J. Riedl. Tagommenders: Connecting users to items through tags. In *18th International World Wide Web Conference*, Madrid, Spain, 2009.

[9] Y. Shi, M. Larson, and A. Hanjalic. Connecting with the collective: Self-contained reranking for collaborative recommendation. In *ACM International Workshop on Connected Multimedia*, Firenze, Italy, 2010.

[10] E. Vozalis and K. G. Margaritis. Analysis of recommender systems' algorithms. Technical report, Department of Applied Informatics, University of Macedonia, Thessaloniki, Greece, 2003.