Social Tag-Based Recommendation Services

Jordan Bentley
Advisor: Elke Rundensteiner
Collaborator: Dr. K. Claypool, MIT Lincoln Lab

Worcester Polytechnic Institute
Computer Science
Major Qualifying Project

May 5, 2009

Contents

1 Introduction

1.1 Motivation

Recommendation systems are a staple of Web 2.0. Sites such as Amazon.com or Netflix, for example, use recommendation systems to suggest products to customers. Having recommendations is valuable to these companies and improves customer retention [5].

Social tagging has also become an important part of Web 2.0 [22]. Sites like del.icio.us, flickr, and citeulike have become hubs of information categorized by social tagging. These sites allow users to upload various types of resources (web bookmarks, photos, etc.) and allow the community to add tagged meta data. Users are socially motivated to add tags for organization and communication with other users [2]. We observe that in the cognitive process for creating a tag, users provide a way for others to see what they are interested in [19]. We believe that this can be exploited for developing better recommendation technology due to knowledge of users interests.

1.2 State of the Art

Most current recommendation systems rely on numerical ratings to judge user interest. Netflix, for example, collects a rating of a film on a 1 to 5 scale and then tries to predict how the user would rate other films. The standard algorithm in the field [14], Resnick's algorithm [17], was first published in 1994.

The standard model for recommender systems is a user-based collaborative filtering model. It is generally the most accurate method and is considered the "gold standard" of recommendations [9]. In this model, given a target user, the system first attempts to find other users that are similar to that user. It then takes that pool of similar users and recommends resources from them.

1.3 Problem Statement

Ratings-based algorithms are effective, but they do not have all the information they need. It is not enough to know how much a user liked something; it is probably even more important to know *why* they liked it. Tags can provide this context [12].

When users provide tags, they tend to follow individual patterns rather than imitating the tag choices of other users [16]. We are looking for a method to exploit this information for recommendations.

Tag based algorithms have began to be studied [11, 12] in recent years, but given their infancy it is our hope that we can offer improvements, or at the very least new concepts. In particular, we have focused on the problem of tag sparsity. That is, the problem that many of the tags are may be infrequently.

The English language has more unique words than any other. Not counting scientific terms, there are 500,000 English words [3]. This is made worse when you add in spelling mistakes, names, and various word forms. For example, in our sample dataset, 37.7% of tags entered by users were used 3 or less times through the whole set.

1.4 Our Proposed Approach

Our main method of handling sparsity in tags was through measures of tag co-occurrence. We hypothesized that we could measure the similarity between two tags by looking at how often they are used together.

1.5 Outline

The first part of this paper explains our data model and notation, so that the reader will better understand our explanations. We then discuss methods for the pre-processing of tagged based data. Specifically, we propose a method to unify tags that is similar in function to a spell checker and lastly, we discuss our experimental results with this method.

The second part discusses an actual recommendation algorithm for tagged data. We then go on to discuss our experiment to evaluate the algorithm and our results.

2 Preliminary Data Model and Definitions

2.1 Data Model

In a conceptual model for tag-based data, as seen in Figure 1, each ’record’ contains a User, a Resource, and a Tag. Every time a user tags a resource, a new line is added. There are no resources that have no tags or users, and no users that have no resources or tags. Individually, from a user you can get a list of all tags in linear time, as well as going from a tag to a list of bookmarks that contain it, etc.

2.2 Notation

We have chosen to denote users with lowercase letters, resources with numbers, and tags with capital letters. Additionally, we will denote the set produced by the relationship of two elements by the result element subscripted by the elements producing it. For example, with the data in Figure 1, $TAGS_{a \to 2}$ would return $\{A, B\}$. On the same principle, $RESOURCES_a$ would return $\{1, 2\}$.

User	Resource	Tag
a	1	A
a	2	A
a	2	B
b	2	A
b	3	B
b	3	C

Table 1: User - Resource - Tag Model

2.3 Implementation

All work was implemented in a mix of mySQL and Java. The reason for this was compatibility with the existing Gnizr [21] software. All of the SQL tables we used were compatible with Gnizr, so if a developer wanted to implement our algorithms in the Gnizr software package, it could be done with minimal effort.

3 Pre-Processing

As in most real-world applications, the data received from users is not always perfect. If we were to run a recommendation algorithm on the data as is, we may not produce results as good as we would on better data. Hence, we opted to employ several pre-processing steps to help improve the quality of the data.

3.1 Tag Unification

Data inputted by human users typically will have many misspellings and typos. This is a problem because if a word is misspelled, any simple data mining algorithm would see it as two unrelated terms instead of one.

There are multiple ways of addressing this issue. The first is to check input as it is entered by the user, and prompt users if a mistake is expected. The second, which is discussed in this paper, is to correct the data after it has been collected.

3.1.1 Word Lists

Dictionary Based Methods

Most spell checkers, such as the one found in Microsoft Word, use a predefined dictionary to compare words against. They are extremely effective for casual words processing, but cannot correct words that are not in their dictionary.

Our first attempt at spell checking of tag-based data was with a dictionary. WordNet [10], created at Princeton, was used to test the method. Words found in technical tags such as *readme*, *biorthogonal*, *multihop*, and others were all missing from the dictionary. No satisfactory dictionary could be found for computer related technical terms. Even if one were to be found, it would be difficult to keep it up to date.

As tags related to new technologies or current events are added, our spell checker would likely become obsolete.

Heuristic Method

In order to have a robust solution, we propose a heuristic method. Rather than have a dictionary predefined, it creates a dictionary that contains every word in the initial data set and for each word tries to find the best correction. Algorithm 3.1 shows the outline of how spelling corrections were made.

Algorithm 3.1: SPELL CHECK($allTags$)

```
global $cutoff$
local $original, potentials, bestDistance, bestPotentialChange$
for each $original \in allTags$
  do
    $bestDistance \leftarrow cutoff$
    $bestPotentialChange \leftarrow original$
    for each $potential \in allTags$
      do
        if (DISTANCE($original, potential$) $>$ $bestDistance$)
          then $bestDistance \leftarrow$ DISTANCE($original, potential$)
               $bestPotentialChange \leftarrow potential$
    $original \leftarrow bestPotentialChange$
return ($allTags$)
```

3.1.2 Correction Decision

The decision of whether to make a correction is based on a calculated numeric score of the potential of a change to be an improvement on the data. There are several challenges here, because the correct words are not known at the start.

Because this spell checking is intended for pre processing only, and would not be seen by users, it is not considered important if the changes are actually correct. For example, if the tags *artificial* and *artaficial* are both switched to *artaficial*, which is not the correct word, the change is still deemed a success. What matters is only that all tags that are supposed to mean *artificial* are set to the same value. The method could be consider "tag unification" instead of "spell checking."

As a starting point for calculating the change potential χ, the following assumptions were made about pairs of tags with the same intended meaning:

- They will be spelled similarly, and therefore have a small edit distance.
- Tags will be more likely to appear on the same bookmark compared to other bookmarks.
- Tags which have a longer string length are more likely to contain spelling errors.

Edit Distance

Generally, the distance between the spelling of two words is based on edit distance. Edit distance is a metric calculated by the minimum number of character operations, such as adding or deleting, that are needed to go from one word to the next. Conceptually, it is as if a person with no knowledge of the target language was asked to compare words. For example, take the words *kitten* and *sitting* in Figure 1. Here the edit distance would be 3.

The edit distance was calculated using a library called Jazzy [7]. It is a Java implementation of the

$$\begin{array}{c}\text{kitten}\\ \Downarrow \\ \underline{\text{s}}\text{itten}\\ \Downarrow \\ \text{sitt}\underline{\text{i}}\text{n}\\ \Downarrow \\ \text{sittin}\underline{\text{g}}\end{array}$$

Figure 1: Edit Distance Between Kitten and Sitting

popular C++ Aspell library [4]. The Aspell algorithm uses its own implementation of edit distance that they have found effective. It also can, given a target word, find every word in a dictionary with a small enough edit distance in a short time.

Because of the way Jazzy calculates edit distances, the distances tended to be around 90 to 120. In order to normalize them, a sigmoid function was used. A sigmoid function is commonly defined by $P(t) = \frac{1}{1+e^{-t}}$. In this case, the function was adjusted with transforms to be $1 - \left(\frac{1}{1+e^{\frac{-(t-95)}{5}}}\right)$ where *t* is the edit distance.

Tag Context

Because of the nature of the tag based data, we propose a method using tag context to improve spell checking. In our data model, tags are related to each other by shared bookmarks. This relation can be exploited to find a distance between two tags. This information, combined with string comparisons, can be used effectively to find misspellings.

Context distance is calculated using a Jaccard Index. A Jaccard index is a measure of similarity between sets equal to the magnitude of the intersection of the sets divided by the union. At first the sets were for a given tag the set of all resources it has been used on, but on typos made only once or twice this was ineffective and connections were missed. Instead it uses the set of all bookmarks which use a tag that is used in the set of bookmarks containing the target tag.

3.1.3 Stemming

In tag based data, the tense or pluralization of a word is not important. *Tree* and *trees*, while both spelled correctly, are intended to express the same concept and hence it is desirable to merge them to one tag. Normally this may be done with specialized stemming algorithms for the target language such as the Porter Algorithm [15]. They would put the words into the root form.

The spell check algorithm proposed here however sees these the same way as any other spelling error and corrects some of these. Spell check accuracy could possibly be improved by performing stemming operations before or after running our spell check algorithm.

3.1.4 Other Considerations

Words with length less than 4 had to be ignored. Running the spell checker without doing this causes a high rate of error. There are many pairs of short words with small edit distances, such as *find* and *fund* which are not related. Perhaps special rules could be applied to shorter words to be able to include them, but it was not done for this study.

Further, all tags without alphabetic characters had to be removed from consideration. In practice it is not likely that many tags will fit this description, but there are some cases where this is an issue, such as with dates.

Other methods were attempted for improving spell checking. On earlier attempts, the correction score was weighted by the frequency of the potential correction. The intuition was that if a word occurred more often, it was more likely to be correct, but when this weight was removed, a large reduction in measured error rate was observed. With frequency weighting, only 750 words (3.2% of all tags) could be corrected with 99% accuracy instead of 1700 (7.4% of all tags). This is shown in Figure 2.

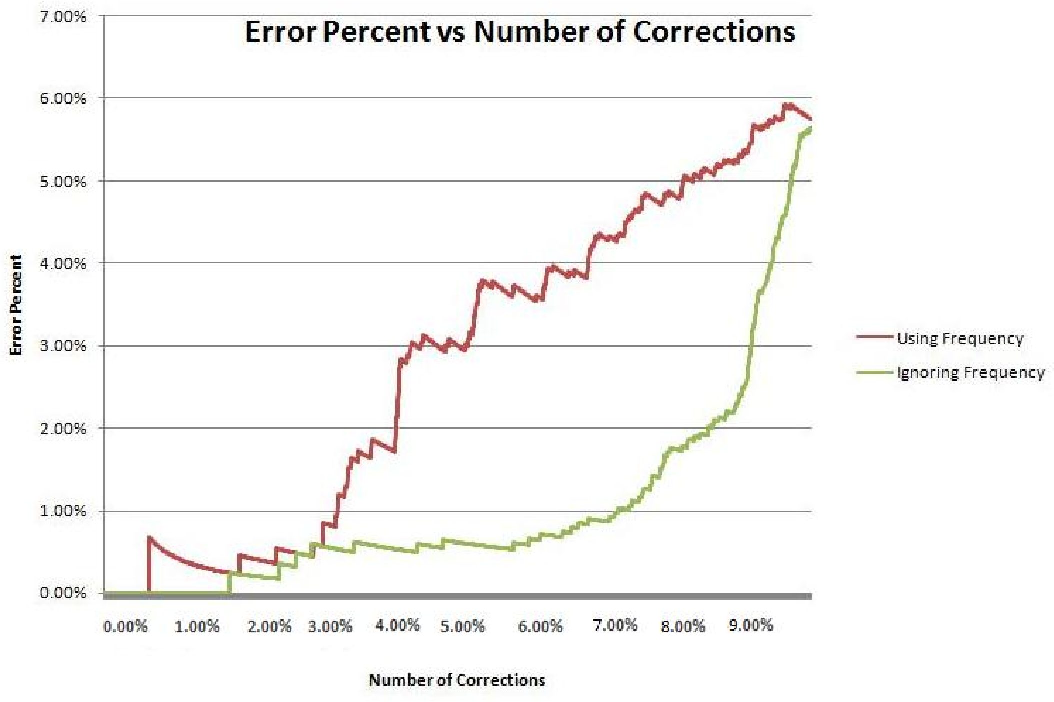

Figure 2: Effects of Frequency Weighting

3.1.5 Evaluation

Evaluation of the spell check effectiveness has to be done manually. A text file can be created containing a list of correct word pairs and incorrect word pairs, each created manually. The testing software parsed these files and used them to generate statistics on its output, so that if changes to the algorithm need to be tested, no redundant manual checking had to be done.

Success was measured by ordering the corrections by score, so that the most likely correct corrections were first. Currently more research needs to be done on selecting a cutoff point when putting spell checking into practice.

In practice, a cutoff point would be determined based on the quality of the data and the needs of data mining methods to be used after spell checking. For example, with this data if the requirement was to have less than 1% error, the cutoff score would need to be

Figure 3 shows the effectiveness of this spell checking method. On the sample dataset 7.4$\%$ tags could be corrected with 99% accuracy.

Normally, the complexity would be $O(n^2)$. When implemented with Jazzy, however, the spell checker runs efficiently. Jazzy is able to quickly weed out unlikely matches, leaving only a few candidate words. The time complexity is nm, where m is the average number of candidates generated for each word by Jazzy. The average number of candidates is dependent on how close the words in the dataset are. In the worst case, for a set of similar words, the complexity would be $O(n^2)$, and in the best case where none of the words are similar it would run in $O(n)$. In practice, most tags only had to be checked against less than 10 candidates.

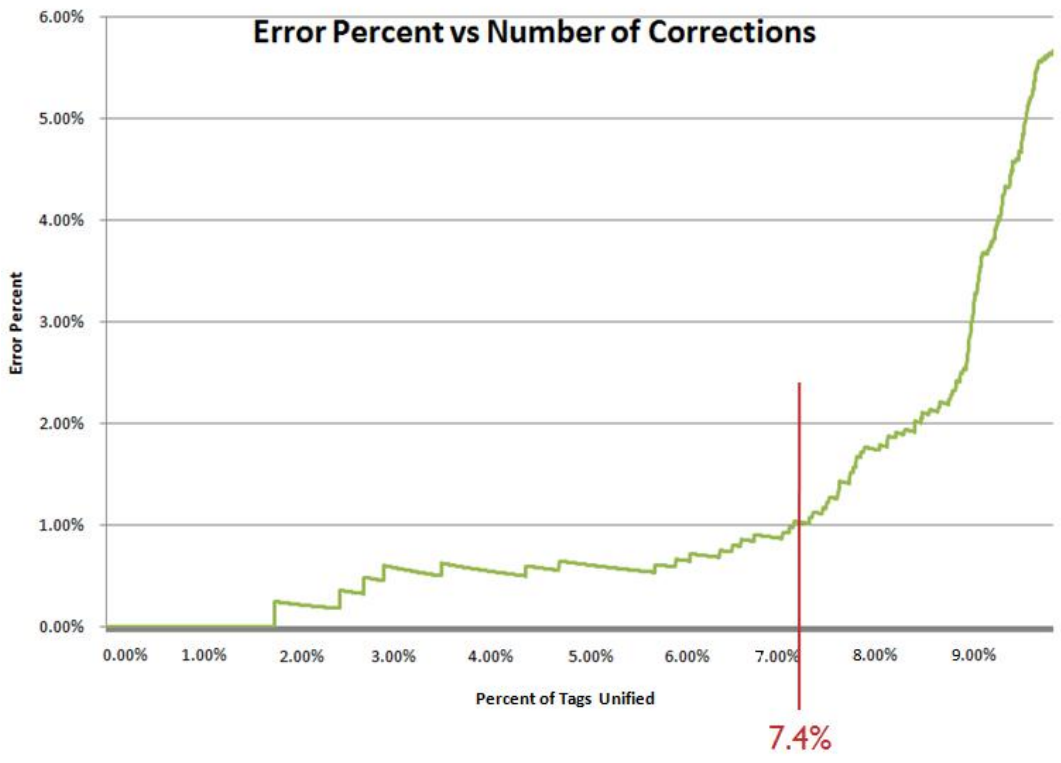

Figure 3: Spell Check Results

4 Collaborative Filtering

The “gold standard” of collaborative filtering algorithms is the user-user model [9]. The idea is simple: first find users whose choices are close to your target users choices, then use their resources to generate a recommendation.

4.1 Finding Like Users

Traditionally, collaborative filtering algorithms assume that users give some kind of rating to their resources. In our case, however, we are using a dataset from Citeulike where there is only tag data and no ratings. We thus would like to exploit this tag data to find users who have similar tagging behaviors.

4.1.1 User Similarity

In order to calculate the similarity between two users, we need to compare the sets of the tags they used on shared resources

First we must create a search space that contains every user that shares resources with the target user. Then for each resource shared with each user, we compute the dot product of the tag vectors for those users on that resource normalized by dividing by the product of their magnitudes. In the dot product, we let the multiplication operation be tag similarity.

We take the summation of these values, then multiply this sum by the log of the number of resources they share in common divided by the number shared. This way, users with more shared resources will get a higher similarity rating, but not by an overwhelming amount.

$$sim_{a,b} = \sum_{i=1}^{n} sim_{TAGS_{a\to i},TAGS_{b\to i}} \times \frac{\ln(n+1)}{n}$$

4.1.2 Tag Vector Similarity

In order to find the distance between two users, we need to be able to compare the way that they tagged resources. We do this with the tag vector similarity, which is the comparison of the sets of tags that two users used on a similar resource [11]. To calculate it, we

$$sim_{TAGS_{a\to 1},TAGS_{b\to 1}} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} sim_{TAGS_{a\to 1}[i],TAGS_{b\to 1}[j]}}{n \times m}$$

For example, if user a tagged resource 1 with tags A, B and user b tagged resource 1 with tags B, C then the similarity would be $\frac{sim_{A,B}+sim_{A,C}+sim_{B,B}+sim_{B,C}}{2\times 2}$.

4.1.3 Tag Similarity

One proposed method of comparing users is to see how many shared resources they tagged with the same tag [12]. One issue with this is the sparsity of the tags used. Two users may be interested in a resource for similar reasons, but may use different tags to express that interest.

The following solution is proposed: Instead of checking to see if tags are identical, we propose to use some measure to compute the distance between them. While many different measures of tag similarity could be used, in our current work, the distance between two tags is based on the co-occurrence on resources. We propose that tags that are more likely to appear together on a resource are more likely to be related.

Tag similarity is where we attempt to compensate for the scarcity problem. Rather then a binary comparison where if the two tags are the same we return 1 and if they are not return 0, we propose a measure based on co-occurrence in resources. If two tags have been used together frequently, they are said to be closer. This is related to the measure used in the spell checking algorithm above, and the Jaccard index is used again. The distance between two tags is the number of resources which contain both tags divided by the number of resources which contain either tag. The values of similarity range from 0 to 1, where 0 is unrelated and 1 means that the tags are identical.

$$sim_{tagA,tagB} = \frac{|Resources_A \cap Resources_A|}{|Resources_A| + |ResourcesB|}$$

Say we have tag A and tag B. $RESOURCES_A = \{1, 2, 3\}$ and $RESOURCES_B = \{2, 3, 4\}$. The similarity would then be $\frac{1}{3}$.

$$sim_{tagA,tagB} = \frac{|\{1,2,3\} \cap \{2,3,4\}|}{|\{1,2,3\}| + |\{2,3,4\}|} = \frac{|\{2,3\}|}{3+3} = \frac{1}{3}$$

4.2 Recommendations

Now that the user similarities have been found, recommendations must be generated for user a. This is done over the set of all resources tagged by all users where $sim_{a,b} > 0$ and $resourcen \notin RESOURCE_a$.

In order for this data to be meaningful, the set of recommendations should be ranked. We propose that to do this, each resource receives a score equal to the sum of user similarities between the user we are generating recommendations for and users who have tagged the resource we are computing a score for. While it is possible to generate recommendations in a way that takes tag context into account, this has already been accounted for in the user similarity formula, and repeating it in this step would narrow the breadth of recommendations. The goal is to recommended papers that are not necessarily the closest to those already bookmarked by the user, but to recommend resources that the user might like.

$$recommendationRanking_{a \to 1} = \sum_{i=1}^{n} sim_{a,i} \times \frac{\sqrt{n}}{n}$$

Consider the example in Table 2. It shows a list of users and whether they have tagged resource 1 as well as their similarity to user a. To find $recommendationRanking_{a \to 1}$, we find $(0.6 + 0.3 + 0.1) \times \frac{\sqrt{3}}{3} = 0.577$.

User	Tagged 1?	$Sim_{a,u}$
a	y	0.6
a	y	0.3
a	n	0.2
b	y	0.1
b	n	0.8
b	n	0.6

Table 2: User Similarity

4.3 Automatic Evaluation

Because the quality of a recommendation is a subjective measure, it is difficult to evaluate the effectiveness of a particular algorithm without user input. We propose the following method for automatic evaluation, which provides a rough estimate of effectiveness.

1. Select user k at random, and from that user select resource n at random.
2. Remove resource n from user k, including all tags in $k \rightarrow n$.
3. Generate ordered recommendations for user k.
4. If resource n appears in the recommendations, record the position on the list.

We ran this method on a sample dataset obtained from citeulike.org. [6] The dataset contained 115,458 unique resources, 23,133 unique tags, and 3,567 users. Over a set of 1000 iterations, the results showed that 24% of removed resources were in the top 50 recommendations, or the top 0.04%. 50% were in the top 270, or the top 0.23% and 95% were in the top 2000, or the top 1.73%.

4.4 User Study

4.4.1 Method

We created a survey using SurveyMonkey [20] which consisted of 61 academic papers. These papers were handpicked from the citeulike dataset previously mentioned [6], and contained a variety of popular computer science related topics. They were chosen in such a way that subjects generally familiar with computer science, would be able to understand all of the resources and find ones that interested them.

For each paper, the user was presented with the title and abstract, as well as a link to the paper itself. Each user was asked to rate the papers on a scale of 1 to 5, with 1 being the lowest and 5 being the highest. The instructions stated that a rating of 4 implied that they found it interesting enough that they would want to share it with other users if they were on a social tagging site. They were then instructed to provide tag data for everything that they rated 4 or higher.

The survey was distributed to graduate and undergraduate computer science students at Worcester Polytechnic Institute through emails and fliers. Cash prizes were offered as an incentive.

In total, 52 users responded and filled out the survey in full. Because in practice collaborative filtering algorithms are intended to run on large datasets, we added our collected data to the citeulike data and noted the ratings the users in the survey had given.

4.4.2 Evaluation

The method for analyzing this data was similar to the previous evaluation method in section 4.3.

1. Select user k at random from the set of users that conducted the survey, and from that user k select resource n at random from the set of items with ratings given by user k. This included both items they had tagged and items that they had not tagged.
2. If user k had tagged resource n, remove resource n from user k so that it is as if user k had never tagged resource n.
3. Compute recommendation $score_{k \to n}$.
4. Record the recommendation score for n to user k as well as the score user k provided in the survey.

After this we had pairs of values of the computer recommendation score and the normalized user score. The traditional method of evaluating collaborative filtering algorithms is to take the error between these two values [8]. Because our algorithm is not rating based, however, we need to address the issue that the values outputted by it are not on the same range of values as the user given scores.

Normally, computing the error for pairs of values is equivalent to plotting them on a graph and finding the distance on the y axis between the points and the line $y = x$.

To still be able to calculate the error with our method, we propose to use regression methods to translate the algorithm output to the user scores.

Ideally, the ratings we generate will directly correlate to the ratings the users originally gave the resources. For this, consider a graph with the user generated score on the x-axis and the algorithms score on the y-axis.

Ideally, there would be a linear relationship between our computed ratings and the ratings supplied by the user. Since this is not necessarily the case, we calculated the line of best fit for this data using various regression methods. The resulting line is a function to transform our algorithms output to something that more closely corresponds to the users ratings.

In practice, these ratings would later be turned into orderings. Because of this, it does not matter if the line of best fit is coded into our algorithm, as long as it preserves the order, it is inconsequential.

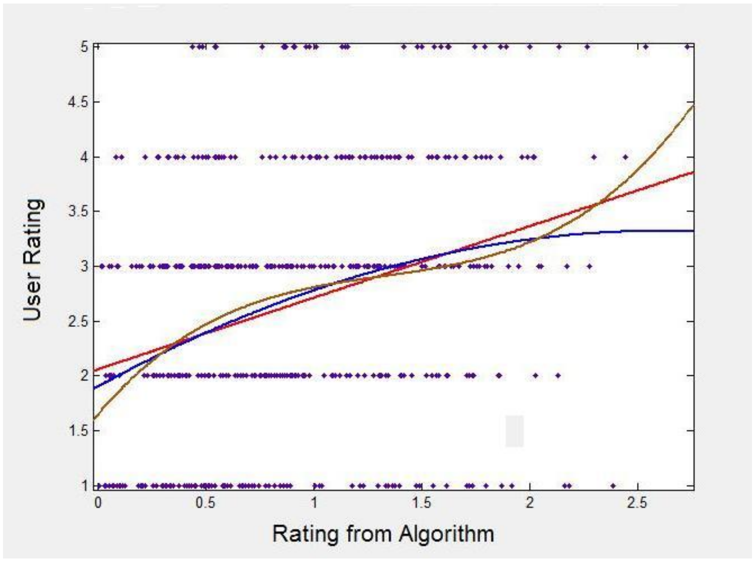

Figure 4: Potential Lines of Best Fit

We computed lines using cubic, quadratic, and linear regression among others. The resulting lines are shown in Figure 4. From these, we selected the cubic regression, which was the best fit for the data. The generated equation was $y = 0.3926x^3 - 1.55x^2 + 2.317x + 1.647$ and can be seen in Figure 5.

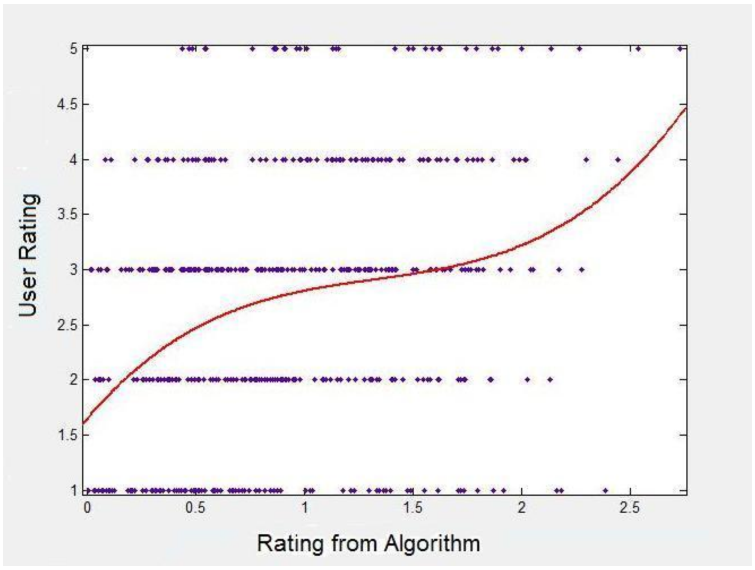

Figure 5: Cubic Regression

Our first calculation was the Mean Absolute Error. This is the average of the absolute values of the errors $\frac{\sum |x_i - y_i|}{n}$. The resulting value was 0.909, meaning that on average, our ratings predictions were off by less than one ratings level.

We then calculated the *Root Mean Square Error* for the data. This method is often used in evaluation of collaborative filtering methods [18]. It is calculated as the square root of the average squared error which is computed by $\sqrt{\frac{\sum_i^n = 1(x_i - y_i)^2}{n}}$.

The RMSE was calculated to be 1.105. The fact that this is close to the Mean Absolute Error implies that there not only was the average error low, but there were few large errors.

Because our dataset does not contain many ratings, namely, only for the users who participated in our study, we did not run other algorithms on the same dataset. Collaborative filtering algorithms under-perform significantly on small data sets [1]. We could, however, draw some information by comparing against rating based algorithms run on other datasets that used a similar 1 to 5 rating scale.

Netflix's ratings based Cinematch algorithm has been run on a much larger dataset than ours, having 1.9 billion ratings [5]. On that dataset, their algorithm runs with a RMSE of 0.9525 over all users [13].

5 Conclusion

5.1 Accomplishments

We have created both a tag unification system and a method for generating tag based recommendations. We have shown that tag co-occurrence is an effective way of comparing tags in a system of tagged resources. We have also presented a method by which a tagged based recommendation system can be compared with a rating based recommendation system.

Tag based recommendations are young in comparison to rating based methods. Rating based algorithms have been studied extensively for decades [17]. Although right now tag based methods are not as well developed as rating based methods, our results show promise. We were able to predict a users score within 1 rating, and when trying to recommend a resource known to be of interest, it was in the top 1.73% 95% of the time.

5.2 Future Work

For the tag unification system, work is still needed to determine the best cutoff points to make changes. Analysis could also be done on the merit of applying conventional stemming and spell checking algorithms before or after running the tag unification. There are also other ideas that could be incorporated into the system, such as estimating user expertise by the detail of their tagging vocabulary.

There is still much to do in the field of tag based recommendations. Tag co-occurrence has been shown to be effective, but we would like to improve our recommendation algorithm. More work could be done to find a method that takes tags in to account again when generating recommendations from user similarity scores.

More analysis is also needed. Although we created a dataset that contained both tags and ratings, it was small. A larger dataset would be useful not only to our work but to others working on tag based methods. We also propose that a hybrid method could be created that takes both ratings and tags in to account.

Several social tagging websites already exist without recommendation services, and it is our hope that through further research they will have robust methods at their disposal to increase their value through tag based collaborative filtering.

References

[1] Hyung Jun Ahn. A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem. *Inf. Sci.*, 178(1):37–51, 2008.

[2] Morgan Ames and Mor Naaman. Why we tag: motivations for annotation in mobile and online media. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 971–980, New York, NY, USA, 2007. ACM.

[3] Michael Angier. Success strategies, May 2009. `http://www.sfpnn.com/Michael/Michael2002/ma05200`

[4] Kevin Atkinson. Gnu aspell, May 2009. `http://aspell.net/`.

[5] James Bennett, Stan Lanning, and Netflix Netflix. The netflix prize. In *In KDD Cup and Workshop in conjunction with KDD*, 2007.

[6] Citeulike. Citeulike: Everyones library, May 2009. `http://www.citeulike.org/`.

[7] Mindaugas Idzelis. Jazzy, May 2009. `http://jazzy.sourceforge.net/`.

[8] Matthew R. McLaughlin and Jonathan L. Herlocker. A collaborative filtering algorithm and evaluation metric that accurately model the user experience. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 329–336, New York, NY, USA, 2004. ACM.

[9] Sean M. McNee, Nishikant Kapoor, and Joseph A. Konstan. Don't look stupid: avoiding pitfalls when recommending research papers. In *CSCW '06: Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, pages 171–180, New York, NY, USA, 2006. ACM.

[10] George A. Miller. Wordnet, February 2009. `http://wordnet.princeton.edu/`.

[11] Reyn Y. Nakamoto, Shinsuke Nakajima, Jun Miyazaki, and Shunsuke Uemura. Tag-based contextual collaborative filtering. 2007.

[12] Reyn Y. Nakamoto, Shinsuke Nakajima, Jun Miyazaki, Shunsuke Uemura, Hirokazu Kato, and Yoichi Inagaki. Reasonable tag-based collaborative filtering for social tagging systems. In *WICOW '08:*

Proceeding of the 2nd ACM workshop on Information credibility on the web, pages 11–18, New York, NY, USA, 2008. ACM.

[13] Netflix. Netflix prize rules, May 2009. `http://www.netflixprize.com/rules`.

[14] John O'Donovan and Barry Smyth. Is trust robust?: an analysis of trust-based recommendation. In *IUI '06: Proceedings of the 11th international conference on Intelligent user interfaces*, pages 101–108, New York, NY, USA, 2006. ACM.

[15] Martin Porter. The porter stemming algorithm, May 2009. `http://tartarus.org/ martin/PorterStemmer/`.

[16] Emilee Rader and Rick Wash. Influences on tag choices in del.icio.us. In *CSCW '08: Proceedings of the ACM 2008 conference on Computer supported cooperative work*, pages 239–248, New York, NY, USA, 2008. ACM.

[17] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *CSCW '94: Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186, New York, NY, USA, 1994. ACM.

[18] Badrul Sarwar, George Karypis, Joseph Konstan, and John Reidl. Item-based collaborative filtering recommendation algorithms. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 285–295, New York, NY, USA, 2001. ACM.

[19] Rashmi Sinha. A cognitive analysis of tagging, May 2009. `http://rashmisinha.com/2005/09/27/a-cognitive-analysis-of-tagging/`.

[20] SurveyMonkey. Surveymonkey.com, May 2009. `http://www.surveymonkey.com/`.

[21] ybishr. Gnizr, May 2009. `http://code.google.com/p/gnizr/`.

[22] Daniel Zeng and Huiqian Li. How useful are tags? – an empirical analysis of collaborative tagging for web page recommendation. In *PAISI, PACCF and SOCO '08: Proceedings of the IEEE ISI 2008 PAISI, PACCF, and SOCO international workshops on Intelligence and Security Informatics*, pages 320–330, Berlin, Heidelberg, 2008. Springer-Verlag.