Tagommenders: Connecting Users to Items through Tags

Shilad Sen
Macalester College
ssen@macalester.edu

Jesse Vig
Grouplens Research
University of Minnesota
jvig@cs.umn.edu

John Riedl
Grouplens Research
University of Minnesota
riedl@cs.umn.edu

ABSTRACT

Tagging has emerged as a powerful mechanism that enables users to find, organize, and understand online entities. Recommender systems similarly enable users to efficiently navigate vast collections of items. Algorithms combining tags with recommenders may deliver both the automation inherent in recommenders, and the flexibility and conceptual comprehensibility inherent in tagging systems. In this paper we explore tagommenders, recommender algorithms that predict users' preferences for items based on their inferred preferences for tags. We describe tag preference inference algorithms based on users' interactions with tags and movies, and evaluate these algorithms based on tag preference ratings collected from 995 MovieLens users. We design and evaluate algorithms that predict users' ratings for movies based on their inferred tag preferences. Our tag-based algorithms generate better recommendation rankings than state-of-the-art algorithms, and they may lead to flexible recommender systems that leverage the characteristics of items users find most important.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information Filtering*; H.5.3 [**Information Interfaces and Presentation**]: Group and Organization Interfaces—*Collaborative computing*

General Terms

Algorithms, Experimentation

Keywords

tagging, recommender systems, collaborative filtering

1. INTRODUCTION

Recommender systems enable users to navigate vast collections of items. Amazon suggests products users may like based on their ratings, clicked items, and purchased items [17]. Users of Digg receive news articles based on other articles they find interesting [25]. Members of Netflix receive movie recommendations based on their movie ratings [3]. In each of these scenarios, recommender systems choose a few items a user will like most from among thousands, or even millions, of possibilities. This task, which we call *recommend*, is one of two tasks supported by nearly all recommender systems [26]. For the second common task, *predict*, recommender systems predict which rating a user will assign to a particular item. For example, a user of "Rate Your Music"[1] might receive a predicted rating of 4.2 out of 5 stars for the album "White Blood Cells" by the White Stripes based on a five star rating for "In Rainbows" by Radiohead. For both the recommend and predict tasks, recommender systems help users understand an unknown relationship between themselves and an item by comparing a user's behavior (e.g. album clicks and ratings) to patterns of behavior in other users.

Tagging systems offer users an alternate way to address the recommend and predict tasks. Shirky suggests that since tags are created by users, they represent concepts meaningful to them [31]. Because tags are easily comprehended by users, tags serve as a bridge enabling users to better understand an unknown relationship between an item and themselves. In previous work, we validated this relationship by finding that certain types of tags help users to find and make decisions about items [29]. For example, Alice[2] is a real user in the MovieLens movie recommendation community we study. She enjoys animated movies, and has assigned five star ratings to "Shrek," "Pinnochio," and "Toy Story". If Alice visits the web page for the movie "Ratatouille" she would see that 5 users have applied the tag *animated* to it. Based on these tags, she might decide she would enjoy the movie (the predict task). Alice might then click on the tag *pixar* to discover the related movie "The Incredibles" (the recommend task).

Recommender algorithms that incorporate tagging information promise to combine the best elements of both types of systems. Lamere refers to these tag-based recommendations as "tagomendations" [16]. We similary refer to tag-based recommender systems as *tagommenders*. Tagommenders offer the automation of traditional recommender systems, but retain the flexibility of tagging systems. Schafer et al. found that users enjoy specifying feedback about items along a variety of dimensions [27]. Tagommenders enable recommenders to use the dimensions of items that users consider most important.

In this paper, we design tagommenders inspired by the way in which humans use tags to evaluate items. Figure 1 presents the model we explore in the movie recommendation domain in this paper. The bottom of the figure shows that traditional recommender systems infer users' preferences for movies based on their movie ratings. The top half of the figure describes the two main components of tagommender algorithms. First, we infer users' preferences for tags based on their interactions with tags and movies. Second, we infer users' preferences for movies based on their preferences for tags.

We define a user's preference for a tag as the user's level of interest in movies exhibiting the concept represented by the tag. For example, Alice indicated that she likes *animated* movies and *swashbucklers*, but dislikes movies about *serial killers*.[3] Her (dis)interest

WWW 2009, April 20–24, 2009, Madrid, Spain.
ACM 978-1-60558-487-4/09/04.

[1] http://www.rateyourmusic.com
[2] Alice is a pseudonym to protect the user's privacy
[3] We learned Alice's preferences for tags through a survey we de-

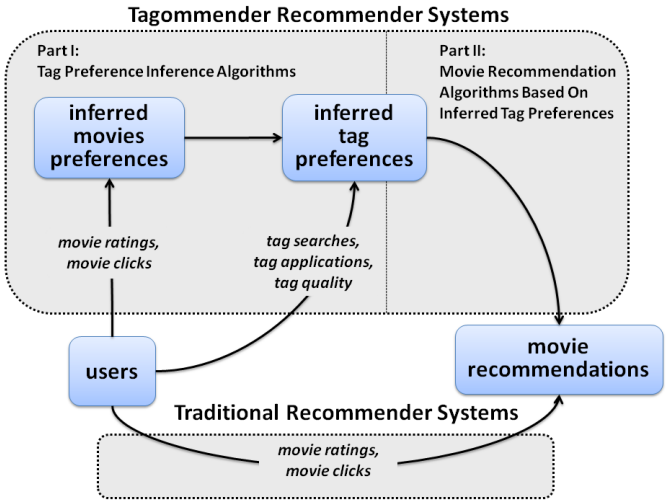

Fig. 1: *Our model of movie tagommenders. Traditional recommender systems (bottom) generate predictions for movies based on movie ratings or clicks. Tagommenders (top) first infer users' preferences for tags (upper left). Based on these inferred preferences for tags, tagommenders generate movie recommendations (upper right). Users' preference for tags can be inferred based on signals of interest in tags (tag applications, searches), or signals of interest in items (movie ratings, clicks). In order to use item signals to infer users' preferences for tags, they must be translated to tag signals (upper left).*

in these concepts may have influenced her 4.5 star rating for "The Mask of Zorro," and her 1.5 star rating for "Hannibal." In the first half of our paper we explore algorithms that infer users' preferences for tags:

RQ1: Can we infer users' preferences for tags?

We consider *tag preference inference* algorithms that analyze *signals* of a user's interest in a tag or movie (Figure 1, upper left). For instance, Alice's application of the tag *shipwrecked* may suggest that she is interested in *swashbucklers* (a signal of tag interest). In addition, Alice's rating of 4.5 stars for "The Mask of Zorro" and her click on a hyperlink leading the movie "The Pirates of Penzance" may also have been a result of her liking for *swashbucklers* (signals of movie interest).

One other signal of tag preference we consider is a tag's *quality.* We define a tag as high quality if it helps the community understand an important aspect of an item. For instance, Alice considers the tags *serial killer* and *animated* as high quality tags that capture important movie concepts but she considers *sure thing* as a low quality tag.[4] Since high quality tags capture important movie concepts, we evaluate an algorithm that infers users' preference for a tag based on the tag's quality.

We evaluate eleven tag preference inference algorithms using 118,017 tag preference ratings collected in a user survey on the MovieLens movie recommender website.

In the second half of this paper we analyze algorithms that predict users' ratings for movies based on their preferences for tags (upper right of Figure 1). We separate our algorithms by the type of signals they use: *implicit* only, or both implicit and *explicit*. Implicit signals such as clicks and searches occur during users' natural interactions with tags and items. Tagommenders for sites that do not support item ratings, such as the online bookmarking site Delicious[5], must generate recommendations based on these implicit signals. Other systems with tags, such as Amazon, support explicit signals of interest in the form of item ratings. Our last two research questions explore the performance of tagommenders in both types of systems.

RQ2: How well do tagommenders perform in systems without ratings?

RQ3: How well do tagommenders perform in systems with ratings?

We evaluate RQ2 and RQ3 using movie ratings and tags created by MovieLens users. Our work offers three contributions to researchers and practitioners:

- We develop and evaluate algorithms that infer users' preferences for tags.
- We develop tag-based recommendation algorithms that infer users' preferences for movies based on their inferred preferences for tags.
- We evaluate the end-to-end predictive performance of tagommender algorithms that combine tag preference inference algorithms with tag-based recommenders.

We believe this work to be important for two reasons. First, we hope that sites with an abundance of tagging activity, such as Delicious or flickr[6], can use our algorithms to improve item recommendations. Second, we believe that tagommenders offer a flexible and comprehensible alternative to traditional recommender systems.

2. RELATED WORK

Many first generation recommenders such as the GroupLens [24] Usenet recommender employed *user-based* algorithms: given a particular user, recommend movies that similar users like. Sarwar et al.'s *item-based* algorithm transposed this model: predict users' ratings for an item based on their ratings for similar items. During the Netflix Prize, two trends emerged in recommender systems research [3]. First, researchers adopted Simon Funk's[7] singular value decomposition algorithm (SVD) due to its accuracy, efficiency, and ease of implementation [8]. Second, researchers such as Bell et al. combined the output of multiple recommender algorithms to improve performance [2] . Our research differs from these collaborative filtering algorithms by using tags as intermediary entities.

Collaborative filtering (CF) algorithms such as the user-based, item-based and SVD algorithms rely on patterns between user ratings, but do not use data about items. They do not, for instance, know that "Toy Story" is an animated movie. Balabanovic et al. were among the first researchers who investigated *content-based* systems that make use of the data about an item such as a movie's genre. Other researchers have studied methods for combining collaborative filtering with content-based systems [22] . Our research extends existing techniques for content-based recommendation in two ways. First, since tags are maintained by community members instead of expert editors their quality varies [29]. We explore how estimates of tag quality improve recommender performance. Second, unlike earlier content-based algorithms, we automatically learn relationships between tags and movies based on inferred tag preferences and movie ratings.

scribe in Section 3

[4]We learned Alice's views about tag quality through the thumb-based quality ratings we describe in Section 3

[5]http://del.icio.us

[6]http://www.flickr.com

[7]Simon Funk is a pseudonym for Brandyn Webb. Since researchers have consistently referred to him by his pseudoname, so do we.

Our algorithms that translate signals of item interest to signals of tag interest build on existing work in user profile extraction for content-based recommenders [1] and web-based systems [21]. However, our algorithms face challenges not present in other domains. While other systems' data are created by domain experts, tags are created by ordinary users. Koutrika et al. suggest that this transfer of control allows tagging systems to be "threatened" by both "malicious" and "lousy" taggers [15]. We differ from previous work on profile extraction by designing algorithms that are robust to differences in tag quality.

Although public bookmarking systems such as Fab [1], and Pharos [4] have been available since the 1990's, Millen et al. point to tagging as a key reason current social bookmarking systems have enjoyed greater success [19]. In early academic research on tagging communities, MacGregor and McCullogh [18] explore the relative merits of controlled versus evolved vocabularies, arguing that evolved ontologies engage users but lack the precision of their controlled counterparts. In earlier work, we show that the tags a user sees influence the tags they create themselves [29]. We also classify tags as generally factual, subjective, or personal (intended for the tag creator themselves), and find that users generally prefer factual tags over subjective tags and strongly dislike personal tags. Our work furthers these studies of tagging communities by analyzing how tags can be incorporated into recommender systems.

In earlier work we explore user interfaces that help systems determine a tag's qulity [28]. In offline results we find that thumb rating feedback significantly improves a system's ability to identify good tags compared to simple implicit signals of tag quality. We verify our results using an online study in the MovieLens movie recommender system [30]. We extend this work by using tag quality (among other signals) to infer users' preferences for tags.

Several researchers have explored algorithms that recommend tags for an item [29] [13]. Hayes et al. examine how tags can be used to cluster bloggers and posts and suggest that tags can be used as a gold standard for cluster coherency [10]. Brooks et al. propose hybrid algorithms drawing on both blog tags and blog text to accurately cluster blogs [5]. We build on this work by providing an end-to-end algorithm that infer preferences for tags and generates movie recommendations based on those preferences.

Three researchers have directly studied tag-based recommenders. In a blog post, Lamere suggests a metric for identifying similar sets of items in tagging sites by measuring the cosine similarity of the tags applied to items [16]. Diederich et al. describe an exploratory study in which users create tag profiles corresponding to their interests and receive recommendations based on those tag profiles [7]. Niwa et al. propose a cluster-based algorithm for recommending web-pages based on the pages users have tagged, and the tags applied to web pages [20]. All three researchers base their recommendations on the similarity of TF-IDF tag profile vectors. We extend this existing research in three main ways. First, we investigate 11 different signals of a user's interest in tags, including tag searches, and item ratings. Second, we explore five different algorithms for calculating item preferences based on tag preferences. Third, we conduct an empirical evaluation using 118,017 star-ratings of tag preference and 1,720,390 star-ratings of item preference.

3. EXPERIMENTAL DATASETS

We conduct our analyses using data collected from the MovieLens website. MovieLens primarily serves as a movie recommender system. Users receive movie recommendations in exchange for rating movies on a five star scale. MovieLens was created in 1997 and maintains an active base of approximately 1,200 users per week. We conduct our analyses using five sets of data from MovieLens

Fig. 2: *Tags as they appear on the MovieLens search results screen.*

Fig. 3: *Tags as they appear on the MovieLens movie details screen.*

described in Table 1. We now describe the data contained in each dataset along with details not specified in the table.

Movie Ratings: MovieLens users rate movies on a one to five star scale.

Movie Clicks: We logged clicks on links to detailed information about a particular movie for approximately 17 months starting in December 2006.

Tag Applications: MovieLens members can tag movies, and use tags contributed by others in the community to find and evaluate movies. MovieLens users most commonly interact with tags through the search results screen (Figure 2) and movie details screens (Figure 3). The movie details screen displays movie information including up to 30 of a movie's tags. Since we introduced tagging features to MovieLens in January 2006, MovieLens members have created 84,155 tag applications resulting in 13,558 distinct tags (a *tag* is a particular word or phrase, a *tag application* is a three way relationship between a user, tag, and item). Further details of MovieLens and the MovieLens tagging system can be found in [29].

Tag Searches: Tag searches are textual searches for tags, or clicks on tag hyperlinks. 1,000 users have searched for at least five distinct tags. 107 users have searched for at least 50 distinct tags.

Tag Preference Ratings: In our model for tag-based recommendation, we first infer users' preferences for tags. In order to evaluate our tag preference inference algorithms, we conducted a survey of tag preferences for MovieLens users. We emailed invitations to 8,361 active users. 995 users responded (11.9% response rate).

In the survey, we showed each user a collection of tags, and asked them to "estimate how much" they "would like movies with each tag using a one to five star scale, or unsure." We asked each user to complete at least 60 tag ratings, but gave them the option to complete more if they wished. In total, users supplied 118,017 ratings for 9,889 distinct tags (mean 117 tags per user, median 78). 800 users completed the requested 60 ratings, while seven provided more than 1,000 ratings.

The breakdown of the survey responses by star ratings is as follows: 6% (7,641) were 5 stars, 17% (20,597) were 4 stars, 22% (26,499) were 3 stars, 13% (15,135) were 2 stars, 13% (15,155) were 1 star, and 28% (32,990) were unsure. The average tag rating was 2.89. The unsure rating was used differently by different users. Among the 25% of users who used the unsure rating most often, unsure ratings accounted for 43% of ratings. Among the 50% of users who used the unsure rating least often, unsure ratings accounted for only 18.3% of ratings.

Pruning: Although we draw on all data when analyzing tag preference inference algorithms in Section 4, we pruned the data sets

Table 1: *Size of different datasets we use in this paper. Count is the number of the entities the dataset contains. Num-users is the number of users that generated those entities. For example, the first two columns in the third row indicate that 84,155 tags have been applied by 3,582 users. The last two columns indicate the same numbers after the pruning we apply for our analyses in the second half of this paper.*

	before pruning		after pruning	
dataset	**count**	**num-users**	**count**	**num-users**
movie ratings	15,395,368	162,556	1,720,390	5,637
movie clicks	552,078	11,997	343,711	5,637
tag apps	84,155	3,582	65,229	2,105
tag searches	48,031	3,314	31,148	1,968
tag pref ratings	118,017	995	n/a	n/a

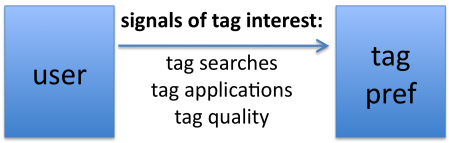

Fig. 4: *Inferring a user's preference for a tag based on her direct interactions with a tag such as her searches for a tag and her applications of a tag.*

for our analyses of tag-based recommendations in section 5. In order to reduce the computational requirements of our analyses, we focused on a set of movies with a minimum threshold of tags, and a set of users with a rich profile of MovieLens behavior.

We began pruning the tag-recommendation dataset by selecting movies that had been tagged with at least five distinct tags. We wanted to focus on tags that represented concepts applicable to multiple movies, so we required that each tag be applied to at least five movies. We iteratively repeated this pruning until we reached a stable set of movies and tags.

After movie ratings, movie clicks are the most abundant source of behavioral information we have for MovieLens users. Since we wanted to explore the effectiveness of tag-based recommendations for domains without tag ratings, we only included users that had clicked five or more movies.

After pruning, 1,720,390 ratings remained from 5,637 users for 2,636 movies. Since applying a tag may indicate interest in a tag, we also track tag applications created by users in the pruned set. In total, 1,315 users in the pruned set applied 50,060 tags (mean of 38 tags per user, median of 2). More statistics are shown in Table 1.

4. INFERRING TAG PREFERENCE

In this section we address RQ1:

RQ1: Can systems infer users' preferences for tags?

We consider two approaches to inferring tag preference. First, algorithms can directly infer a user's preference for a tag based on her direct interactions with the tag (Figure 4). For example, if Alice searches for *animation*, she is probably interested in it. Second, an algorithm may indirectly infer a user's preference for a tag based on her interactions with items having the tag (Figure 5). For example, Alice has assigned five-star ratings to three movies tagged with *animation*: "Shrek", "Pinnocchio", and "Toy Story". Based on these movie ratings, we may infer that she would enjoy other movies tagged with *animation*.

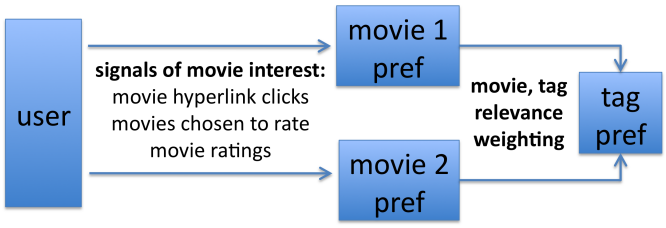

Fig. 5: *Inferring a user's preference for a tag indirectly based on her interactions with items having a tag such as her rating of items with the tag.*

4.1 Inferring Preference using Tag Signals

We consider three algorithms based on direct signals of a user's interest in a tag (Figure 4). Users may be more interested in tags they themselves apply. **Tag-applied** infers higher preference for those tags a user has applied. Users may also be interested in the tags for which they have searched. **Tag-searched** infers higher preference for tags for which a user has searched. Both tag-applied and tag-searched use a simple 0 or 1 numeric coding.

We also use a third implicit tag signal: a tag's quality (**tag-quality**). As we mentioned in the introduction, a user's preference towards a tag may be correlated with the tag's quality. In order to examine this relationship, we include the best performing tag quality prediction algorithm from our previous research [30].[8] The algorithm draws on many signals of tag quality including the number of users who apply a tag, and the number of users who search for a tag. In order to make our results more generalizable to other sites, we do not draw on the tag quality thumb ratings unique to MovieLens.

All tag preference algorithms translate between a score (i.e. 0 or 1 for tag-applied) and a one-to-five star inferred tag preference according to a simple linear relationship. This relationship is estimated by performing a least-squares regression between the algorithm scores and users' actual preference for tags as reported in the survey.

4.2 Inferring Preference using Item Signals

In this section, we explore algorithms that calculate a user's preference for a tag based on her interactions with movies related to the tag (Figure 5). We found that our inference algorithms performed better when they took into account the relevance of a tag to a movie. For example, if we wish to infer a user's preference for the tag *cars* we might treat her interactions with each of 46 movies tagged with cars as equally important. However, *cars* may be more relevant for certain movies than others. For instance, *cars* accounts for 9 of the 38 tag applications for "Gone in 60 Seconds," but only 1 of the 36 applications for "Cast Away."

To account for differences in a tag's relevance, each inference algorithm in this section includes a weighting quantifying the relevance of a tag to a movie similarly to Vig et al. [32]. As a measure of a tag's relevance to a movie, we use the tag quality measure we discussed in the previous section [30]. We found that applying a sigmoid transformation improved the performance of weighting by tag quality. If $w(m,t)$ represents the relevance weighting between a movie and tag:[9]

$$w(m,t) = \frac{1}{e^{-\text{tag-quality}(m,t)}}.$$

[8] We use the all-implicit tag quality inference algorithm from [30].

[9] We explored five other weightings (such as TF-IDF), and found that the tag quality-based weighting performed within 3% of the best novel weighting algorithm (a graphical bayesian network). The tag quality-based weighting also outperformed TF-IDF. For simplicity, we choose to build on our existing research instead of introducing a new weighting scheme.

We normalize $w(m,t)$ so that the weights for each movie's tags sum to 1.0.

We explore six different algorithms for calculating a user's preference for a tag based on her interactions with items having the tag. The six algorithms can be grouped according to the type of signal of movie interest that they use. The first two algorithms (movie-clicks, movie-log-odds-clicks) use clicks on movie hyperlinks as a signal of a user's interest in a movie. The third and fourth algorithms (movie-r-clicks, movie-r-log-odds-clicks), analyze the specific movies a user chooses to rate. The last two algorithms (movie-ratings, movie-bayes) draw on a user's numeric ratings for movies.

Movie-clicks: The movie clicks algorithm is based on the hypothesis that users click on movies with tags they like more often. This algorithm estimates a user's preference for a tag based on the fraction of clicked movies that have the tag. Instead of weighting each movie equally, we weight movies according to the relevance weighting based on quality we described above: If clicked(u) is the set of movies clicked by user u, then:

$$\text{movie-clicks}(u,t) = \frac{\sum_{m \in \text{clicked}(u)} w(m,t)}{|\text{clicked}(u)|}.$$

Movie-log-odds-clicks: Similar to movie-clicks, movie-log-odds-clicks assumes that users click movies with tags they like more often, but it adjusts for overall tag popularity. Movie-log-odds-click uses the log odds metric to compare the movie-specific tag frequency to the overall tag frequency. If M is the set of all movies, and M_t is the set of all movies with tag t,

$$\text{logit}(p) = log\left(\frac{p}{1-p}\right).$$

$$\text{movie-log-odds-clicks}(u,t) =$$

$$\text{logit}(\text{movie-clicks}(u,t)) \; - \; \text{logit}(\frac{\sum_{m \in M_t} w(m,t)}{|M|}).$$

Movie-r-clicks: Users' movie viewing decisions may correlate with their tag preferences. For instance, a user may choose to watch a movie because it contains "violence." We assume that users have watched the movies they have rated. Based on this, we consider a version of the movie-clicks algorithm that substitutes the movies a user has rated for the movies they have clicked.

Movie-r-log-odds-clicks: Similar to movie-r-clicks, movie-r-log-odds-clicks uses the movie-log-odds-click algorithm, but substitutes the movies a user has rated for the movies they have clicked.

Movie-ratings: Perhaps users rate movies with a particular tag consistently. For example, Alice consistently rated three *animated* movies five stars. Movie-ratings draws on this signal by predicting that a user's preference for a tag is the user's average rating for movies with the tag. As with the previous inference algorithms, we draw on tag quality for the tag relevance weighting w. If $r_{u,m}$ is user u's rating for movie m:

$$\text{movie-ratings}(u,t) = \frac{\sum_{m \in M_t} w(m,t) \cdot r_{u,m}}{\sum_{m \in M_t} w(m,t)}.$$

The sums in both the numerator and denominator ignore movies the user has not rated.

Movie-bayes: Movie-bayes is a bayesian generative model for how users rate movies with a particular tag [12]. Figure 6 describes the model. For every user u and tag t we select a user-tag-specific distribution $N(\mu_{\text{t,u}}, \sigma_{\text{t,u}})$ from hyper-distributions. For each rating $r_{u,m}$ by user u for movie m with tag t, the tag may be a relevant tag for the movie, or an irrelevant for the movie. If t is a relevant tag, the rating is chosen from the user-tag-specific distribution. If t is not relevant, the rating is chosen from the user's background ratings distribution $N(\mu_u, \sigma_u)$.

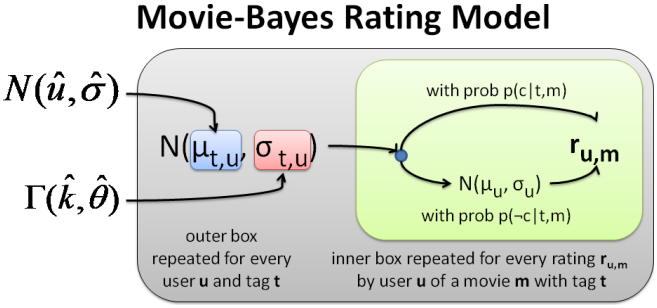

Fig. 6: *Movie-bayes is a generative model for how users rate movies with a particular tag. For every user u and tag t we select a user-tag-specific distribution $N(\mu_{t,u}, \sigma_{t,u})$ from hyper-distributions. For each rating $r_{u,m}$ by u for movie m with tag t, the tag may be relevant, or irrelevant for the movie. If t is relevant, the rating is chosen from the user-tag-specific distribution. If t was not relevant, the rating is chosen from the user's background ratings distribution $N(\mu_u, \sigma_u)$. We estimate the hyperparameters for hyperdistributions $N(\hat{\mu}, \hat{\sigma})$ and $\Gamma(\hat{k}, \hat{\theta})$ using the empirical bayes methodology. We calculate the expected parameters for a particular user and tag, $\mu_{t,u}$ and $\sigma_{t,u}$, using MCMC.*

We adopt the bayesian paradigm of considering all possible user-tag-specific normal distributions[9]. For each distribution, we calculate the probability that that distribution generated the user's ratings. We then take the expectation of the mean for a particular tag and user ($\mu_{t,u}$) over all possible distributions by applying bayes rule based on the user's ratings. In the following formulas $R_{u,t}$ is the set of ratings by user u for movies tagged with t, and $E(X)$ denotes the expectation of random variable X. $\Gamma(\hat{k}, \hat{\theta})$ and $N(\hat{\mu}, \hat{\sigma})$ specify the gamma and normal hyperdistributions for the standard deviation and mean respectively.

$$\begin{aligned}
&\text{movie-bayes}(u,t) = E(\mu_{t,u}|R_{u,t}).\\
&= \int_\mu \int_\sigma \mu \cdot p\Big(\mu, \sigma | R_{u,t}, N(\hat{\mu},\hat{\sigma}), \Gamma(\hat{k},\hat{\theta})\Big).\\
&= \int_\mu \int_\sigma \mu \cdot p(R_{u,t}|\mu,\sigma) \cdot p\Big(\mu, \sigma | N(\hat{\mu},\hat{\sigma}), \Gamma(\hat{k},\hat{\theta})\Big).\\
&= \int_\mu \int_\sigma \mu \cdot p(R_{u,t}|\mu,\sigma) \cdot p\Big(\mu | N(\hat{\mu},\hat{\sigma})\Big) \cdot p\Big(\sigma | \Gamma(\hat{k},\hat{\theta})\Big). \quad (1)
\end{aligned}$$

In equation 1 the second term, $p(R_{u,t}|\mu,\sigma)$, is the probability of the user's ratings for movies with a tag based on a user-tag-specific ratings distribution. To calculate this probability, we treat the ratings as independent events. As described earlier, each rating may be the result of the user's background ratings distribution, or a rating may be the result of the user's tag specific distribution. The user's background distribution, $N(\mu_u, \sigma_u)$, is fit to all of the user's ratings. The user-tag-specific distribution is chosen with probability equal to the relevance weighting $w(m,t)$:

$$p(R_{u,t,}|\mu,\sigma) = \prod_{r \in R_{u,t}} p(r|\mu,\sigma) =$$

$$\prod_{r \in R_{u,t}} \Big[p(r|N(\mu,\sigma))w(m,t) + p(r|N(\mu_u,\sigma_u))(1-w(m,t))\Big]$$

The third term in Equation 1, $p(\mu|N(\hat{\mu},\hat{\sigma}))$, is the prior probability of a mean for a particular user-tag-specific normal distribution.

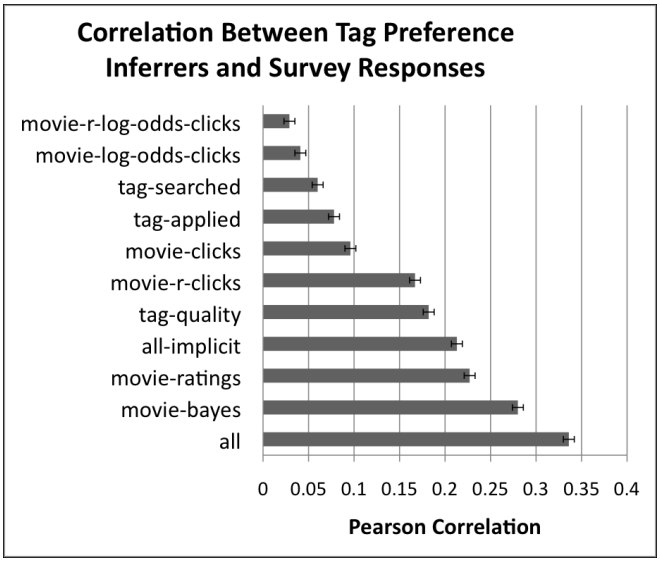

Fig. 7: *Pearson correlation between inferred tag preference, and actual tag preference. All pairwise differences are significant at the 0.05 level according to a two-tailed t-test. Algorithms based on explicit item signals performed best, followed by those based on implicit item signals, followed by those based on tag signals. All, a linear combination of all algorithms, performed best.*

We assume the user-tag-specific mean is drawn from a normal distribution with hyper-parameters chosen using the empirical bayes methodology [9]:

$$p(\mu_| N(\hat{\mu}, \hat{\sigma})) = p(\mu | N(2.885, 1.0)).$$

The fourth term in equation 1, $p\Big(\sigma|\Gamma(\hat{k}, \hat{\theta})\Big)$, is the prior probability a of a standard deviation for the user-tag-specific normal distribution. We assume the deviation is drawn from a gamma distribution with hyper-parameters chosen using the empirical bayes methodology:

$$p(\sigma|\Gamma(\hat{k}, \hat{\theta})) = p(\sigma|\Gamma(2.0, 1.0)).$$

The choice of a gamma and normal distributions as hyper-distributions for a normal distribution is common in the Bayesian literature [9]. We evaluate the complete integral using a Markov Chain Monte Carlo estimate.

4.3 Tag Preference Inference Results

As an evaluation metric, we calculate the Pearson correlation between each one to five star survey preference response and the inferred value from a particular algorithm (e.g. log-odds-clicks).

Figure 7 shows pearson correlations for all tag preference inference algorithms. 95% confidence intervals are displayed on the graph. All pairwise differences are significant. The two algorithms based on explicit item rating signals (movie-ratings, movie-bayes) outperformed all implicit measures. Both the click-based and ratings-based movie-log-odds-click algorithms performed poorly. The two tag-based algorithms (tag-searched and tag-applied) did not perform as well as the movie-clicks algorithm. We suspect that this is due to the relatively small amount of tagging activity on MovieLens. For example, only 2.8% of the survey rating responses were associated with a tag the user had applied. Similarly, only 0.6% of the survey responses were associated with a tag the user had searched for. Tag quality performed best among the algorithms using tag signals. Based on its relatively strong correlation (0.17), we conclude that the quality of a tag affects a user's preference towards the tag.

We also evaluated linear combinations of tag preference algorithms. We combined algorithms using a least square regression between tag preference responses and algorithm outputs.[10] **All-implicit** is the best combination of all tag preference inference algorithms that do not use movie ratings. **All** is the best combination of all tag preference algorithms. "All-implicit" outperformed each individual implicit feature. "All" outperformed all other algorithms.

In the following section we use the tag preference algorithms in tag-based recommendation algorithms. We either use "all", or "all-implicit" depending on whether the tag-based algorithm is designed for systems with or without ratings.

5. TAG-BASED RECOMMENDERS

In the previous sections we evaluated methods for inferring users' preferences for tags based on signals of interest in tags and items (Figure 1, upper left). We now shift our focus to using those inferred tag preferences to predict ratings for movies (Figure 1, upper right). We present five tag-based recommendation algorithms — two based on implicit data, and three based on explicit data. We then describe our methodology including our evaluation metrics and baseline recommender algorithms. Finally, we present results for all algorithms as they relate to our research questions.

5.1 Implicit Tag-Based Algorithms

We first consider two tag-based recommendation algorithms that use implicit data in order to support sites without item ratings. As input, these algorithms use tag preferences inferred by all-implicit, the top performing implicit tag preference algorithm. As output, these algorithms produce a score suitable for ranking items in a recommendation list. Recommender systems that do not collect ratings generally do not predict ratings; therefore, the values output by the implicit tag recommendation algorithms are suitable for the recommend task but not the predict task.

In the previous section we saw that the average tag preference differed by tag. For example, users generally preferred a high-quality tag to a low-quality one. During our analyses of tag-based recommenders, we found that these per-tag differences skewed results. We accounted for these per-tag differences by normalizing each tag's inferred preference to have mean 0 and standard deviation 1. In addition, we found that more active users generally had higher tag preferences than less active ones. To neutralize this effect, we subtracted the average tag preference for each user. We use these normalized tag preference values throughout this section.

We now describe the two implicit tag-based recommender algorithms:

Implicit-tag: The implicit-tag algorithm is inspired by algorithms from information retrieval that calculate the similarity between a user's profile vector and a document's term vector [23]. In information retrieval, the columns in each vector correspond to words. In the implicit-tag algorithm, the columns correspond to tags. To generate a prediction for a movie m, implicit-tag calculates the dot product between users' preferences for movie m's tags and the weighting $w(t, m)$ between tag t and movie m. We use prob-informed as a weighting based on its strong performance in section 4.4. If $ntp(u, t)$ is user u's normalized inferred tag preference for

[10]We tested support vector machines as well, but they yielded no significant improvement.

tag t, then user u's predicted score for movie m is:

$$\text{implicit-tag}(u,m) = \sum_{t \in T_m} ntp(u,t) \cdot w(m,t).$$

Implicit-tag-pop: Implicit-tag ignores the overall popularity of a particular movie, an important signal of a users's liking for a movie. We next consider implicit-tag-pop, a version of the algorithm that adds pop(m), a term estimating a movie's popularity:

$$\text{implicit-tag-pop}(u,m) = \text{implicit-tag}(u,m) + \text{pop}(m).^{11}$$

We experimented with a variety of signals of popularity for a movie based on the number of clicks, tags, clickers, and taggers for a movie. For each possible signal, we fit a function between the signal value for each movie (e.g. num clicks) and the average rating for the movie. We evaluated each signal of popularity based on how well implicit-tag-pop performed using the signal. Although we omit the detailed results due to space, we found that tags outperformed clicks, counting users (clickers) outperformed counting events (clicks), and log transforming signals improved results. The best overall estimate was a linear estimate based on the log of the number of users who tagged a movie. If $users(A_m)$ is the set of users who applied a tag to movie m, then:

$$\text{pop}(m) = 0.31 \cdot \log(|\text{users}(A_m)|) + 3.16.^{12}$$

5.2 Explicit Tag-Based Algorithms

The final three tag-based algorithms are intended for sites with item ratings; as a result, they rely on both implicit and explicit data. As input these algorithms use tag preferences inferred by *all*, the top performing tag preference algorithm using both implicit and explicit signals. Since these algorithms output a value between 1.0 and 5.0 corresponding to a star rating for a movie, they support both the predict and recommend tasks. We choose three algorithms that model increasingly complex relationships between tag preferences and movie ratings.

Cosine-tag: The success of the traditional item-based rating models that use cosine similarities inspired us to create a similar model based on tags. Cosine-tag predicts that a user's rating for a movie is a weighted average of the user's preferences for the movie's tags. Cosine-tag weights a particular tag according to the adjusted cosine similarity between ratings for a movie and inferred preferences for a tag. We refer to user u's mean movie rating as $\bar{r_u}$, and U_m is the collection of users who rated movie m. The adjusted cosine similarity between movie m and tag t is:

$$\text{sim}(m,t) = \frac{\sum_{u \in U_m} (r_{m,u} - \bar{r_u}) \cdot \text{ntp}(t,u)}{\sqrt{\sum_{u \in U_m} (r_{m,u} - \bar{r_u})^2} \sqrt{\sum_{u \in U_m} \text{ntp}(u)^2}}.$$

Note that $ntp(t,u)$ is already average adjusted, so there is no additional adjusting performed. Given this definition of similarity, cosine-tag constructs a prediction for a movie as the average of the user's preferences for its tags, weighted by the tags similarities to the movie. If T_m is the collection of all tags applied to movie m:

$$\text{cosine-tag}(u,m) = \frac{\sum_{t \in T_m} \text{sim}(m,t) \cdot \text{ntp}(t,u)}{\sum_{t \in T_m} \text{sim}(m,t)} + \bar{r_u}.$$

One choice in this algorithms is the tags over which the average should be calculated. We found that the algorithm performed best when averaging over the 10 most similar tags.

Linear-tag: Since cosine-tag predicts a weighted average of a user's inferred tag preferences, the variability of movie predictions it outputs depend on the variability of its inputs (inferred tag preferences). Linear-tag models a more complex relationship between an inferred tag preference and a predicted movie rating. For each tag t applied to movie m, linear tag estimates a least-squares fit $y_{t,m}(u)$ between users' inferred tag preferences for t and their ratings for m:

$$\text{y}_{t,m}(u) = \alpha_{t,m}\text{ntp}(t,u) + \beta_{t,m} + \epsilon_{t,m}.$$

In the linear equation above, $\alpha_{t,m}$ is the coefficient between tag t and movie m, $\beta_{t,m}$ is the intercept, and $\epsilon_{t,m}$ is the residual error term.

Linear-tag generates user u's prediction for movie m by averaging the values predicted by each of the linear fits $y_{t,m}(u)$. We found that weighting by inverse residual improved performance because it gave greater importance to more accurate fits. If A_m is the set of all tag applications for movie m, then:

$$\text{linear-tag}(u,m) = \frac{\sum_{t \in tags(A_m)} \left(y_{t,m}(u)/\epsilon_{t,m} \right)}{\sum_{t \in tags(A_m)} \left(1.0/\epsilon_{t,m} \right)} + \bar{r_u}.$$

As with cosine-tag, we experimented with averaging over different sets of tags. Averaging over the 5 tags with smallest residual performed best.

Regress-tag: The linear-tag model treats each linear fit between a tag and a movie as independent. This may not be optimal. For example, both *animated* and *animation* have been applied to "Toy Story." It seems plausible that users' inferred preferences for these two tags would correlate. Algorithms aware of relationships between tags may perform better than those that do not.

Regress-tag constructs a linear equation for each movie m. The input variables are all users' inferred tag preferences for tags applied to m. The output is each user's rating for m. If m has tags $t_1, \ldots, t_n$ then:

$$\text{regress-tag}(u,m) = h_0 + h_1\text{ntp}(u,t_1) + \ldots + h_n\text{ntp}(u,t_n).$$

We experimented with three methods for choosing the coefficients h_i: simple least-squares multiple regression, regularized multiple regression, and regression support vector machines. We found that the least-squares and regularized multiple regressions overfit movies with few ratings. For example, several movies had the same number of tags and ratings applied to them. In this case, multiple regression can build an equation that perfectly fits the input data. These fits often lead to large values h_i that seemed intuitively incorrect and performed poorly. SVMs performed best due to their robustness to overfitting. We used the libsvm library based on its java implementation and efficient performance for linear kernels [6]. We found that libsvm performed best when c, the tradeoff between the margin and error penalty, was set to 0.005.

[11] We experimented with different weightings of the pop term, but found 1.0 to perform optimally.

[12] Recall that we use the implicit tag-based algorithms for recommendation but not for prediction. Although we report the the intercept value (3.16), the choice of intercept does not affect the relative ordering. Therefore, the intercept is unnecessary - we could simply use 0.0.

5.3 Tagommenders Methodology

We compare the tag-based recommendation algorithms to three naive baselines:

Overall-avg: Overall-avg generates a prediction equal to the overall average rating (3.55):

$$\text{overall-avg}(m, u) = 3.55.$$

In the recommend task, movies are ordered by predicted rating. Since overall-avg returns the same value for every movie, we randomly order recommendation lists.

User-avg: User-avg predicts a user's average for all of his or her movies. If R_u is the set of all movie ratings by user u:

$$\text{user-avg}(m, u) = \frac{\sum_{r \in R_u} r_{m,u}}{|R_u|}.$$

As with overall-avg, given a particular user, user-avg returns the same value for every movie. Thus, we randomly order recommendation lists.

User-movie-avg: User-movie-avg begins by average adjusting all of a user's ratings. The prediction for a movie is the average of all users' adjusted ratings for the movie. If U_m is the collection of users who rated movie m, then:

$$\text{user-movie-avg}(m, u) = \frac{\sum_{u' \in U_m} \left(r_{m,u'} - \text{user-avg}(u') \right)}{|U_m|} + \text{user-avg}(u).$$

While these three naive baselines provide insight into algorithm performance, we ultimately compare our tag-based algorithms to top-performing traditional CF algorithms. We consider three traditional algorithms:

Explicit-item: We include the item-based algorithm introduced by Sarwar et al. based on its accuracy and popularity in real-world systems such as Amazon [17]. The explicit item-based model calculates similarities between the ratings for each pair of movies. In order to predict for a particular movie m, the item model constructs a weighted average of the user's ratings for the movies most similar to m. The rating weights used for the weighted average are based on the similarities to m.

Implicit-item: We compare our implicit tag-based algorithms to Karypis et al.'s item-based algorithm for unary data (such as click and transaction data) [14]. We selected this algorithm based on its accuracy and popularity. The item-based model calculates similarities between each pair of movies based on the number of times movies co-occur in user baskets. In order to predict for a particular movie m, the movie model sums the similarities between m and the movies in the user's basket that are most similar to m.

Funk-svd: We include Simon Funk's Singular Value Decomposition algorithm due to its strong performance in the Netflix competition [8]. The Funk SVD approximates the full users $\times$ movies rating matrix using a matrix of lower dimension, and uses regularization to manage the sparsity of the ratings matrix.

We used five-fold cross validation in our analyses. For each each of the five test / train splits, we hide 30% of user ratings in the test set, and evaluate the performance of an algorithm by comparing the ordering of a recommendation list to the hidden ratings. Herlocker et al. find two important classes of evaluation metrics: those that evaluate an algorithm's performance on the predict task, and those that focus on the recommend task [11]. We choose one evaluation metric from each class:

Top-5: As an evaluation metric for the recommendation task, we use top-5, the fraction of the top five recommended movies for a user that are rated four stars or higher by the user.[13] We only consider elements the user has rated when selecting the top 5 movies. The 95% confidence intervals for top-5 was $\pm 0.57\%$ ($n = 28,185$).

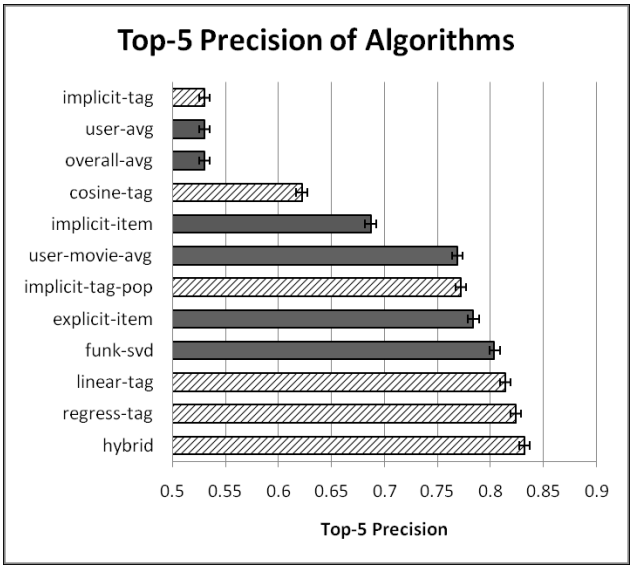

Fig. 8: *Top-5 precision for recommender algorithms. 95% confidence intervals are displayed for each algorithm. Higher top-5 values correspond to better performance. CF algorithms are displayed in solid bars and tag-based algorithms are displayed in striped bars. The best of the tag-based algorithms perform better than the best CF algorithms.*

MAE: In addition to top-5 we report mean absolute error (MAE), the average absolute difference between the value predicted by a recommender system and the user's actual rating value. MAE reflects an algorithm's performance on the predict task. We examined the distribution of MAE values produced in our analyses and found the 95% confidence intervals for MAE was ± 0.001 ($\mu = 0.577, \sigma = 0.491, n = 516,441$). As we discussed in Section 5.1, the implicit algorithms support only the recommend task. Therefore, we only report MAE for explicit algorithms.

5.4 Tagommenders Results and Discussion

Figure 8 shows the top-5 precision for the five tag-based algorithms, the three naive baselines, and the three collaborative filtering (CF) baselines. Higher top-5 values correspond to better performance. The traditional CF algorithms are displayed in solid bars and the tag-based algorithms are displayed in striped bars. We also include **hybrid**, a simple linear combination of the best performing tag-based algorithm (regress-tag) and traditional algorithm (funk-svd).[14] 95% confidence intervals are displayed for each algorithm.

Implicit-tag, user-tag, and overall-tag all achieve a top-5 of 53%, the same as randomly ordering a recommendation list. Differences between other pairs are significant ($p \leq 0.05$) except for those between user-movie-avg and implicit-tag-pop, and between regress-tag and hybrid. The tag-based algorithms generally perform well. Implicit-tag-pop, the best implicit algorithm, achieves a top-5 of

[13] We choose 4 stars as the cutoff for a "good" rating since it is the first star rating higher than the overall average rating of 3.55 stars. We experimented with other values for n in top-n: 1, 3, 5, 10, 20. We found the results to generally be consistent regardless of choice of n.

[14] We experimented with all mixtures between 0 and 100 in increments of ten, and found that a 50-50 average performed best

77%. Regress-tag, the best performing explicit algorithm achieves a top-5 of 83%.

Figure 9 shows the MAE for all explicit algorithms. The traditional CF algorithms are displayed in solid bars. Lower MAE values correspond to better performance. The tag-based algorithms are displayed in striped bars. All pairwise differences are significant ($p \leq 0.05$) except hybrid and funk-svd. As discussed in Section 5, the implicit algorithms only support the recommend task. Therefore, we only report MAE for explicit algorithms. In general, the tag-based algorithms outperformed the naive baselines, and the traditional CF algorithms outperformed the tag-based algorithms. Regress-tag performed best among the tag algorithms, achieving an mae of 0.584. As with top-5, cosine-tag performed poorly, achieving an mae of 0.639. Among the CF algorithms, funk-svd performs best, achieving an mae of 0.555.

Given these results, we return to our second research question:

RQ2: How well do tagommenders perform in systems without ratings?

As shown in Figure 8, implicit-tag-pop (77%) performs significantly better than the popular implicit-item algorithm (69%) according to top-5. We wondered if the strong performance of implicit-tag-pop compared to implicit-item was due to its inclusion of popularity. To test this possibility, we experimented with different methods for building popularity into the implicit-item algorithm. None of them significantly improved its performance. We conclude that the tagommender algorithm performs better than traditional CF algorithms in our evaluation without ratings.

Finally, we address our last research question:

RQ3: How well do tagommenders perform in systems with ratings?

Among the explicit tag algorithms, regress-tag performs best in both top-5 (83%) and MAE (0.584). Among the traditional CF algorithms, funk-svd performs best in both top-5 (80%) and MAE (0.555). Both these differences are significant ($p < 0.05$). We conclude that tagommenders appear to perform better than traditional CF algorithms for the recommend task, but worse for the predict task. However, the recommend task seems to be more prevalent in real world systems. Among all popular recommender systems we investigated, only three (Netflix, MovieLens, Rate Your Music) offer predicted ratings. Most recommender systems now follow Amazon's model. Amazon does support the recommend task. However, instead of supporting the predict task through CF, they offer users rich product data, user reviews, and average user ratings. Thus, tagommenders perform better than traditional CF algorithms in the task most important to real world recommender systems.

We conclude by noting that hybrid, the linear combination of funk-svd and regress-tag, offers the best of both tag-based and CF algorithms. Hybrid achieves a top-5 of 83%, equal to that of the top performing tag-based algorithm, and significantly better than traditional CF algorithms. In addition, hybrid's MAE equals that of funk-svd, the best performing traditional CF algorithm. Thus, a simple hybrid algorithm performs better than any CF algorithm on the recommend task and it matches the best CF algorithm on the predict task.

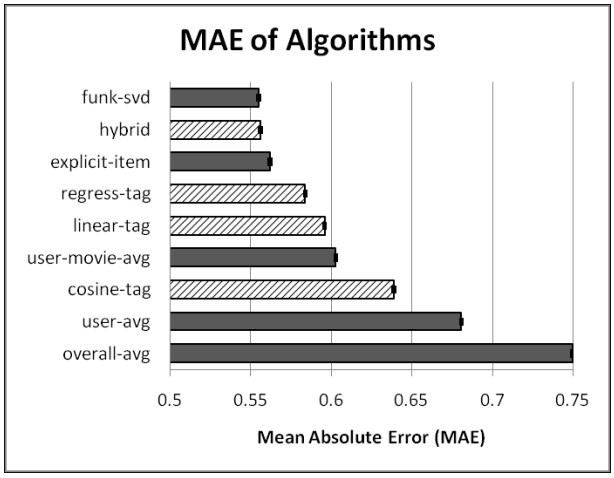

Fig. 9: *MAE for explicit algorithms. We do not include implicit algorithms since they do not support the predict task. 95% confidence intervals are displayed for each algorithm. Lower MAE values correspond to better performance. CF algorithms are displayed in solid bars and tag-based algorithms are displayed in striped bars. In general, tag-based algorithms perform better than naive baselines but worse than their CF counterparts.*

6. CONCLUSION

In this paper we introduced and evaluated tagommenders, recommender algorithms that make use of tags. We evaluated a wide variety of tag preference inference algorithms and found that algorithms combining a variety of signals performed best. We constructed implicit and explicit tag-based recommendation algorithms based on users' inferred tag preferences. These tagommenders outperformed existing CF algorithms in the recommend task most critical to real world recommender systems. Finally, we showed that a hybrid tag and CF algorithm combines the strong predict performance of CF algorithms with the strong recommend performance of tag based algorithms.

We believe that tagommenders may lead to novel interfaces for recommender systems. Since tagommenders use tags as an intermediary entity, their recommendations can be explained based on users' preferences for tags. MovieLens users often ask for the opportunity to rate movies on a more diverse set of dimensions. Tagommenders might prove a way to meet that desire.

Relationships between ratings and tags may also be used to infer the tags that should be applied to a movie. Although previous researchers have investigated the tag inference problem [13], they have not made use of patterns in users' ratings of items. For each movie, the cosine tag recommender calculates the similarity between between users inferred preferences for each tag and their ratings for the movie. For example the most similar tags for the movie "Last of the Mohicans" starring Daniel Day-Lewis are the tags *tribal*, *sword fight*, *cavalry charge*, *historical*, and *stirring*. The only one of these tags that users actually applied to the movie is *tribal.* Figure 6 lists 10 <tag, movie> pairs with highest similarity scores.

One important question related to our findings is how tagommenders will perform in domains other than MovieLens. While we cannot be certain, the high tag density of a system such as Delicious might lead to more accurate recommendations.

Finally, we believe there are a number of fundamental issues surrounding the relationship between preference and quality. It may be challenging to design interfaces that collect ratings along both the quality and preference dimension without confusing users. Perhaps because of this, most systems such as YouTube and Netflix only collect ratings along the preference dimension. Future research might explore interfaces for differentiating between quality and preference and examine the role quality and preference play in different domains.

Table 2: *Top 10 inferred tags not applied to movies based on the similarity between tag preferences and movie ratings. We do not include movies in a series (e.g. trilogies), and only include the top entry for tags or movies that appear more than once.*

movie	tag	cosine sim
Pearl Harbor (2001)	*disaster*	0.47
Runaway Bride (1999)	*girlie movie*	0.45
Beauty and the Beast (1991)	*talking animals*	0.42
Armageddon (1998)	*will smith*	0.41
Cinderella (1950)	*cartoon*	0.40
Inconvenient Truth (2006)	*documentary*	0.40
The Little Mermaid (1989)	*musical*	0.40
Gone in 60 Seconds (2000)	*exciting*	0.39
My Best Friend's Wedding (1997)	*chick flick*	0.39
Billy Madison (1995)	*very funny*	0.39

7. ACKNOWLEDGMENTS

The authors thank Arindam Banerjee, F. Maxwell Harper, Andrew Sheppard, Melissa Skeans, Katy Sen, and the entirety of GroupLens for their feedback on the ideas presented in this paper. We also thank the MovieLens community for their ratings, feedback and suggestions. This paper is funded in part by National Science Foundation grants IS 03-24851 and IIS 05-34420.

8. REFERENCES

[1] M. Balabanovic and Y. Shoham. Content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.

[2] R. Bell and Y. Koren. Lessons from the Netflix prize challenge. 2007.

[3] J. Bennett and S. Lanning. The Netflix Prize. In *Proceedings of KDD Cup and Workshop*, 2007.

[4] V. Bouthors and O. Dedieu. Pharos, a collaborative infrastructure for web knowledge sharing. In *ECDL*, pages 215–233, London, UK, 1999. Springer-Verlag.

[5] C. Brooks and N. Montanez. Improved Annotation of the Blogosphere Via Autotagging and Hierarchical Clustering. In *Proceedings of the 15th International Conference on World Wide Web*, pages 625–632. ACM New York, NY, USA, 2006.

[6] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[7] J. Diederich and T. Iofciu. Finding Communities of Practice from User Profiles Based On Folksonomies. In *Proceedings of the 1st International Worskhop on Building Technology Learning Solutions for Communities of Practice*, 2006.

[8] S. Funk. Netflix Update: Try This At Home. *sifter. org/ simon/journal/20061211. html*, 2006.

[9] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian Data Analysis, Second Edition*. Chapman & Hall/CRC, July 2003.

[10] C. Hayes, P. Avesani, and S. Veeramachaneni. An analysis of the use of tags in a blog recommender system. *ITC-IRST Technical Report. http://sra. itc. it/people/hayes/pubs/06/hayes-ijcai07-tech-report. pdf, June*, 2006.

[11] J. Herlocker, J. Konstan, L. Terveen, and J. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, Jan. 2004.

[12] T. Heskes. Empirical Bayes for Learning to Learn. In *International Workshop on Machine Learning*, pages 367–374, 2000.

[13] R. Jaschke, L. Marinho, A. Hotho, L. Schmidt-Thieme, and G. Stumme. Tag Recommendations in Folksonomies. *Lecture Notes In Computer Science*, 4702:506, 2007.

[14] G. Karypis. Evaluation of Item-Based Top-N Recommendation Algorithms. In *Proceedings of the tenth international conference on Information and knowledge management*, pages 247–254. ACM New York, NY, USA, 2001.

[15] G. Koutrika, F. Effendi, Z. Gyöngyi, P. Heymann, and H. Garcia-Molina. Combating spam in tagging systems. In *Proceedings of the 3rd international workshop on Adversarial information retrieval on the web*, pages 57–64. ACM Press New York, NY, USA, 2007.

[16] P. Lamere. Tagomendations - making recommendations transparent. http://blogs.sun.com/plamere/entry/tagomendations_making_ recommedations_transparent, 2007. Retrieved on October 30, 2008.

[17] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.

[18] G. MacGregor and E. McCulloch. Collaborative tagging as a knowledge organisation and resource discovery tool. *Library View*, 55(5), 2006.

[19] D. Millen, J. Feinberg, and B. Kerr. Social bookmarking in the enterprise. *ACM Queue*, 3(9):28–35, 2005.

[20] S. Niwa, T. Doi, and S. Hon'Iden. Web Page Recommender System Based on Folksonomy Mining. *Transactions of Information Processing Society of Japan*, 47(5):1382–1392, 2006.

[21] M. Pazzani and D. Billsus. Learning and Revising User Profiles: The Identification of Interesting Web Sites. *Machine Learning*, 27(3):313–331, 1997.

[22] M. J. Pazzani. A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, 13(5-6):393–408, 1999.

[23] V. Raghavan and S. Wong. A critical analysis of vector space model for information retrieval. *Journal of the American Society for Information Science*, 37(5):279–287, 1986.

[24] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. 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, Chapel Hill, North Carolina, United States, 1994. ACM Press.

[25] K. Rose. Digg: Recommendation engine rolling out this week. http://http://blog.digg.com/?p=127, 2008. Retrieved on October 30, 2008.

[26] J. Schafer, D. Frankowski, J. Herlocker, and S. Sen. Collaborative Filtering Recommender Systems. *Lecture Notes In Computer Science*, 4321:291, 2007.

[27] J. B. Schafer, J. Konstan, and J. Riedl. Meta-recommendation systems: User-controlled integration of diverse recommendations. In *Proceedings of the ACM Conference on Information and Knowledge Management*, pages 43–51, MacLean, VA, Nov. 2002.

[28] S. Sen, F. M. Harper, A. LaPitz, and J. Riedl. The quest for quality tags. In *GROUP '07: Proceedings of the 2007 international ACM conference on Supporting group work*, pages 361–370, New York, NY, USA, 2007. ACM.

[29] S. Sen, S. K. Lam, A. M. Rashid, D. Cosley, D. Frankowski, J. Osterhouse, F. M. Harper, and J. Riedl. tagging, communities, vocabulary, evolution. In *Proceedings of the ACM 2006 Conference on CSCW*, Banff, Alberta, Canada, 2006.

[30] S. Sen, J. Vig, and J. Riedl. Learning to Recognize Quality Tags. In *Proceedings of IUI*, 2009.

[31] C. Shirky. Ontology is overrated. http://www.shirky.com/writings/ontology_overrated.html, 2005. Retrieved on May 26, 2007.

[32] J. Vig, S. Sen, and J. Riedl. Tagsplanations: Explaining Recommendations Using Tags. In *Proceedings of the International Conference on Intelligent User Interfaces*, 2009.