Care to Comment? Recommendations for Commenting on News Stories

Erez Shmueli *
Dept. of Information Systems Engineering
Ben-Gurion University, Israel
erezshmu@bgu.ac.il

Amit Kagian
Yahoo! Labs
Haifa, Israel
akagian@yahoo-inc.com

Yehuda Koren
Yahoo! Labs
Haifa, Israel
yehuda@yahoo-inc.com

Ronny Lempel
Yahoo! Labs
Haifa, Israel
rlempel@yahoo-inc.com

ABSTRACT

Many websites provide commenting facilities for users to express their opinions or sentiments with regards to content items, such as, videos, news stories, blog posts, etc. Previous studies have shown that user comments contain valuable information that can provide insight on Web documents and may be utilized for various tasks. This work presents a model that predicts, for a given user, suitable news stories for commenting. The model achieves encouraging results regarding the ability to connect users with stories they are likely to comment on. This provides grounds for personalized recommendations of stories to users who may want to take part in their discussion. We combine a content-based approach with a collaborative-filtering approach (utilizing users' co-commenting patterns) in a latent factor modeling framework. We experiment with several variations of the model's loss function in order to adjust it to the problem domain. We evaluate the results on two datasets and show that employing co-commenting patterns improves upon using content features alone, even with as few as two available comments per story. Finally, we try to incorporate available social network data into the model. Interestingly, the social data does not lead to substantial performance gains, suggesting that the value of social data for this task is quite negligible.

Categories and Subject Descriptors

J.4 [**Computer Applications**]: Social and Behavioral Sciences

General Terms

Human Factors

*Work done while interning at Yahoo! Labs, Haifa.

WWW 2012, April 16–20, 2012, Lyon, France.
ACM 978-1-4503-1229-5/12/04.

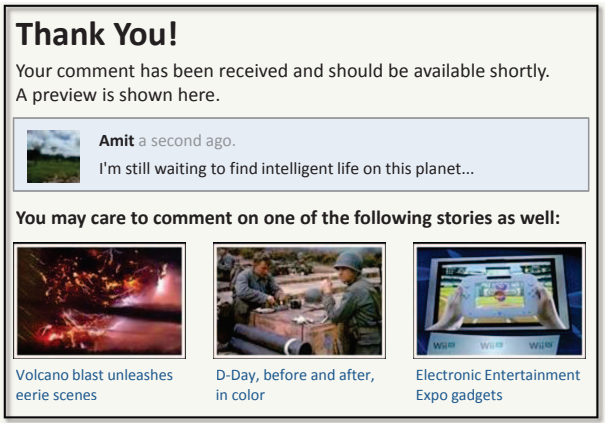

Figure 1: Demonstration of a potential application. After a user submits a comment to a story, the application recommends, along with the confirmation message, other stories to comment on.

Keywords

Personalization, recommendation system, user generated content, comment recommendation, latent factor models, collaborative filtering

1. INTRODUCTION

Much of today's media is consumed online, and media Web sites – whether online outlets of media companies with traditional forms of distribution (newspapers, TV stations, etc.) or portals of Web media companies – vie for the attention of consumers. New engaging user experiences enable sites to increase their user base as well as understand their users better, making them more attractive to advertisers.

A common feature of many present-day media sites is their attempt to enrich user engagement by soliciting and sharing user generated content (UGC). One popular form of UGC-centric engagement stems from comments on content that media sites solicit from users. Users are empowered to write comments and are able to rate others' comments, reply to comments (thereby generating discussion threads), etc.

This work focuses on news sites, and aims to present to each user a personalized ranked list of stories (that the user has yet to consume) that may engage the user to the point of commenting on them. An illustration of the planned experience is presented in Figure 1. This is an extreme form of content recommendation, as we go beyond engaging the user to click on or to consume information - we hope the user will be passionate enough about the presented stories to actually generate content. Furthermore, there is a time-critical dimension to our task – stories should be recommended to the users soon after they are published, as freshness is a major ingredient in a story's attractiveness.

Technically, we tap each user's propensity to comment on stories of certain topics, as well as collaborative filtering (CF) signals that rely on co-commenting patterns of users (tendency of different users to comment on the same stories). Note that when trying to recommend fresh stories, a trade-off emerges. Fresh stories have few comments, leading to lower contribution of the co-commenting signal. This is a form of the *item cold-start* problem in recommendation systems. On the other hand, waiting for more commenters to comment on the story helps in identifying additional users who would find the story engaging, but in the meantime the story will have "aged".

Another signal we tap in our experiments is social data. One of our datasets sports a social network between its users, prompting us to investigate whether and to what extent do stories, on which a user's friends have commented, engage the user to the point of commenting.

We apply a memory based recommendation algorithm to the signals above, as well as several flavors of a *latent factor model* (LFM). The LFM flavors deviate from each other by the loss function applied while learning them. Our experiments, on two different datasets with thousands of stories and users, achieve encouraging results.

The contributions of this paper are along two axes. First, with respect to the application domain, we measure the relative strengths of three signals (textual tags, co-commenters, and friends) as predictors of a person's tendency to comment on stories. Our findings show that:

- Textual tags provide a signal that is already surpassed by roughly 5 co-commenters in the tested datasets. Furthermore, combining tags and co-commenter information improves on what each signal can achieve alone.
- We study the tradeoff between recommendation accuracy and the number of comments available per story.
- The contribution of social signals to commenting activity prediction is lower than either of the other signal types.

Second, with respect to the art of recommendation systems, we find that:

- One can improve the accuracy of latent factor models (LFMs) by tailoring the loss function to the specific application at hand rather than using the standard squared error loss function.
- In cases where uniformly unattractive items can be easily identified, i.e. stories on which almost no users will comment, the AUC metric may fail to measure the personalization capabilities of competing recommendation schemes. We therefore introduce a novel variant metric, called *stratified AUC*, that measures the accuracy of ranking items in separate bins that correspond to the items' global popularity. This isolates the uniformly unattractive items and prevents them obscuring the quality of the evaluated recommendation scheme.

The rest of this paper is organized as follows. Section 2 surveys related work. Section 3 presents the LFM and memory based recommendation algorithms we evaluated. Section 4 describes the two datasets in our experiments, and Section 5 reports our results. We conclude in Section 6.

2. RELATED WORK

Since the appearance of Web 2.0 and, with it, user generated comments, various studies explored different aspects of user comments, showing that comments can provide many insights about Web documents. Mishne and Glance [16] presented a study of weblog comments and their relation to blog posts, for example, exploring the relation between the weblog popularity and users' commenting patterns. Siersdorfer at al. [26] studied user comments on YouTube videos and gave an analysis of the dependencies between comments, views, comment ratings and topic categories along with a predictor of comments' ratings. Since user comments often express apparent feelings of users, the study of user comments has been also associated with sentiment analysis (see survey by Pang and Lee [17]). For example, a study by Pavlou and Dimoka [19] that demonstrated the role of user comments in building sellers' benevolence and credibility in online marketplaces.

Other studies utilized user comments for the completion of various tasks: user comments on blog posts were utilized for automatic blog summarization [9] and for clustering of blog documents [11]. Yee et al. [31] demonstrated the potential of Youtube user comments to annotate videos by incorporating comments into the search index, which yielded up to a 15% increase in search accuracy. Studying the content sharing website Digg, Rangwala and Jamali [21] defined a co-participation network between users based on Digg's comment information and used this data to predict the popularity of online content linked at Digg. A recent study by Chen et al. [3] focused on user reputation in a comment rating environment and showed that the quality of a comment judged editorially is almost uncorrelated with the ratings that it receives, but can be predicted using standard text features.

Several past studies focused on user comments on news articles. Li et al. [12] employed user comments for dynamically suggesting related stories to a given news article, showing that suggestions improve when the content of user comment threads is considered. A recent study by Park et al. [18] utilized user comments in order to identify the political orientation of news articles by uncovering user sentiments expressed in their comments. Other studies [28, 29] explored the distributions of comments on news from various news agents and tried to predict the comments' volume for news stories. Schuth et al. [25] examined relations between news comments by extracting discussion structures ('replies-to' relations) from flat comment threads. Hsu et al. [8] presented a machine learning approach for ranking comments based on the expressed preferences of a social web community, which can be used to promote high-quality comments and filter out low-quality ones. A recent study by Agar-

wal et al. [2] added personalization to comment ranking and used a latent factor model to rank comments according to personalized preferences of specific users. The work presented in this paper adds to the line of user generated comments research by describing a model that predicts, for a given user, the most probable news stories for commenting and thus, provides personalized recommendations of stories to comment on.

Our work lies within the field of recommender systems [23]. Broadly speaking, recommender systems are based on two different strategies. The *content filtering* approach creates a profile for each user or product to characterize its nature. As an example, a news article profile may include its publisher, author, subject tokens, categories, tags, textual attributes, etc. Similarly systems may employ user profiles including demographic information or answers to a suitable questionnaire. The resulting profiles allow programs to associate users with matching products. Much more details can be found in Ricci et al. [23].

An alternative to content filtering relies only on past user behavior—e.g., previous transactions or product ratings—without requiring the creation of explicit profiles. This approach is known as *Collaborative Filtering* (CF), a term coined by the developers of the first recommender system - Tapestry [7]. CF analyzes relationships between users and interdependencies among products, in order to identify new user-item associations. A major appeal of CF is that it is domain free, yet it can address aspects of the data that are often elusive and difficult to profile using content filtering.

A common division of CF is between a class of "memory-based" methods, and a class of "model-based" methods [1]. Memory-based methods are essentially heuristics that generate recommendations by directly referring to the raw user feedback data, or to values immediately derived from it. A known example is the item-item recommendation method [14, 24]. Model-based methods initially construct a compact model for representing the inherent structure essential for relating user with items. Consequently, recommendations are made by referring to the created model. The models often follow a latent factor representation (or, LFM), such as the widely used matrix factorization method [10]. LFM tries to explain the user behavior by characterizing both items and users on factors automatically inferred from the patterns of user-item interactions.

While generally being more accurate than content based techniques, CF suffers from what is called the "cold start" problem, due to the inability to address products or users new to the system. This often leads to favoring *hybrid methods* [1] that combine content and collaborative filtering together. The methods proposed in this paper are indeed such a hybrid.

Most works in the recommenders field deal with entertainment and media product. More relevant to us are the works dealing with recommending textual items. Such textual items, like news articles, are characterized by high volatility and churn rate, while offering an opportunity of content analysis of their text. One such work [4] describes Google news recommendation engine, where the authors use a blend of three separate CF methods in a highly scalable fashion. Moving closer to the UGC domain, several recent works [5, 13] addressed question recommendation in community question answering sites.

Table 1: Summary of Notations

$T(s)$	The set of textual tags associated with story s
$C(s)$	The set of users that commented on story s
U	The set of all users in the data
S_u^+	The set of stories a user u commented on
S_u^-	The set of stories a user u did not commented on
S_u	A set of positive-negative story pairs for user u
s	Index letter for a story
u	Index letter for a user receiving recommendations
c	Index letter for a user as a commenter
t	Index letter for a textual tag
r_{us}	An affinity score for user u and story s
$\bar{x}$	Vector representation of x in the K-dimensional latent space (where x can be a user, story, tag)

3. RECOMMENDATION METHODS

We begin by presenting some notations used throughout this section. For a given story s, we denote the set of textual tags associated with it by $T(s)$. Similarly, we denote the set of users that commented on s by $C(s)$. Let U denote the set of all users in the data. For each user $u \in U$, let S_u^+ be the set of stories that u commented on, also referred to as the *positive examples* of u. Similarly, let S_u^- be the set of stories that u did not comment on, also called the *negative examples* of u. As part of our LFM learning procedure (section 3.2) we train our models on the same number of positive and negative examples for each user u. To this end, we pair each positive example $s^+ \in S_u^+$ with a sampled negative example $s^- \in S_u^-$. Generally, negative examples are drawn uniformly from S_u^-, unless otherwise stated. We denote by S_u the set of positive-negative story pairs (s^+, s^-) over which the LFMs are trained with respect to a user u. As for the LFM parameters, we use a bar notation to distinguish vectors in the latent space from the scalars they represent. For example, $\bar{u} \in \mathbb{R}^K$ is the vector representation of user u and $\bar{s} \in \mathbb{R}^K$ is the vector representation of story s. Table 1 summarizes these notations for easy referencing.

We treat the problem of commenting recommendation as a ranking problem. Given a user u, the goal is to rank the test stories so that if story s_1 is ranked higher than s_2 then u is more likely to comment on s_1 than on s_2. The ranking of test stories is accomplished by using a prediction model that given a user u and a story s, generates a score $r_{u,s}$ reflecting how likely is u to comment on s. We addressed this problem with two approaches, a memory based approach and a more complex latent factor model in several variations, some of which are novel with respect to recommendation system usage. The rest of this section describes the methods that were implemented in the two approaches.

3.1 Memory Based Approach (MB)

The memory based approach directly leverages the notion of co-occurrence. This is done by calculating a co-occurrence similarity measure for a given user and all story features (textual tags and given commenters) and averaging this measure over all features of a given story.

Thus, the memory based formula for $r_{u,s}$ is given by

$$r_{u,s} = \frac{1}{|\{t \in T(s)\}|} \sum_{t \in T(s)} \frac{n_{u,t}}{\sqrt{n_u n_t}} + \frac{1}{|\{c \in C(s), c \neq u\}|} \sum_{c \in C(s), c \neq u} \frac{n_{u,c}}{\sqrt{n_u n_c}} \quad (1)$$

The variable $n_{u,t}$ is the number of stories with tag t that user u had commented on, n_u is the total number of stories that u had commented on and n_t is the total number of stories with tag t. Similarly, $n_{u,c}$ is the number of stories which both users u and c have commented on. Implementation-wise, we build an inverted index [15] over training period stories, along with their tags and commenting users. This results in posting lists per tag and commenter, whose posting elements contain (and are sorted by) story ID. The index allows us to quickly compute co-occurrence (intersection) counts on the fly, by running two-term conjunctive queries against it.

3.2 Latent Factor Model

The basic idea behind latent factor models is to represent each user u and story s in a K-dimensional latent space, so that, if $\bar{u} \in \mathbb{R}^K$ and $\bar{s} \in \mathbb{R}^K$ are the representations of u and s in the latent space, then the score $r_{u,s}$ is calculated by the inner product of the representation vectors, plus an additive user bias

$$r_{u,s} = \langle \bar{u}, \bar{s} \rangle + b_u = \left(\sum_{k=1}^{K} \bar{u}_k \bar{s}_k \right) + b_u \quad (2)$$

where $b_u \in \mathbb{R}$ is a user bias parameter. Note that while the user biases have no effect on the ranking of stories for a fixed user, we still find them improving results as their inclusion leads to a more accurate training of the latent factors.

In many recommender system settings, items are directly modeled based on user interactions with them (during the training period). Our setting is different because of the ephemeral nature of news stories. Hence, we assume no user interactions are available for test stories (see also the related design of the train-test split in Section 4). Therefore latent space representation for stories ($\bar{s}$) would not be a distinct parameter of the model. Rather, we assign latent space parameters to the story features (textual tags and commenters) and construct the stories' representations through their features. This is feasible since the same features do appear with other stories during training. Formally, each textual tag t and commenting user c are represented by factor vectors, $\bar{t} \in \mathbb{R}^K$ and $\bar{c} \in \mathbb{R}^K$, respectively. The representation of a story s is given by averaging the representations of its features

$$\bar{s} = \frac{1}{|\{t \in T(s)\}|} \sum_{t \in T(s)} \bar{t} + \frac{1}{|\{c \in C(s)\}|} \sum_{c \in C(s)} \bar{c} \quad (3)$$

Note that we chose to use two different latent representations for each user: $\bar{u}$ as a user receiving a recommendation (see equation (2)) and $\bar{c}$ as a commenter (being a feature of a story). This is done in order to allow the model to address the different roles a user assumes. In our experiments, we used a value of $K = 10$ for the latent space dimensionality. Other values we experimented with (20,50), did not produce any observable gain in performance.

The parameters of the latent factor models were learned by employing a stochastic gradient descent algorithm with early stopping (see, e.g., [27]). The algorithm attempts to minimize the cost function

$$cost = \sum_{u \in U} \sum_{\{s^+, s^-\} \in S_u} loss(u, s^+, s^-) \quad (4)$$

where $loss(u, s^+, s^-)$ is one of the loss function described in the rest of this subsection.

The training procedure is an iterative process and a typical number of iterations is 20. At each iteration, we iterate over all users $u \in U$. For each user, we train the model over all positive-negative story pairs in S_u. Recall that the story pairs in S_u include all the positive examples in S_u^+ and for each $s^+ \in S_u^+$ a sampled negative example $s^- \in S_u^-$. Negative examples are sampled uniformly except for the case of the last loss function (see 'Rank Loss' below).

Following is a high level pseudocode description of the training procedure:

```
while validation performance is improving do
  for all u ∈ U do
    for all {s+, s−} ∈ S_u do
      calculate the gradient ∇(loss(u, s+, s−))
      change model parameters by −η·∇(loss(u, s+, s−))
    end for
  end for
end while
```

We used a learning rate of $\eta = 0.1$. The model's latent factor parameters are regularized by penalizing their squared magnitude with a regularization coefficient $\lambda = 0.1$ (also known as Tikhonov regularization). Following is a description of the various loss functions used in our experiments.

Squared Error (SE)

The most common loss function in recommendation systems latent factor models is the squared error (e.g. [10]). Since recommendation systems commonly try to predict a user-item rating (e.g., movie ratings), the SE loss function aims at minimizing the distance of predictions from target ratings; in our case it aims at getting the score of positive examples close to 1 and of negative examples close to 0

$$se\_loss(u, s^+, s^-) = (1 - r_{u,s^+})^2 + (0 - r_{u,s^-})^2 \quad (5)$$

Classification Error (CE)

While observing the SE loss function, one may argue that it is unjustified to penalize positive examples with scores higher than 1. The same holds for negative examples with scores lower than 0. The 'Classification Error' deals with that by using a hinge loss that better meshes with misclassification. Thus, the loss term is

$$ce\_loss(u, s^+, s^-) = |1 - r_{u,s^+}|_+ + |r_{u,s^-}|_+ \quad (6)$$

where $|x|_+ = x$ if $x \geq 0$ and 0 otherwise.

Bayesian Personalized Ranking (BPR)

Rendle et al. [22] suggested a loss function that is designed for ranking problems. The method treats pairs consisting of a positive and a negative example. The goal is to score the

positive example higher than the negative example's score. The loss term is

$$bpr\_loss(u, s^+, s^-) = -log(\sigma(r_{u,s^+} - r_{u,s^-})) \quad (7)$$

where $\sigma(x)$ is the sigmoid function $\frac{1}{1+e^{-x}}$.

Rank Loss (RL)

Bengio et al. [30] suggested a loss function that utilizes a different sampling method for negative examples, which we adopt here for recommendation purposes. For each positive example $s^+ \in S_u^+$, negative examples are drawn with replacement until a negative example $s^- \in S_u^-$ that violates the desired ranking is encountered. A negative example $s^- \in S_u^-$ is considered violating, with respect to a positive example $s^+ \in S_u^+$, if its score is within a margin of 1 from the positive example's score or higher, that is, if it satisfies $r_{u,s^+} \leqslant 1 + r_{u,s^-}$. The model is then updated based on such violating pair of examples. No update is performed if no violating example could be found after $|S_u^-|$ draws. The number of non-violating negative examples drawn while sampling is used to estimate the total number of negative examples that violate the desired ranking for a given positive example. We denote this as *rank* in the following formulation

$$rank = \frac{total\ number\ of\ negative\ examples}{number\ of\ negative\ examples\ drawn}$$

We used a binary *rankloss* function after other suggested variations [30] we experimented with did not improve performance.

$$rankloss(rank) = 1 \text{ if } rank > 1,\ 0 \text{ otherwise}$$

Then the loss function for user u and positive-negative story pair $\{s^+, s^-\} \in S_u$ becomes

$$rl\_loss(u, s^+, s^-) = rankloss(rank) \cdot (1 - r_{u,s^+} + r_{u,s^-}) \quad (8)$$

4. DATASETS & EXPERIMENTAL SETUP

In order to evaluate our methods, we examined two different datasets of news articles, comments and users. We proceed to describe how each of the datasets was obtained and preprocessed.

The first dataset was crawled from the Newsvine news site[1]. Newsvine is a community-powered, collaborative journalism news website, owned by msnbc.com. Users can write articles, seed links to external content, and discuss news items submitted by both users and professional journalists. We have chosen to crawl the Newsvine site, among dozens of other available news sites, since: (1) Newsvine is relatively easy to crawl due to the static HTML nature of its content pages; and (2) its registered users constitute a social network that is publicly visible. Our crawler fetched all news articles published from May 18, 2011 till September 27, 2011. Each fetched article was parsed in order to extract its title, content, editorial tags, time of publication and set of associated comments. From each comment we extracted its commenter ID. Finally, the set of friend IDs of each commenter was obtained. As shown in Table 2, the Newsvine dataset contains 111,893 articles with 3,009,247 comments and 221,809 users. Furthermore, its social network contains 106,705 (symmetric) friendship relations, i.e. undirected friendship edges.

[1]www.newsvine.com

dataset	type	stories	users	comments
Newsvine	Original	111,893	221,809	3,009,247
	Train	44,573	6,173	485,247
	Test	6,232	6,173	177,304
Yahoo!	Original	25,527	320,425	1,664,917
	Train	15,236	8,015	355,762
	Test	2,815	8,015	96,947

Table 2: Data statistics for the Newsvine and Yahoo! datasets before and after splitting and preprocessing.

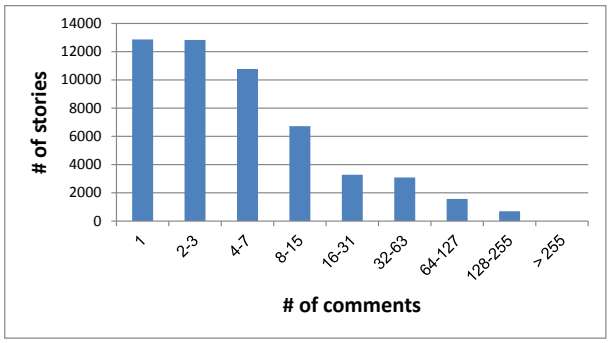

Figure 2: The distribution of stories by their number of comments in the Newsvine dataset (after preprocessing).

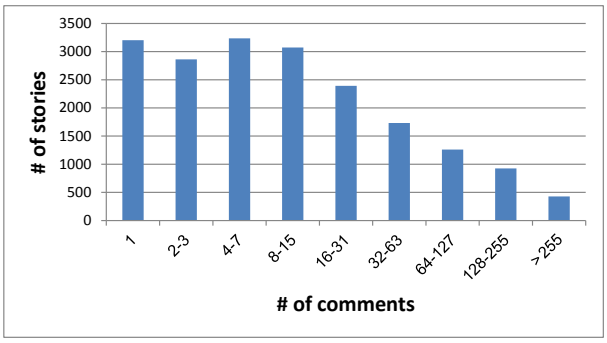

Figure 3: The distribution of stories by their number of comments in the Yahoo! dataset (after preprocessing).

The second dataset was a set of articles obtained from Yahoo! News[2]. Yahoo! News is an Internet-based news aggregator provided by Yahoo!. Articles in Yahoo! News come from news services, namely Associated Press (AP), Reuters, Fox News, CNN.com, BBC News, and others. Similarly to Newsvine, Yahoo! News also allows users to comment on news articles. Our set of articles was sampled so as to contain articles with varying number of comments. As shown in Table 2, this dataset contains 25,527 articles with 1,664,917 comments and 320,425 users.

Since the first dataset was crawled from the Newsvine website we could not obtain any click data that can validate

[2]news.yahoo.com

which uncommented stories were actually viewed by a user. In order to compare the two datasets properly we did not use any click data on the Yahoo! dataset as well.

Each article (in both datasets) was later represented as a set of tags and a list of commenters (ordered by their commenting time on the article). The set of tags contained both editorial tags (only available for the Newsvine dataset) and named entities which were extracted from the article's title and content. In order to keep the article's representation as compact as possible, we chose to only extract named entities, such as names of people and places, and not its entire set of terms. In order to identify named entities, we applied a very simple heuristic that looked for sequences of words beginning with a capital letter, except for sequences of length one appearing at the beginning of a sentence. The results of this simple yet fast heuristic sufficed for our needs.

Then, each dataset was split into a training set, containing 60% of the articles, and a test set, containing the remaining 40%. The articles were split according to the chronological order of their publish dates, where older articles constituted the training set and newer ones the test set.

Finally, users and stories that did not meet certain conditions were filtered out, as hereby described. Users with less than ten comments in the training set were removed, being an inappropriate target of recommendation. Users without comments in the test set were also removed, as they cannot be evaluated properly. In addition, we filtered out from the training set stories without comments, and from the test set stories with less than five comments. We required stories in the test set to have at least five comments due to the offline nature of our evaluation, as will become clearer in Section 5. Table 2 summarizes the number of stories, users and comments for the training and test sets of each dataset. Figures 2 and 3 illustrate the distribution of stories by their number of comments in the Newsvine and Yahoo! datasets, respectively, after the preprocessing phase.

5. RESULTS

5.1 Evaluation: Predicting Comments

We trained five different models over the training set data: the Memory Based model (MB) and Latent Factor Models with four error function variants (see Section 3). Once all models were trained, they were used to generate, for each user, a score for all stories in the test set.[3] Then, test stories were ranked for each user according to the assigned scores. We used the Area Under ROC Curve (AUC) measurement in order to evaluate the rankings. In general, the AUC value is equivalent to the probability that a randomly chosen positive example is ranked higher than a randomly chosen negative one [6]. Eventually, we averaged the achieved AUC values over all users. Table 3 shows the obtained mean AUC values for both datasets. Overall, all methods perform quite well on the Newsvine dataset, with a minor advantage to BPR with 87% AUC. On the Yahoo! dataset, performance differences are more significant; RL is the best performing method with 72.2% AUC while BPR is lagging behind significantly with AUC=64.3%.

[3]Stories in which the user was among the first five commenters were excluded from the test for that user, since the first 5 commenters per story are treated as known commenters and comprise the co-commenting features of a story.

Method	Newsvine AUC	Yahoo! AUC
MB	0.845	0.661
SE	0.867	0.699
CE	0.859	0.696
BPR	**0.870**	0.643
RL	0.863	**0.722**

Table 3: Obtained AUC values for the various methods over the Newsvine and Yahoo! datasets.

Stratified AUC

The AUC metric, while common in the literature, is susceptible to easily-forecasted elements. Whenever a considerable portion of the ranked elements are easily classified as positive or negative, the AUC value increases and its power of discrimination between better and worse methods deteriorates.

In our case, many stories are *predictably unpopular*, i.e. it is easy to identify many stories that are unlikely to be commented on by any user. We call these *easy negatives*. As our emphasis is on evaluating the methods by their ability to personalize recommendations, we would like to deemphasize the masking effect that easy negatives have on the AUC metric. In a broader sense, we would like to promote methods that are good at personalization, and demote methods that are merely able to predict the universal popularity of stories. To demonstrate that the AUC metric is biased toward popularity prediction, we measured the success of a *popularity oracle* that ranks (in hindsight) the test stories, *uniformly for all users*, by the number of comments they eventually received. Obviously this oracle does no personalization at all, as it recommends to all users the same order of stories[4]. We also reiterate that it is an offline algorithm that "learns" from the test data and thus naturally overfits. Indeed, the ranking produced by the "popularity oracle" achieved very high AUC values, with 89.7% AUC on Newsvine and 85.8% AUC on Yahoo!, handily beating all other methods (compare with the values in Table 3).

In order to mitigate the influence of the popularity bias in our evaluation and to refocus it on personalization, we propose a novel *stratified-AUC* performance measurement (sAUC). The sAUC metric calculates the probability of a random positive example to be ranked higher than a random negative example of (approximately) equal popularity. More specifically, sAUC partitions the test stories into popularity bins (see Figures 2 and 3), and computes AUC separately within each bin. These bin-specific AUC values are then averaged proportionally to the number of positive examples in each bin.

In order to suit the sAUC measure better, we retrained our latent factor models with an emphasis on discriminating between positive and negative stories within the same popularity bin. To this end, when training the model on a positive example and a negative example (see Section 3), we sampled the latter from the same popularity bin as the positive one. Table 4 displays sAUC values for both datasets. As evident, the "popularity oracle" loses its power when eval-

[4]The popularity oracle is the optimal user-independent ranking of test stories, i.e. no other such ranking of stories achieves a better AUC.

Method	Newsvine sAUC	Yahoo! sAUC
MB	0.682	0.639
SE	0.700	0.686
CE	0.702	**0.690**
BPR	**0.713**	0.660
RL	0.700	0.685
"popularity oracle"	0.582	0.565

Table 4: Obtained stratified-AUC values for the various methods over the Newsvine and Yahoo! datasets. In contrast to the classic AUC measurement, here the *"popularity oracle"* ranking achieves the poorest results.

uated with sAUC, while the other methods still achieve fair performance, with BPR attaining the best performance of 71.3% sAUC on Newsvine and CE reaching 69% sAUC on Yahoo!. It is interesting to note that while all the latent factor models outperform the memory based approach, there is still no clear winning method for both datasets. Namely, while BPR achieves the best AUC and sAUC on Newsvine, its performance on the Yahoo! dataset is inferior to the other LFMs. This promotes a possible invstigation of the differences between the two datasets and a research of better adjusted LFM loss functions in future research.

5.2 Content vs. Collaborative Filtering

As described in section 3, our methods utilize two types of data signals, textual tags and co-commenting patterns, as predictors of a person's tendency to comment on stories. In this section we attempt to quantify the impact of each signal alone, and of their combination on the recommendation accuracy. In order to achieve this goal, we created three variations for each one of our prediction models:

Tags. This variation takes into account the textual tags signal alone while completely ignoring the co-commenting signal.

Co-Commenting. This variation takes into account the co-commenting signal solely while completely ignoring the textual tags signal. Furthermore, we used this variation in order to investigate the tradeoff between recommendation accuracy and the number of comments available per story. In order to accomplish this task we tried different values (one to five) of given comments for test stories.

Both. Taking into account both of the co-commenting and textual tags signals. Similarly to the Co-Commenting variation, we tried different values (one to five) of given comments for test stories.

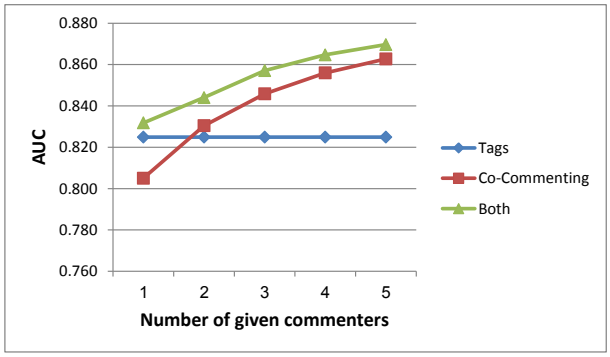

Figure 4: Evaluating the AUC metric on a varying number of given commenters. The LFM model is trained and tested using the BPR loss function over the Newsvine dataset.

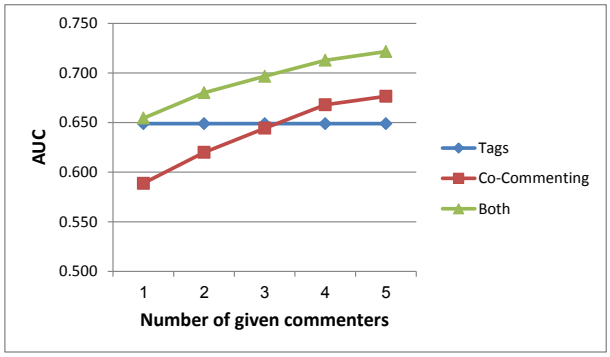

Figure 5: Evaluating the AUC metric on a varying number of given commenters. The LFM model is trained and tested using the RL loss function over the Yahoo! dataset.

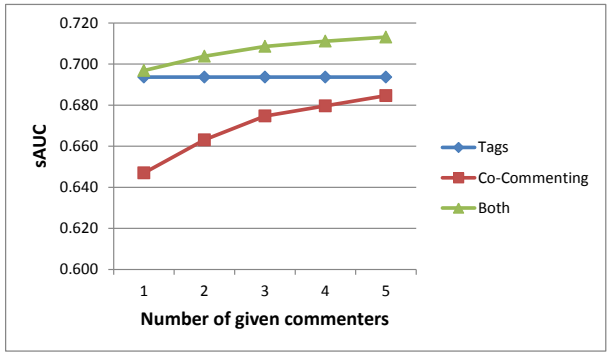

Figure 6: Evaluating the stratified-AUC metric on a varying number of given commenters. The LFM model is trained and tested using the BPR loss function over the Newsvine dataset.

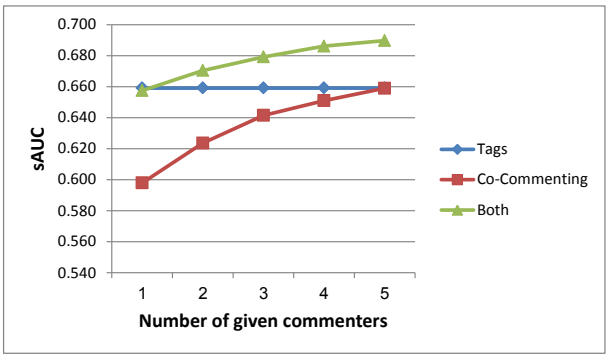

Figure 7: Evaluating the stratified-AUC metric on a varying number of given commenters. The LFM model is trained and tested using the CE loss function over the Yahoo! dataset.

Figures 4-7 show the results of applying the three different variations over the prediction models that achieved the best results in the previous subsection. More specifically, Figures 4 and 6 show results for BPR when evaluating the AUC and sAUC measures on the Newsvine dataset. Figure 5 shows results for RL when evaluating the AUC measure on the Yahoo! dataset. Finally, Figure 7 shows results for CE when evaluating the sAUC measure on the Yahoo! dataset. A careful examination of the four figures implies that when using 5 given commenters, the co-commenting signal provides a signal that is equivalent or higher than the textual tags signal. Interestingly, this reaffirms the findings by Pilászy and Tikk [20] in the very different domain of movies. Furthermore, combining tags and co-commenting information improves on what each signal achieves alone, even for as little as two given commenters.

5.3 Social Data

As mentioned in Section 4, the Newsvine site has a dedicated social network among its users. Furthermore, the Newsvine friendship relations are publicly crawlable. We thus examined whether tapping the co-commenting patterns of a user's friends can help improve our personalized recommendation for the user. We created a subset of the Newsvine dataset that includes only users with at least one friend (and stories commented by such users, etc.). Data statistics on the resulting dataset are shown in Table 5.

type	stories	users	comments	friendships
Train	40,823	2,747	272,357	36,284
Test	5,207	2,747	177,304	36,284

Table 5: Social data statistics for the Newsvine dataset.

We experimented with multiple approaches for representing the social data in the memory based and LFM models. In all tested approaches, adding social data had a negligible effect on our ability to predict users' comments. We therefore only report below the results of the best variation among those tested.

The approach that obtained the maximal recommendation accuracy was an LFM based on modifying each user representation to include information about his friends. We denote the friends of user u by $friends(u)$, and represent each friend f by latent factor vector $\bar{f} \in \mathbb{R}^K$. The vector representation of u, $\bar{u}$ (see Section 3.2) is now replaced by the vector $\bar{u_F}$

$$\bar{u_F} = \bar{u} + \frac{1}{|friends(u)|} \sum_{f \in friends(u)} \bar{f} \qquad (9)$$

We also tried an alternative approach of enhancing the story representation based on friendship relationships, which resulted in inferior performance. Table 6 shows the obtained results when using the tags, co-commenting and social signals, compared to using only the tags and co-commenting signals. The results are reported for the BPR loss function, which achieved the best results for the Newsvine dataset (in accordance with the previous subsection). As can be seen from the results, the improvement in prediction due to the addition of the social signal is negligible: 0.1% when using the AUC metric and 0.4% when using the stratified-AUC metric.

Signals	AUC	sAUC
Tags & Co-Commenting	0.822	0.774
Tags & Co-Commenting & Social	0.823	0.777

Table 6: Performance effect of adding the social signal on the Newsvine dataset.

Obviously, our inability to extract strong signals from the social data is not a proof that such signals does not exist. Furthermore, the above experiment was done on a single dataset. However, with those grains of salt in mind, we posit the following:

- Commenting activity indicates very strong engagement and passion about a story, and friendship (as reflected in social networks) is not a strong indication of the friends sharing that level of topical passion.
- Affinity trumps friendship - collaborative filtering techniques that (implicitly) identify like-minded users from the population at large, are better at representing one's passions than one's (explicit) selection of friends. Note that one's selection of friends is not task-oriented, i.e. people do not mark online friendships for information filtering purposes. Thus, the expressed friendships are not tuned to compete against task-oriented CF algorithms. Still, when unwillingly competing in the information filtering race, they lose.
- Related to the previous point, it could very well be that the major attraction of a social networks in a site like Newsvine lies in one's ability to keep informed of friends' activities and interests, without necessarily sharing those same interests (or at least not at the same level of passion). In other words, one may appreciate knowing that a friend has commented on a story, even if that story is of low personal interest.

6. CONCLUSIONS & FUTURE WORK

In this study, we propose a method for personalized recommendation of news stories for commenting. This is achieved by predicting the stories that are most probable for commenting by a given user. We combine content-analysis (utilizing stories textual tags) together with collaborative filtering (utilizing users' co-commenting patterns) in a memory based and latent factor model (LFM) approaches.

With respect to the application domain, it is shown that when using 5 given commenters, co-commenting provides a signal that is equivalent or higher than the textual tags signal. Furthermore, combining both signals improves on what each signal achieves alone. Moreover, prediction accuracy grows with the number of comments available per story. Finally, our attempts to combine a third signal, that of social relationships among users, did not result in an observable gain in prediction accuracy.

With respect to the art of recommendation systems, we exemplify how one can improve the accuracy of latent factor models by tailoring the loss function to the specific application at hand, rather than using the standard squared error loss function. Moreover, we demonstrate the bias of the AUC metric in cases where one can easily identity items that are not suitable for most users. Therefore, we suggest a variation of the AUC metric, to which we call *stratified AUC*. The stratified-AUC metric compares the ranking of items in bins corresponding to their global popularity.

We plan to deploy our model on live traffic data, which will allow a more realistic evaluation of the methods described in this paper. Such an online setting will bring us new challenges and opportunities. It will necessitate the replacement of the static train-test split by a more realistic incremental evaluation, where the amount of information per story gradually increases over time. Such an evaluation will incorporate the temporal popularity of stories and the aging of old stories. No less important is the opportunity to better identify stories regarded as "negative" for the user. As previously explained, we currently draw negative examples from those stories on which the user has not commented. In the future we will be exposed to additional signals allowing us a more refined identification of negative stories. We would like to discard stories which were never presented to the user (e.g., she was inactive during their serving time). Then, we should distinguish between stories presented to user but not viewed (clicked) by her and stories viewed by the user without initiating commenting activity. Such considerations impact not only the evaluation procedure, but also the modeling methodology. Another aspect that was not addressed in this work is the discrimination between "types" of commenters; while many users have a rich commenting history, a large portion of the users rarely comment and their commenting history is very poor. A future research may want to examine the possibility of modeling users differently according to their commenting habits. In additions, our preliminary attempts to employ friendship connections to the recommendation model promote further investigation of the benefit of using social signals vs. users' behavioral similarity.

7. ACKNOWLEDGMENTS

We would like to acknowledge Michal Aharon, Yoad Lustig and Yoelle Maarek for their contribution to this work in many fruitful discussions.

8. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng.*, 17(6):734–749, 2005.

[2] D. Agarwal, B.-C. Chen, and B. Pang. Personalized recommendation of user comments via factor models. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 571–582, July 2011.

[3] B.-C. Chen, J. Guo, B. Tseng, and J. Yang. User reputation in a comment rating environment. In *Proceedings of the 17th international conference on Knowledge discovery and data mining (KDD 2011)*, pages 159–167, August 2011.

[4] A. S. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 271–280, New York, NY, USA, 2007. ACM.

[5] G. Dror, Y. Koren, Y. Maarek, and I. Szpektor. I want to answer; who has a question?: Yahoo! answers recommender system. In *KDD*, pages 1109–1117, 2011.

[6] T. Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861–874, June 2006.

[7] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35:61–70, December 1992.

[8] C.-F. Hsu, E. Khabiri, and J. Caverlee. Ranking comments on the social web. In *Proceedings of the 2009 International Conference on Computational Science and Engineering-Volume 04*, pages 90–97, August 2009.

[9] M. Hu, A. Sun, and E.-P. Lim. Comments-oriented document summarization: understanding documents with readersâĂŹ feedback. In *Proc. of ACM SIGIR 08*, pages 291–298, July 2008.

[10] Y. Koren, R. M. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30–37, 2009.

[11] B. Li, S. Xu, and J. Zhang. Enhancing clustering blog documents by utilizing author/reader comments. In *In Proceedings of the 45th Annual Southeast Regional Conference, 2007, Winston-Salem, North Carolina*, pages 94–99, March 2007.

[12] Q. Li, J. Wang, Y. P. Chen, and Z. Lin. User comments for news recommendation in forum-based social media. *Information Sciences*, 180(24):4929–4939, December 2010.

[13] S. Li and S. Manandhar. Improving question recommendation by exploiting information need. In *ACL*, pages 1425–1434, 2011.

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

[15] C. D. Manning, P. Raghavan, and H. Schïć Ţtze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[16] G. Mishne and N. Glance. Leave a reply: An analysis of weblog comments. In *Proceedings of the 15th international conference on World Wide Web*, pages 625–632, May 2006.

[17] B. Pang and L. Lee. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135, 2008.

[18] S. Park, M. Ko, J. Kim, Y. Liu, and J. Song. The politics of comments: Predicting political orientation of news stories with commenters‘ sentiment patterns. In *Proceedings of ACM Conference on Computer Supported Cooperative Work (CSCW) 2011*, pages 113–122, March 2011.

[19] P. A. Pavlou and A. Dimoka. The nature and role of feedback text comments in online marketplaces: Implications for trust building, price premiums, and seller differentiation. *Information Systems Research*, 17(4):392–414, December 2006.

[20] I. Pilászy and D. Tikk. Recommending new movies: even a few ratings are more valuable than metadata. In *Proc. of the 3rd ACM Conf. on Recommender Systems (RecSys'09)*, page 93100, New York, NY, USA, October 23–25 2009. ACM, ACM.

[21] H. Rangwala and S. Jamali. Co-participation networks using comment information. In *Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media*, pages 315–318, May 2010.

[22] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proc. 25th Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 452–461, 2009.

[23] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors. *Recommender Systems Handbook*. Springer, 2011.

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

[25] A. Schuth, M. Marx, and M. de Rijke. Extracting the discussion structure in comments on news-articles. In *WIDM*, pages 97–104, November 2007.

[26] S. Siersdorfer, S. Chelaru, W. Nejdl, and J. S. Pedro. How useful are your comments? analyzing and predicting youtube comments and comment ratings. In *roceedings of the 19th international conference on World Wide Web*, pages 891–900, April 2010.

[27] G. TakŒŚcs, I. Pilászy, B. NŒŹmeth, and D. Tikk. On the gravity recommendation system. In *KDD Cup Workshop at SIGKDD 07*, pages 22–30, August 2007.

[28] M. Tsagkias, W. Weerkamp, and M. de Rijke. Predicting the volume of comments on online news stories. In *Proceeding of the 18th ACM conference on Information and knowledge management (CIKM)*, pages 1765–1768, November 2009.

[29] M. Tsagkias, W. Weerkamp, and M. de Rijke. News comments: Exploring, modeling, and online prediction. In *Advances in Information Retrieval, 32nd European Conference on IR Research, ECIR 2010. Proceedings*, pages 191–203, March 2010.

[30] J. Weston, S. Bengio, and N. Usunier. Large scale image annotation: Learning to rank with joint word-image embeddings. *Machine Learning Journal*, 81:21–35, 2010.

[31] W. G. Yee, A. Yates, S. Liu, and O. Frieder. Are web user comments useful for search? In *7th Workshop on Large-Scale Distributed Systems for Information Retrieval (LSDS-IR) at SIGIR 09*, pages 63–70, July 2009.