Short and Tweet: Experiments on Recommending Content from Information Streams

Jilin Chen[*], Rowan Nairn[†], Les Nelson[†], Michael Bernstein[Δ], Ed H. Chi[†]

[*] University of Minnesota
200 Union Street SE,
Minneapolis, MN 55455
jilin@cs.umn.edu

[†] Palo Alto Research Center
3333 Coyote Hill Road, Palo Alto,
CA 94304
{rnairn, lnelson, echi}@parc.com

[Δ] MIT CSAIL
32 Vassar Street, Cambridge,
MA 02139
msbernst@mit.edu

ABSTRACT
More and more web users keep up with newest information through information streams such as the popular micro-blogging website Twitter. In this paper we studied content recommendation on Twitter to better direct user attention. In a modular approach, we explored three separate dimensions in designing such a recommender: content sources, topic interest models for users, and social voting. We implemented 12 recommendation engines in the design space we formulated, and deployed them to a recommender service on the web to gather feedback from real Twitter users. The best performing algorithm improved the percentage of interesting content to 72% from a baseline of 33%. We conclude this work by discussing the implications of our recommender design and how our design can generalize to other information streams.

Author Keywords
Information stream, recommender system, topic modeling, social filtering.

ACM Classification Keywords
H.5.3: Group and Organization Interfaces.

General Terms
Algorithms, Experimentation

INTRODUCTION
Information streams have recently emerged as a popular means of information awareness. By *information streams* we are referring to the general set of Web 2.0 feeds such as status updates on Twitter and Facebook, and news and entertainment in Google Reader or other RSS readers. Although they have notable differences, the above examples share two key commonalities: (1) they deliver to each user a stream of text entries over time that are personalized to the user's subscriptions, and (2) they allow users to explicitly interact with each other. As information distribution platforms, Twitter, Facebook and Google Reader have all enjoyed great popularity and are drawing ever more new users into them. For instance, according to compete.com's traffic statistics, the total number of people visiting Twitter has been rising from about 6 million per month in January 2009 to over 23 million per month as of July 2009 (http://siteanalytics.compete.com/twitter.com/).

With an abundance of information comes the scarcity of attention [20]. Two user needs arise from attention scarcity: *filtering* and *discovery*. On the one hand, a user's stream will often receive hundreds of items each day, much beyond what users have time to process. Users would like to filter the stream down to those items that are indeed of interest. On the other hand, many users also want to discover useful content outside their own streams, such as interesting URLs on Twitter posted by friends of friends, or relevant blogs in Google Reader that are subscribed by other friends. This discovery task is formidable, given the vast amount of information that are disseminated daily through information stream services.

One approach is to proactively recommend interesting content to users so as to better direct their attention. Google Reader has implemented a discovery feature that recommends interesting RSS feeds, and a number of third-party websites provide filtering or recommendation services for Twitter users. So far there has been little discussion regarding the effectiveness of such solutions, and little is known regarding the design space of information stream recommenders.

As a domain for recommendation, information streams have three interesting properties that distinguish them from other well-studied domains:

(1) Recency of content: Content in the stream is often considered interesting only within a short time of first being published. As a result, the recommender may always be in a "cold start" situation [19], i.e. there is not enough data to generate a good recommendation.

(2) Explicit interaction among users: Unlike other domains where users interact with the system as isolated individuals, with information stream users explicitly interact by subscribing to others' streams or by sharing items.

CHI 2010, April 10–15, 2010, Atlanta, Georgia, USA.

(3) User-generated content: Users are not passive consumers of content in information streams. People are often content producers as well as consumers. Micro-blogging software such as Twitter and Facebook status updates are prominent examples.

In this paper we describe our design and empirical studies of a recommender system built on top of Twitter, called zerozero88, which recommends URLs that a particular Twitter user might find interesting. The recommender we developed is publicly available at www.zerozero88.com.

We chose Twitter as our target platform for several reasons, most importantly because it shares all the common features of information streams described earlier. As a successful platform, Twitter also provides a chance to recruit real users and alleviate their real attention scarcity problems. Finally, Twitter provides a set of public APIs, enabling us to implement and deploy our recommender. We chose to focus on recommending URLs, because the URL represents a common 'unit' of information on the web, and previous research has identified sharing URLs and reporting news as common uses of Twitter [9].

We wish to investigate:

(a) Whether recommender systems can help users find interesting content on Twitter?

(b) What elements lead to an effective Twitter-based recommendation? How can this understanding inform recommender design for other information streams?

To achieve our research goals, we first conducted pilot interviews to elicit early qualitative feedback and refine our system design. After implementing the system, we conducted a controlled field study on our web service to gather quantitative results.

The rest of the paper is structured as follows. First, we discuss how existing research relates to our work. We then provide an overview of information production and information seeking practices on Twitter. We describe the design space of our recommender, and then detail our studies and the results. We conclude with discussions of our findings that may generalize to other information streams.

RELATED WORK

Recommenders as a solution to attention scarcity have been studied for years. Perhaps the most well-known approach is collaborative filtering (CF), which recommends items (such as news stories) using similarities of preferences among users [10]. This approach does not rely on the content of items, but instead requires users to rate items to indicate their preferences, and infers preference similarity from the overlap of rated items across users.

CF recommenders commonly suffer from little user rating overlap early on, known as the "cold-start" problem; a common solution is to use other information like the textual content of the items to be recommended [4, 19].

There is a wealth of research on recommenders that utilize the content of items. Such recommenders are often used in domains where extensive textual content is available for items, such as websites [14] and books [13]. For example, to recommend websites, Pazzani et al. first created bag-of-word profiles for individuals from their activities and then chose websites most relevant to the profile of the individual as recommendations [14]. Because activities of an individual are often insufficient for creating useful profiles, Balabanovic et al. proposed to create profiles not from an individual's activity but from a group of related individuals [4]. This work can be viewed as a hybrid of collaborative filtering and content-based approaches [12].

Recommendations can be generated from explicit social information and social processes as well. For example, Hill et al. described a social filtering recommender on Usenet newsgroups [8]. For each newsgroup, they recommended the most frequently mentioned URLs to that group. Andersen et al. proposed the concept of a trust-based recommender [2]. From a theoretical perspective they discussed ways to employ users' opinions toward other users to compute recommendations. Several other papers investigated the possibility of using social network structures for recommendation [5, 7]. For example, Chen et al. recommended friends-of-friends as potential friends to users of a social networking site, and showed that this scheme is accepted more often than recommending people sharing common keywords [5].

Prior research in developing scalable recommenders [6, 15, 18] is also relevant here because the Twitter ecosystem is so huge that many otherwise useful algorithms become intractable. For example, Sarwar et al. applied clustering algorithms to partition user population, built neighborhoods for users from the partition, and considered only those neighborhoods when computing recommendations [18]. Another relevant work integrated distributed computation techniques for recommendation in Google News [6]. These techniques recursively chop a full problem into sub-problems, so that in the end they can utilize all information in the system despite the large scale of the data.

Outside of academic research, several start-up companies provide information stream filtering or recommendation services, such as my6sense.com, feedafever.com, and MicroPlaza.com. Both my6sense and feedafever filter RSS feeds, including Twitter streams. MicroPlaza recommends personalized news for Twitter users. As start-ups, none of them disclose their approaches or benchmarks.

Because Twitter has both textual and social information available, key parts of the past work described above may be applicable for a Twitter recommender. However, most of them have not yet been implemented and evaluated on Twitter or information streams in general. As a result, it is unclear whether these techniques function well given the differences between their original domains and Twitter, or if some techniques need to be changed to fit the needs of

Twitter users. Our work not only depict the design space for a Twitter recommender, but also better inform designers of recommenders for other information streams.

INFORMATION PRODUCTION & SEEKING ON TWITTER

Twitter describes itself as a micro-blogging service. Users of the site can post short messages, each up to 140 characters, commonly known as tweets. As information producers, people post 'tweets' for a variety of purposes, including daily chatter, conversation, sharing information/URLs and reporting news [9]. Other information streams may have different dominating purposes for posting. For example, on Facebook most of status updates are daily chatter and conversation, while a majority of blog posts in Google Reader may be for information sharing.

As an information seeker, each Twitter user sees a tweet stream when visiting Twitter. A new account only includes tweets posted by one's self; one can include another user's tweets by *following* that user. Throughout this paper, whenever user A follows user B, we refer to A as B's *follower*, and B as A's *followee*.

While some might refer to their followees as their "friends", the following relationship on Twitter is not reciprocal, and does not necessarily imply friendship or even acquaintance between two users. For example, over two million users follow Barrack Obama, few of whom he follows back. Obviously, those people follow President Obama because they are interested in what he says, not because they are personal friends with him. This mechanism of following is different from friendship in other sites such as Facebook, where connections between people are always reciprocal and require confirmation from both sides.

A typical Twitter user picks a list of followees by hand and monitors her personal stream over time. People can also discover information outside their stream in a number of ways, including typing the username of an arbitrary user to see her stream, checking the most popular topics across the whole Twitter site, searching for tweets over the whole Twitter site by keywords, or using one of many third party services that support exploration on Twitter.

DESIGNING RECOMMENDERS FOR TWITTER

We form our design space into three dimensions: (1) how to select candidate URLs, (2) how to use content information, and (3) how to use social information. We illustrate the full design space in Table 1, where each cell is a possible design choice we can make in one of the three dimensions.

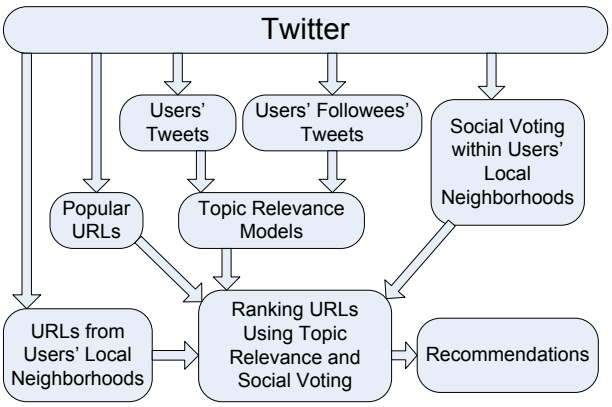

Figure 1. Conceptual Model of the Whole Recommender

We discuss each dimension in the following subsections. Then, we will elaborate on possible system designs and articulate design questions that we answer through empirical studies. The conceptual model of the system that we built is shown in Figure 1.

We did not consider collaborative filtering in our design, as this would require each URL to have feedback from several users to compute reliable recommendations. Moreover, the real-time value of URLs on Twitter requires recommenders to consider new URLs as soon as possible. Under those two constraints, in order to obtain enough feedback for URLs before they become too old to be valuable, the recommender needs a large volume of real-time usage data, as demonstrated in the Google News recommender [6]. However, since we do not have access to large amounts of usage data, this is not a viable option for us. As a result, in formulating our design space, we focused on using content of the tweets and information from social processes.

Selecting the Candidate Set

In building our Twitter based URL recommender, we must first select a limited candidate set of URLs for recommendations due to the high volume of tweets on Twitter. According to TweeSpeed.com, as of September 2009, the number of tweets sent per hour on Twitter ranges from 400,000 to 1,400,000. Scanning those tweets for URLs in real time is a technical challenge. Given limited access to tweets and processing capabilities, our first design question is how to select the most promising candidate set of URLs to consider for recommendations.

Our problem of selecting a candidate set of URLs bears similarities to prior work on scalable recommenders [15,

Design Dimension	Possible Design Choices		
***CandidateSet**: Selecting Candidate Set*	**FoF** (followee-of-followees)	**Popular**	
***Ranking-Topic:** Ranking Using Topic Relevance*	**Self-Topic** score	**Followee-Topic** score	**None**
***Ranking-Social**: Ranking Using Social Voting*	**Vote** score	**None**	

Table 1. The Design Space of the Recommender, Spanning 2x3x2=12 Possible Algorithm Designs

18], because they encountered the same challenge of not being able to process the full dataset due to its scale.

In particular, Sarwar et al. [18] have shown that by considering only a small neighborhood of people around the end user, we can reduce the set of items to consider, and at the same time expect recommendations of similar or higher quality. While Sarwar et al. built the neighborhood based on similarity in preferences, for a Twitter user we limit our consideration to her social neighborhood: for a user Alice, we consider only URLs posted by her followees and followees of followees.

This scheme makes sense intuitively on Twitter as well: Imagine Alice follows Bob. In doing this, Alice has treated Bob as a promising information source. As a result, it is reasonable to assume that Alice's interest in URLs from Bob and people that Bob considers promising should be higher than URLs from a random stranger on Twitter. This comes from the principle of *locality*.

A second intuition is the popularity of URLs: URLs that are posted all over Twitter are probably more interesting than those rarely mentioned by anyone. Popular Twitter news website Tweetmeme.com operates with this intuition, where users can browse the most popular URLs in the last 24 hours or in the last week on Twitter. This approach yields an alternative way of choosing the candidate set: popular URLs on Twitter. We use the public API from Tweetmeme to gather such URLs.

In summary, we decided to consider two approaches in selecting candidate sets of URLs, referred as *FoF* (followee-of-followees) and *Popular*. Because URLs posted on Twitter are usually highly interesting only within a small timeframe, we further limit our consideration to URLs created within the last 7 days.

Ranking URLs Using Topic Relevance

Using topic relevance is an established approach to compute recommendations [4, 5, 7, 12, 13, 14]. The topic interest of a user is modeled from text content the user has interacted with before, and candidate items are ranked by how well they match the topic interest profile of the user.

Following the approach in Pazzani et al. [14], we build a bag-of-words profile for each Twitter user. Unlike in Pazzani et al., where the profile consists of words from web pages that the user has rated explicitly, here we build the profiles from words that users have included in their tweets.

The detail of this approach is as follows: We extract and stem words from all tweets we collected, and then filter them through a standard stop word list. Then for each user u we create a profile – a vector $V_u = (v_u(w_1), \ldots, v_u(w_m))$, where m is the total number of distinct words in all tweets, and each $v_u(w_i)$ describes the strength of u's interest in word w_i. The value of $v_u(w_i)$ is calculated using a term-frequency inverse-user-frequency weighting scheme (TF-IDF) [17]:

$TF_u(w_i)$=(frequency of w_i in u's tweets)

$IDF_u(w_i)$=log[(#all users)/(#users using w_i at least once)]

$v_u(w_i) = TF_u(w_i) \cdot IDF_u(w_i)$, and then normalized so that the norm of V_u is 1.

Intuitively, high TF of a word means that the user mentions the word frequently, indicating higher interest, while high IDF of a word means that few other users mention this word, indicating that the word can better distinguish one user from other users.

This approach builds u's profile from u's own tweets, which we later refer to as u's *Self-Profile*. It assumes that u's interest can be modeled by what u talks about, and thus captures u's interest as an information producer.

However, u's Self-Profile may not capture u's interest as an information seeker, for u may follow many different other users. For example, u may tweet only about HCI research, but follow people not only for HCI research but also pop music. In this case, u's Self-Profile will capture HCI research, but miss pop music completely.

To capture u's interest as an information seeker, we build another profile for u, referred to as u's *Followee-Profile*, by combining the Self-Profiles of u's followees. Prior works [4, 12] have demonstrated the effectiveness of combining text content from a user group to capture the interest of single user, although their motivation is to solve the cold-start problem and the data sparsity problem and not to model a different type of interest.

We build u's Followee-Profile as follows: For each of u's followees f, we denote f's Self-Profile vector as V_f. We pick all words that f has mentioned at least once, rank them by decreasing order of their v_f in V_f, select the top 20% of words in the ranked word list, and then remove words that none of u's other followees mention.

We call the resulting set of words f's *high-interest words*, because intuitively they are the words that f is most interested in as information producers. We remove words that only f tweets about because otherwise many incidental words that only f cares about would be included, bringing in too much noise into the model.

We then compute u's Followee-Profile from the high-interest words of u's followees. u's Followee-Profile takes the same form as Self-Profile, but with a different TF value, denoted as $FTF_u(w_i)$=(# u's followees who have w_i as their high-interest words).

Intuitively, high FTF of a word in u's Followee-Profile means that many of u's followees commonly tweet using the word. Thus, by modeling from salient words used by people that u decides to follow, u's Followee-Topic captures u's interest as an information seeker.

The topic of a URL can also be modeled as a word vector. Its formulation is the same as Self-Profile of a user, except that in this case the TF is the number of times a word has been used to describe the URL in tweets. Intuitively, the more often a word has been used to describe a URL, the more likely the word is relevant to the URL. This approach has the benefit that the topic of the URL can be modeled independently from the actual web page content. Ignoring the web page content enables us to model the topics of URLs that contain little reliable textual content in themselves, such as URLs of images and videos (e.g., TwitPic and TwitVid). In the case that a URL is only mentioned in a small number of tweets, we employ an additional term expansion technique to obtain more related words for the URL, in an approach similar to what has been used in Sahami et al. [16].

Given the topic profile vector for a user (either Self-Profile or Followee-Profile) and the topic vector for an URL, we compute the cosine similarity between the two vectors as the topic relevance score between the user and the URL. Given the score, we then recommend the URLs with highest scores. Relevance ranking with cosine similarity is commonly used in information retrieval, and has been used for recommenders as well [14].

We refer to the topic relevance score using Self-Profile as *Self-Topic*, and the score using Followee-Profile as *Followee-Topic*. Intuitively, a high Self-Topic score means that the URL matches the user's interest as information producer, while a high Followee-Topic score means that the URL matches the user's interest as information seeker.

Ranking URLs Using Social Process

We draw insight from Hill et al. [8] to utilize social processes for recommendation. Hill et al. described a social filtering system that recommends news URLs on Usenet newsgroups. The system works like a within-group popular vote: in each group (e.g. *comp.software*), it recommends most popular URLs on a "one person, one vote" basis – the more people in the group who mention a URL, the more likely the URL will be recommended.

This approach is easily adapted to Twitter, by replacing the notion of a newsgroup with a user's followee-of-followees neighborhood. Assuming the user has a stable interest and follows people according to that interest, people in the neighborhood should be similar minded enough so that voting on the neighborhood can function effectively just like within a Usenet newsgroup of a specific topic.

However, the "one person, one vote" basis in the approach above may not be the best design choice in Twitter, because some people may be more trustworthy than others as information sources. Andersen et al. discussed several key insights in their theory of trust-based recommender systems [2], one of which is trust propagation. Intuitively, trust propagation means my trust in Alice will increase when the people whom I trust also show trust in Alice. Following this argument, a person who is followed by many of a user's followees is more trustworthy as an information source, and thus should be granted more power in the voting process.

Another intuition on Twitter regards the frequency with which a person tweets. Some people may post chatter or a fun video every hour, while others may only post when they feel the information is truly worthwhile to share. We thus weigh people differently based on their tweet frequency, and grant people who tweet less frequently more vote power. This design intuition has been noted in the interviews from several Twitter users in a pilot study.

We then define our weighted voting process as follows: For a user u, the vote score of a URL is the total vote power of all u's followee-of-followees who have mentioned the URL. The vote power of a followee-of-followee f is defined to be proportional to the logarithm of the number of u's followees who follow f, and also proportional to the logarithm of the average time interval between f's consecutive tweets.

If a URL has never been mentioned by any followee-of-followees, its vote score is as if it was mentioned by a single person with the lowest possible voting power.

We refer the vote score computed above simply as *Vote*. We pick URLs with high *Vote* scores as recommendations.

Putting Everything Together

We have described two methods for selecting candidate URLs, two methods of using topic relevance to rank, and one method of using social process to rank. We can decide which method to use in each of those dimensions separately, and can choose to use no topic relevance or no social process as well. As a result, there are in total 2 (candidate URLs) x 3 (topic relevance) x 2 (social process) = 12 possible algorithm designs, as illustrated in Table 1.

Every one of those 12 algorithms follows a paradigm of "choose and rank" – the system first chooses a candidate set, and then ranks URLs within the set by a single score. If we use only topic relevance or social process, then the ranking score is the output of that dimension alone. If we use both topic relevance and social process (i.e. Self-Topic with Vote or Followee-Topic with Vote), we use the product of the two scores to rank. Finally, if neither is used (i.e. None with None), we choose URLs randomly from the candidate set.

We implemented all 2x3x2=12 algorithms in the design space so that we could compare the algorithms side by side and investigate the effect of each design choice. Having formulated the design space, we expand our two research goals stated in the introduction section into the following five research questions, thus approaching our research goals through quantitative studies:

Q1. Do the approaches of ranking using topic relevance help at all, and if yes, which one is better?

Q2. Does ranking using social voting process help?

Q3. Which source of candidate URLs is better?

Q4. If both topic relevance ranking and social voting process help, do their benefits complement each other?

Q5. Among all 12 algorithms, which one performs the best?

EMPIRICAL STUDIES

We first conducted pilot interviews to elicit early qualitative feedback and refine our system design. We then conducted a controlled field study to gather quantitative results.

Pilot Interviews

We invited a small sample of four active Twitter users in our research organization to participate in in-person interviews. Of the four subjects, three were male, one was female, and all were in their 20s or 30s. Occupations ranged from full-time employee to contractor to summer intern.

The interview was split into two parts, for a total of 30-60 minutes per subject. In the first part, we asked subjects how they choose people to follow, how they decide which URLs to click on when using Twitter, and whether they use Twitter as a way to track news and current events. In the second part, we showed subjects the recommendations from several algorithms, explained the differences in the algorithms at a very high level (e.g. this algorithm selects URLs from people you follow and use topic relevance to recommend, etc.), and asked them to give feedback on a variety of algorithms.

All interviews were audio-taped and later transcribed. The first half of the interviews confirmed a number of our key design intuitions, including those regarding topical relevance, social voting, and the particular weighting scheme we used in the social voting process. The second half of the interview helped shape the UI of the system. The interviews indicated the trade-off between relevance and serendipity in recommender design, which we will elaborate in the discussion section.

Controlled Field Study

We conducted a field experiment on our publicly available recommender website – zerozero88.com. We publicized the site as a news recommendation service based on Twitter.

We recruited subjects through word-of-mouth on Twitter, where we simply asked people to try a new recommender designed for Twitter users. As such, all subjects were already Twitter users and none of them were paid. To make sure that the algorithms had enough data to compute recommendations, we required subjects to have at least 20 followees and 50 tweets.

We made the recommender service freely available. However, before qualifying subjects could use the service, we required them to rate our different recommendation algorithms for analysis. We also asked subjects to complete a brief questionnaire focused on the types of news they track on Twitter.

For each subject, each of the 12 algorithms independently

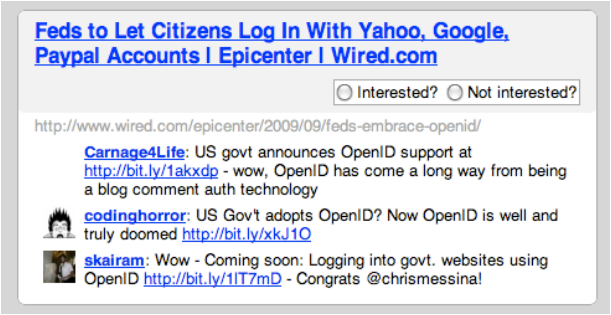

Figure 2. Recommendation Widget on zerozero88.com

recommended its five highest-ranked URLs. URLs recommended by the 12 algorithms were then combined and randomized. We displayed each URL within a recommendation widget (Figure 2) which shows the title, URL address, and up to three tweets that mentioned the URL to provide context. In choosing the three tweets to display, we preferred tweets from followees or followee-of-followees of the subject, if there were any.

Subjects rated each URL as either interesting or not. When algorithm A and B both recommend the same URL, we only showed one copy of the URL to the user. The user's rating for the URL was then reflected in the scores for both algorithm A and B, to ensure a fair comparison among all algorithms.

In the end, in the dataset for the following data analysis, every one of the 12 algorithms has exactly 5 binary rating samples per subject, with possible duplicated ratings for recommendations shared between algorithms. No algorithm was penalized due to duplication or ordering.

QUANTITATIVE RESULTS

We ran the field experiment for three weeks and collected results from 44 subjects. Our subjects tracked diverse types of news on Twitter, including local news, entertainment, and technology (Figure 3). In total, subjects produced ratings for 2640 (possibly non-unique) URLs, resulting in a dataset similar to that illustrated in Table 2.

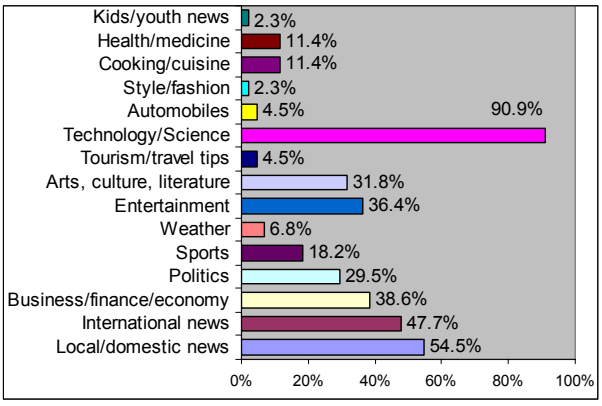

Figure 3. Percentages of Field Study Subjects who Use Twitter to Track Different Types of News

Subject	*Candidate Set*	*Ranking-Topic*	*Ranking-Social*	*Seq*	*Interest*
Alice	FoF	Self-Topic	Vote	1	1
Alice	FoF	Self-Topic	Vote	2	0
Bob	Popular	None	None	5	0

Table 2. Format of Dataset Gathered in Controlled Field Study

In the dataset, each row represents a rating of URL from a subject. CandidateSet, Ranking-Topic and Ranking-Social encode the algorithm that generates the URL, and Seq distinguishes the 5 URLs for a single subject-algorithm pair. Interest encodes a binary interest rating from the subject (interested/ not interested), where 1 means that the subject thought the URL was interesting.

We analyzed our dataset using logistic regression. Logistic regression uses input variables to predict the probability of a binary output – in our case, how likely the interest value would be 1 given a specific algorithm design. For regression problems with binary outputs as in our case, logistic regression is superior to commonly used ordinary linear regression with ANOVA [1].

We use the GENMOD procedure in SAS to perform logistic regressions in our study [1]. GENMOD has the ability to model correlated data, which is necessary for our dataset: URL ratings are nested within subjects and should thus be assumed correlated. Further, because rating samples of two algorithms will be identical if both samples are obtained from the same subject regarding the same URL, we have introduced additional within-subject correlation to the dataset. GENMOD can estimate significance of factors using either Wald tests or score tests. The two kinds of tests agreed qualitatively throughout our study, and as a result we report only Wald test results.

Research Question 1, 2 and 3: Main Effects

To answer questions 1, 2, and 3, we build Model 1, which predicts the probability of Interest being 1 (the subject being interested in the URL) using CandidateSet, Ranking-Topic and Ranking-Social as factors. This regression tells us whether the probability of generating interesting URLs would change significantly if we change our choice on one design dimension, such as changing the way of selecting candidate sets from Popular to FoF, or changing the way of ranking by topic relevance from None to Self-Topic.

In Model 1, we found a significant increase in the probability when changing Ranking-Topic from None to Self-Topic (beta=0.58, Z=4.95, p<.001), and to Followee-Topic (beta=0.27, Z=2.48, p=.01). The increase from Self-Topic is larger than from Followee-Topic (0.58 vs. 0.27), and this difference is significant (Chi-sq(1)=14.89, p<.001). These results answer Q1: Both ways of ranking URLs using topic relevance help, and using Self-Topic works better than using Followee-Topic.

We also observed a significant increase in the probability when changing Rank-Social from None to Vote (beta=1.02, Z=6.53, p<.001). This answers Q2: Using social voting process indeed helps.

We observed an increase in the probability when changing CandidateSet from Popular to FoF; the increase evinced a trend but was not significant (beta=0.22, Z=1.78, p=.08). This answers question 3: FoF might be working better than Popular.

Research Question 4: Interaction Effects

To answer question 4, we built Model 2 by adding an interaction term between Ranking-Topic and Ranking-Social into Model 1. Model 2 can tell us whether the increase caused by using a topic relevance approach is dependent on whether social voting is used or not. With the interaction effect added, the estimated beta coefficients of other factors in Model 2 varied from Model 1, but no change in sign or level of significance happened.

We observed in Model 2 a significant negative interaction effect for both Self-Topic * Vote (beta= -0.76, Z= -4.04, p<.001) and Followee-Topic * Vote (beta= -0.39, Z= -2.19, p=.03). This indicates a diminished return between each pair of combinations, i.e. the benefits by having both topic relevance ranking and social voting process are smaller than the sum of the benefits they have individually.

We can quantitatively estimate the degree in which the benefit diminishes from beta values in Model 2. All beta values in logistic regressions can be transformed into odds-ratio effects, i.e. how much more likely the algorithm would produce interesting URLs than non-interesting ones.

For example, in Model 2 the beta of Vote for Ranking-Social is 1.40, which means that adding social voting process alone has an odds-ratio of exp(1.40) = 4.06, i.e. the system is 4.06 times likely than before to generate interesting URLs than non-interesting ones. Similarly, using Self-Topic ranking alone has an odds-ratio of 2.59.

However, when we combine Self-Topic with Vote, because the two have a significant interaction effect of odds-ratio exp(-0.76) = 0.47, the combined odds-ratio is 4.06 * 2.59 * 0.47 = 4.94, instead of 4.06 * 2.59 = 10.52 if the two were independent. In terms of odds-ratio, a Self-Topic plus Vote combo is 90% better than using Self-Topic alone and 22% better than using Vote alone.

Repeating the above process between Vote and Followee-Topic in Model 2, we found that their combined odds-ratio is 4.06 * 1.60 * 0.68 = 4.41. Given that the odds-ratio of Vote alone is already 4.06, adding Followee-Topic on top of Vote provides less than 10% additional benefit in the odds-ratio of recommending interesting URLs.

Research Question 5: Best Performing Algorithm

To answer question 5, we built Model 3, which predicts the probability of interest using CandidateSet, Rank-Topic and Rank-Social combined as a single factor. This allows us to compare all 12 algorithms individually side by side.

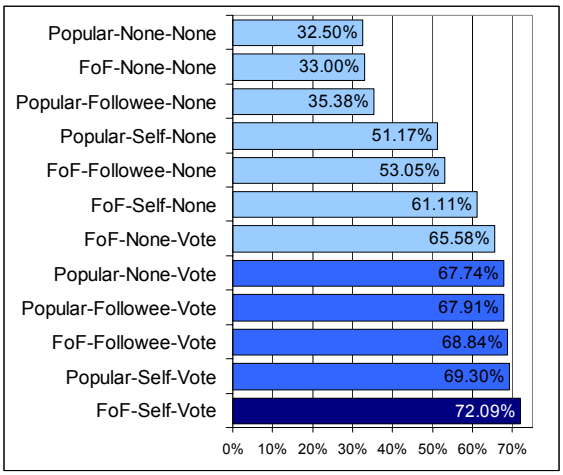

Figure 4. Percentage of Interesting URLs Recommended per Algorithm. FoF-Self-Vote at the bottom is significantly better than the top 7 algorithms in the list

The result from Model 3 suggests that the combination of FoF on CandidateSet, Self-Topic on Ranking-Topic and Vote on Ranking-Social has the highest probability of producing interesting URLs. Using p-value < 0.05 as the cut-off, this algorithm is statistically indistinguishable from Popular-Self-Vote, FoF-Followee-Vote, Popular-Followee-Vote and Popular-None-Vote. Nevertheless, it is significantly better than the other 7 algorithms.

Figure 4 illustrates this result by showing the percentage of interesting URLs produced by each of the 12 algorithms.

DISCUSSION

Understanding the Effectiveness of Our Approach

We summarize our results qualitatively in Table 3. We have found favorable results regarding the effectiveness of recommenders. In the best performing algorithm, our Twitter recommender can provide up to 72% interesting items. Modeling the topic interest of users and leveraging a social voting process were both beneficial for recommending URLs on Twitter.

The degree of such benefits can be understood more intuitively from several reference points in Figure 4. For example, Popular-None-None recommends URLs randomly from most popular URLs on Twitter. Looking at its recommendations would be close to looking at the front page of a popular URL aggregator such as Tweetmeme. As Figure 4 suggests, on average the chance that a URL there would be interesting is 32.5%.

Likewise, looking at URLs recommended by FoF-None-None is more or less similar to scanning one's own stream and streams of followees. There is a 33.0% chance that a URL in there would be interesting to read.

Using ranking algorithms can greatly increase this chance. Generally speaking, all of the Vote-based algorithms outperformed non-Vote-based algorithms. The best performing algorithm using only topic relevance ranking is FoF-Self-None, which improves the chance to 61.1%.

The best performing algorithm overall is FoF-Self-Vote. It recommends interesting URLs 72.1% of the time, more than doubling the chance compared to cases where no ranking is performed.

We also examined transcriptions from the interview study and found explanations for improvements from the user's point of view. For example, with respect to Social Voting, one participant reported: "The fact that a few different people retweet it may make it more likely [to be interesting.]" In this case, the interviewee is following the tweets of others, and he particularly pays attention when multiple others in his stream are agreeing.

Generalizability of Our Approach

We believe that our algorithms are general enough that they can function and provide benefits not only in Twitter but also in other information streams, because fundamentally the algorithms assume little that is specific to Twitter: (1) Our model of topic interest is widely used in other domains, and only requires that users or their friends send and receive text updates. (2) Our social voting process requires explicit social interaction between users, which is present in many different types of online services. (3) While in this study we recommend URLs in particular, we did not use the content of the pages at the URL. As such, our techniques can be used to recommend other content, such as images or videos. Therefore, one can adapt our system to recommend photos on Facebook or news stories on Google Reader.

Q#	Research Question	Answer
1	Is the Topic relevance helpful?	Yes, and Self-Topic (relevance to one's own tweets) is significantly better than Followee-Topic (relevance to followees' tweets).
2	Is the Social voting process helpful?	Yes.
3	How to selecting candidate set?	FoF (followee-of-followees) seems to be a bit better than Popular, but the difference is not significant.
4	How well topic relevance ranking and social voting process work together?	There is a diminishing return when combining the two approaches. Social voting is the biggest contributor in itself. On top of that, Self-Topic adds 22%, and Followee-Topic adds less than 10%.
5	Which algorithm seems best?	The best performing algorithm is FoF-Self-Vote (This algorithm selects URLs posted by followee-of-followees, and ranks them by both relevance to user's own tweets and social voting).

Table 3. Summary of Quantitative Results of the Controlled Field Study

Moving away from Web 2.0 websites, if members of an open source project use RSS feeds to track their work and send each other messages for coordination, our system can recommend them work items and bug reports. As another example, if we view emails sent over time as streams and view sending and receiving email as social interaction, we can adapt our system to recommend useful tips, proposals and meeting calls within an enterprise email system.

However, we would like to caution designers that, in other domains, the degree of benefits from algorithms might vary dramatically from this study. For example, in our study we found modeling user interests as information producers (Self-Topic) being superior to modeling their interests as information seekers (Followee-Topic). This result may be due to many Twitter users producing information actively but not following a coherent set of people with a single interest. If this were true, it would provide reliable information to the producer model and make the seeker model noisy. In other domains, the user behavior may be different and yield different results.

As another example, our social voting process assumes people subscribing to stream of a user based on how valuable the user is as an information source. Therefore, its results may become inferior in other domains where the semantics of subscription is different. For instance, in Facebook, subscribing to one's status updates is more likely due to friendship than to expected information value, so Facebook recommenders might benefit more by leveraging tie-strength instead of a voting process.

Distinguishing Algorithms: Relevance vs. Serendipity

We found no significant difference between the two ways of selecting candidate URL sets. Comparing the 12 algorithms individually, we found the bottom five algorithms in Figure 4 to be statistically indistinguishable from each other. However, this does not necessarily mean that those algorithms are the same, because the needs of users can be much more nuanced than what can be expressed in a simple binary rating.

Indeed, several subjects in our interviews mentioned a key distinguishing factor that they care about – whether the algorithm provides more relevance or serendipity. For example, one participant expressed that she wanted affirmation of what is already known or familiar as opposed to discovery of new or contrary ideas: "There is a tension between the discovery and the affirming aspect of things. I am getting tweets about things that I am already interested in. Something I crave when I am in a recommendation environment, where something is filtered, or brought to the surface, is an element of surprise or whimsy. That's one of the things I love about Twitter. Because in the RSS feeds you self-select things to get. I am getting a lot of things I am interested in, but that is not necessarily a good thing for me personally"; and she goes on to say, "I am also very interested when people take the opposite point of view from the 'mainstream'."

Almost every design choice inevitably moves the system towards either relevance or serendipity. For example, in our design, URLs from the local neighborhood (as in the FoF candidate set) are more similar to what users already see in their own streams, while global popular URLs (as in the Popular candidate set) may contain more surprises. Modeling interest using commonly used words in the local neighborhood (as in Followee-Topic) and using local social voting (as in Vote) both make recommendations relevant to locally prevalent interests, but rule out minority ideas. Finally, modeling interest using words that the user tweets about (as in Self-Topic) promotes content relevant to one's previous speech and buries everything else.

Because users care about the balance between relevance and serendipity, designers may want to research the preferences of users to tune the recommender accordingly. However, having such knowledge beforehand may be hard because preferences vary between users.

First, the user's preference may depend on the volume of the incoming stream. Some users already receive too much to read in their own stream, so that they want to filter away everything except the most relevant pieces. Other users may be more skilled at keeping up with their stream and thus value serendipity more. For example, among the four interview participants, the one with the highest number of followees and heaviest incoming tweet volume is the only one who uses third-party tools to filter his stream. In contrast, another participant only follows a few friends, has time to read every single tweet he receives and has no need for filtering at all. Instead, he finds random popular URLs in our system quite interesting to read.

Second, the user's opinion may depend on how he or she uses Twitter in comparison to alternative channels. Some users receive news solely through Twitter, and thus want more random discovery, while for others Twitter may serve a very specific information need, such as keeping up with current HCI research. One participant described, "[I am not interested] because those are just general news that I probably follow in some other way and I don't need to come to Twitter." Another concurred, saying "I am pretty good at finding articles I like to read already" – he feels less of a need for a recommender service.

Third, users' preferences may depend on the context in which information is received. For example, while at work a user may be very task-focused and only read information relevant to his or her job. But, at home that same user may be willing to entertain much broader interests, including YouTube videos completely unrelated to work. One of our interview participant remarked: "There are tweets that I am personally interested in and tweets that I have for information gathering and documentation and recordkeeping purposes." Similarly, another said: "There are situations where I would look for entertaining [URLs]."

One way to solve the problem without prior knowledge of user preference may be constructing a recommender using

many algorithms spreading over the spectrum of relevance vs. serendipity. The system gathers user feedback, and over time learns whether each user prefers algorithms that better support relevance or those that better support serendipity. Then for each user, the system uses a personalized subset of algorithms that the user prefers the most. The system may even learn preference changes and adapt accordingly, such as supporting more relevance at work and supporting more serendipity at home. In fact, zerozero88.com already uses a simple version of this idea: After users have rated recommendations from all algorithms in the sign-up process, they will receive more recommendations from algorithms they have rated high and fewer from algorithms they have rated low.

CONCLUSION AND FUTURE DIRECTIONS

In this paper we studied URL recommendation on Twitter as a means to better direct user attention in information streams. We implemented 12 algorithms in the design space we formulated, and through a controlled field study of 44 Twitter users demonstrated that both topic relevance and the social voting process were helpful in providing recommendations.

As mentioned earlier, while our algorithms are general and can be directly applied to many other information streams, domain-specific properties may have great impact on the effectiveness of these algorithms. Future research may explore other domains so as to deepen our understanding in the design space to finer details, add more design options, or add completely new dimensions to the design space.

The issue of serendipity has been raised in both web search [3] and recommender systems [11]. However, we are aware of little research in how this issue plays out in information stream recommenders. Further research in serendipity in this context may bring information stream users much richer experiences than they have now.

REFERENCES

1. Allison, P.D. 1999. Logistic regression using the SAS system: theory and application. SAS Institute.
2. Andersen, R., Borgs, C., Chayes, J., Feige, U., Flaxman, A., Kalai, A., Mirrokni, V., and Tennenholtz, M. 2008. Trust-based recommendation systems: an axiomatic approach. In *Proc of WWW '08*.
3. André, P., Teevan, J., and Dumais, S. T. 2009. From x-rays to silly putty via Uranus: serendipity and its role in web search. In *Proc of CHI '09*.
4. Balabanović, M. and Shoham, Y. 1997. Fab: content-based, collaborative recommendation. *Commun. ACM* 40, 3 (Mar. 1997), 66-72.
5. Chen, J., Geyer, W., Dugan, C., Muller, M., and Guy, I. 2009. Make new friends, but keep the old: recommending people on social networking sites. In *Proc of CHI '09*.
6. Das, A.S., Datar, M., Garg, A., and Rajaram, S. 2007. Google news personalization: scalable online collaborative filtering. In *Proc of WWW '07*.
7. Geyer, W., Dugan, C., Millen, D., Muller, M., Freyne, J. 2008. Recommending Topics for Self-Descriptions in Online User Profiles. In *Proc of ACM RecSys'08*.
8. Hill, W. and Terveen, L. 1996. Using frequency-of-mention in public conversations for social filtering. In *Proc of CSCW '96.*
9. Java, A., Song, X., Finin, T., and Tseng, B. 2007. Why we twitter: understanding microblogging usage and communities. In *Proc of the 9th WebKDD and 1st SNA-KDD 2007 Workshop on Web Mining and Social Network Analysis*, 56-65.
10. Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Gordon, L. R., and Riedl, J. 1997. GroupLens: applying collaborative filtering to Usenet news. *Commun. ACM* 40, 3 (Mar. 1997), 77-87.
11. McNee, S. M., Riedl, J., and Konstan, J. A. 2006. Being accurate is not enough: how accuracy metrics have hurt recommender systems. In *CHI '06 Extended Abstracts*.
12. Melville, P., Mooney, R. J. and Nagarajan, R. 2001. Content-boosted collaborative filtering. In *Proc of 2001 SIGIR Workshop on Recommender Systems*.
13. Mooney, R.J., and Roy, L. 2000. Content-based book recommending using learning for text categorization. In *Proc of ACM DL'00.195–204*.
14. Pazzani, M.J., Muramatsu, J. and Billsus, D. 1996. Syskill & webert: Identifying interesting web sites. *AAAI/IAAI*, Vol. 1, 54-61.
15. Rashid, A.M., Lam, S.K., Karypis, G., Riedl, J. 2006. ClustKNN: A highly scalable hybrid model- & memory-based CF algorithm. In *Proc of WebKDD'06: KDD Workshop on Web Mining and Web Usage Analysis.*
16. Sahami, M. and Heilman, T.D. 2006. A web-based kernel function for measuring the similarity of short text snippets. In *Proc of WWW '06*.
17. Salton, G & Buckley, C. 1988. Term-weighting approaches in automatic text retrieval. *Information Processing & Management* 24 (5): 513-523.
18. Sarwar, B.M., Karypis, G., Konstan, J.A., Riedl, J. 2002. Recommender systems for large-scale E-Commerce: Scalable neighborhood formation using clustering. In *Proc of ICCIT 2002*.
19. Schein, A.I., Popescul, A., Ungar, L.H., and Pennock, D.M. 2002. Methods and metrics for cold-start recommendations. In *Proc of SIGIR '02*.
20. Simon, H.A. 1971. Designing Organizations for an Information-Rich World. *Computers, Communications, and the Public Interest* (1971), pp. 37-72.