From Chatter to Headlines: Harnessing the Real-Time Web for Personalized News Recommendation

Gianmarco De Francisci Morales
IMT & ISTI-CNR
Lucca & Pisa, Italy
gdfm@yahoo-inc.com

Aristides Gionis
Yahoo! Research
Barcelona, Spain
gionis@yahoo-inc.com

Claudio Lucchese
ISTI-CNR
Pisa, Italy
claudio.lucchese@isti.cnr.it

ABSTRACT

We propose a new methodology for recommending interesting news to users by exploiting the information in their twitter persona. We model relevance between users and news articles using a mix of signals drawn from the news stream and from twitter: the profile of the social neighborhood of the users, the content of their own tweet stream, and topic popularity in the news and in the whole twitter-land.

We validate our approach on a real-world dataset of approximately 40k articles coming from Yahoo! News and one month of crawled twitter data. We train our model using a learning-to-rank approach and support-vector machines. The train and test set are drawn from Yahoo! toolbar log data. We heuristically identify 3 214 users of twitter in the log and use their clicks on news articles to train our system.

Our methodology is able to predict with good accuracy the news articles clicked by the users and rank them higher than other news articles. The results show that the combination of various signals from real-time web and micro-blogging platforms can be a useful resource to understand user behavior.

Categories and Subject Descriptors

H.4.m [**Information Systems**]: Miscellaneous

General Terms

Algorithms, Experimentation

Keywords

recommendation systems, news recommendation, real-time web, micro-blogging applications, personalization

1. INTRODUCTION

Information overload is a term referring to the difficulty in taking decisions or understanding an issue when there is more information than one can handle. Information overload is not a new problem. The term itself precedes the Internet by several years [23]. However, digital and automated information processing has aggravated the problem to unprecedented levels, transforming it into one of the crucial issues in the modern information society.

In this work we focus on one of the most common daily activities: news reading. Every day the press produces thousands of news articles covering a wide spectrum of topics. For a user, finding relevant and interesting information in this ocean of news is a daunting task.

News portals like Yahoo! news and Google news often resort to recommender systems to help the user find relevant pieces of information. In recent years the most successful recommendation paradigm has been *collaborative filtering* [2]. Collaborative filtering requires the users to rate the items they like. The rating can be explicit (e.g., 'like', '+1', number of stars) or implicit (e.g., click on a link, download). In both cases, collaborative filtering leverages a closed feedback loop: user preferences are inferred by looking only at the user interaction with the system itself. This approach suffers from a data sparsity problem, namely it is hard to make recommendations for users for whom there is little available information or for brand new items, since little or no feedback is available for such "cold" items.

At the same time, at an increasingly rate, web users access news articles via micro-blogging services and the so-called real-time web. By subscribing to feeds of other users, such as friends, colleagues, domain experts and enthusiasts, as well as to organizations of interest, they obtain timely access to relevant and personalized information. Yet, information obtained by micro-blogging services is not complete as highly-relevant events could easily be missed by users if none of their contacts posts about these events.

We propose to recommend news articles to users by combining two sources of information, news streams and micro-blogs, and leveraging the best features of each. News streams have high *coverage* as they aggregate a very large volume of news articles obtained from many different news agencies. On the other hand, information obtained by micro-blogging services can be exploited to address the problems of *information filtering and personalization*, as users can be placed in the context of their social circles and personal interests.

Our approach has the following advantages. First, we are able to overcome the data-sparsity problem: if we do not have enough information for a user, there should be significantly more information from the social circle of the user, which is presumably relevant. Second, more than once it has been reported that news break out earlier on the real-time

WSDM'12, February 8–12, 2012, Seattle, Washington, USA.

web than in traditional media, and we would like to harness this property to provide timely recommendations.

With more than 200 million users, twitter is currently the most popular real-time web service. Twitter is an emerging agora where users publish short text messages, also known as tweets, and organize themselves into social networks. Interestingly, in many cases, news have been published and commented on twitter before any other news agency, as in the case of Osama Bin-Laden's death in 2011,[1] or Tiger Wood's car crash in 2009.[2] Due to its popularity, traditional news providers, such as magazines, news agencies, have become twitter users: they exploit twitter and its social network to disseminate their contents.

Example. In Figure 1(a) we show the normalized number of tweets and news articles regarding "Osama Bin Laden" during the time period between May 1st and 4th, 2011. For the news, we also report the number of clicks in the same period. We notice that the number of relevant tweets ramps up earlier than news, meaning that the information spread through twitter even before any press release. Later on, while the number of tweets decreases quickly, and stabilizes around a relatively small number of tweets, the number of published news continues to be quite large. The number of clicks on news articles follows somewhat the number of published news, even though there is a significant delay until the time the users actually click and read an article on the topic. Figure 1(b) presents similar information for the "Joplin tornado." In this case, even though there is a great activity both on twitter and on news streams, users start reading the news only after one day. About 60% of the clicks on the news occur after 10 hours from its publication. □

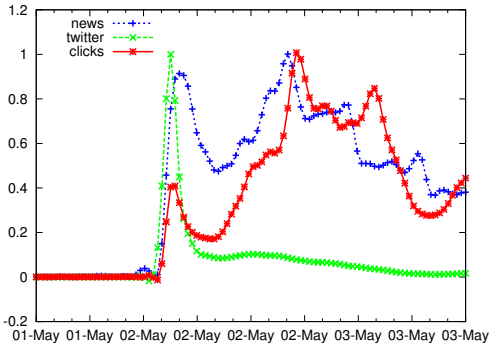
The goal of this work is to reduce the delay between the publication of a news and its access by a user. We aim at helping the users in finding relevant news as soon as they are published. Our envisioned application scenario is a feature operating on a news aggregator like Yahoo! news or Google news. Our objective is to develop a recommendation system that provides the users with fresh, relevant news by leveraging their tweet stream. Ideally, the users log in into the news portal and link their twitter account to their portal account in order to provide access to their tweet stream. The portal analyzes the tweet stream of the users and provides them personalized recommendations.

Solving this problem poses a number of research challenges. First, the volume of tweets and news is significantly large. It is necessary to design a scalable recommender system able to handle millions of users, tweets and news. Second, both tweets and news are unbounded streams of items arriving in real-time. The set of news from which to choose recommendations is highly dynamic. In fact, news are published continuously and, more importantly, they are related to events that cannot be predicted in advance. Also, by nature, news have a short life cycle, since they are replaced by updated news on the same topic, or because they become obsolete. Therefore, a recommender system should be able to find relevant news early, before they lose their value over time. Third, the nature of tweets, short and jargon-like, complicates the task of modeling the interests of users.

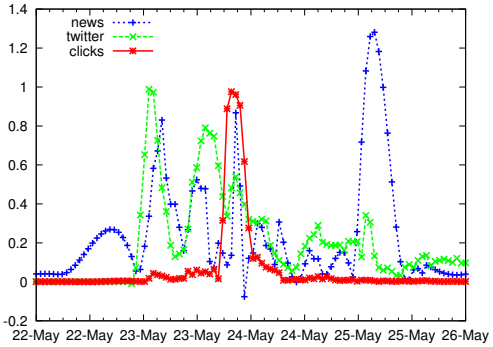
Finally, personalization should leverage user profiles to drive the user to less popular news. But it should not prevent from suggesting news of general interest, even when unrelated to the user profile, e.g., the Fukushima accident.

[1] www.bbc.co.uk/news/technology-13257940

[2] techcrunch.com/2009/11/27/internet-twitter-tiger-woods



(a) *Osama Bin Laden*



(b) *Joplin tornado*

Figure 1: Trends on twitter and news streams.

In summary, our contributions are the following.

- We propose a light-weight, adaptive, online recommendation system. In contrast, typical recommendation systems usually operate in offline mode.
- We present a new application of usage of tweets. Twitter has mainly been used to identify trending topics and analyzing information spread. Its application to recommendation has received little interest in the community. Our main recommendation system is called `T.Rex`, for *twitter-based news recommendation system*.
- Our system provides personalized recommendations by leveraging information from the tweet stream of users, as well as from the streams of their social circle. By incorporating social information we address the issue of data sparsity. When we do not have enough information for a user, for example for a new user or if a user rarely tweets, the available information in the social circle of the user provides a proxy to his interests.

The rest of the paper is organized as follows. In Section 2 we formalize the news recommendation problem, and introduce our proposed personalized news ranking function. In Section 3 we describe the learning process of the ranking function based on click data. In Section 4 we present the experimental setting used in the evaluation of our algorithms and our results. Finally, in Section 5 we discuss other work related to our paper, and Section 6 is a short conclusion.

Table 1: Table of symbols.

Symbol	Definition
$\mathcal{N} = \{n_0, n_1, \ldots\}$	Stream of news
$\mathcal{T} = \{t_0, t_1, \ldots\}$	Stream of tweets
$\mathcal{U} = \{u_0, u_1, \ldots\}$	Set of users
$\mathcal{Z} = \{z_0, z_1, \ldots\}$	Set of entities
$\tau(n_i)$	Timestamp of the i-th news article n_i
$\tau(t_i)$	Timestamp of the i-th tweet t_i
$c(n_i)$	Timestamp of the click on n_i
S	Social network matrix
S^*	Social-influence network matrix
A	User × tweet authorship matrix
T	Tweet × entity relatedness matrix
N	Entity × news relatedness matrix
$M = T \cdot N$	Tweet × news relatedness matrix
$\Gamma = A \cdot M$	User × news content relatedness
$\Sigma = S^* \cdot A \cdot M$	User × news social relatedness
Z	Entity popularity
$\Pi = Z \cdot N$	News popularity
$R_\tau(u, n)$	Ranking score of news n for user u at time τ

2. MODEL

Our goal is to harness the information present in tweets posted by users and by their social circles in order to make relevant and timely recommendation of news articles. We proceed by introducing our notation and define formally the problem that we consider in this paper. For quick reference, our notation is also summarized in Table 1.

Definition 1 (News stream) *Let $\mathcal{N} = \{n_0, n_1, \ldots\}$ be an unbounded stream of news arriving from a set of news sources, where news article n_i is published at time $\tau(n_i)$.*

Definition 2 (Tweet stream) *Let $\mathcal{T} = \{t_0, t_1, \ldots\}$ be an unbounded stream of tweets arriving from the set of twitter users, where tweet t_i is published at time $\tau(t_i)$.*

It is worth noting that a tweet stream for a user is composed of tweets authored by the user and people in the social neighborhood of the user. This is an extension of the concept of "home timeline" as known in twitter.

Problem 1 (News recommendation problem) *Given a stream of news $\mathcal{N}$, a set of users $\mathcal{U} = \{u_0, u_1, \ldots\}$ and their stream of tweets $\mathcal{T}$, find the top-k most relevant news for user $u \in \mathcal{U}$ at time τ.*

We aim at exploiting the tweet and news streams to identify news of general interest, but also at exploiting a twitter-based user profile to provide personalized recommendations. That is, for any user $u \in \mathcal{U}$ at any given time τ, the problem requires to rank the stream of past news in $\mathcal{N}$ according to a user-dependent relevance criteria. We also aim at incorporating time recency into our model, so that our recommendations favor the most recently published news articles.

We now proceed to model the factors that affect the relevance of news for a given user. We first model the social-network aspect. In our case, the social component is induced by the twitter *following* relationship. We define S to be the social network adjacency matrix, were $S(i, j)$ is equal to 1 divided by the number of users followed by user u_i if u_i follows u_j, and 0 otherwise. We also adopt a functional ranking [5] that spreads the interests of a user among its neighbors recursively. By limiting the maximum hop distance d, we define the social influence in a network as follows.

Definition 3 (Social influence S^*) *Given a set of users $\mathcal{U} = \{u_0, u_1, \ldots\}$, organized in a social network where each user may express an interest to the content published by another user, we define the social influence model S^* as the $|\mathcal{U}| \times |\mathcal{U}|$ matrix where $S^*(i, j)$ measures the interest of user u_i to the content generated by user u_j and it is computed as*

$$S^* = \left(\sum_{i=1}^{i=d} \sigma^i S^i \right),$$

where S is the row-normalized adjacency matrix of the social network, d is the maximum hop-distance up to which users may influence their neighbors, and σ is a damping factor.

Next we model the profile of a user based on the content that the user has generated. We first define a binary authorship matrix A to capture the relationship between users and the tweets they produce.

Definition 4 (Tweet authorship A) *Let A be a $|\mathcal{U}| \times |\mathcal{T}|$ matrix where $A(i, j)$ is 1 if u_i is the author of t_j, and 0 otherwise.*

The matrix A can be extended to deal with different types of relationships between users and posts, e.g., weigh differently *re-tweets*, or *likes*. In this work, we limit the concept of authorship to the posts actually written by the user.

We observe that news and tweets often happen to deal with the same topic. Sometimes a given topic is pushed into twitter by a news source, and then it spread throughout the twitter social network. However sometimes a given topic is first discussed by twitter users, and it is later reflected in the news, which may or may not be published back on twitter. In both cases, it is important to discover which are the current trending entities in order to promptly recommend news of interest. We model the relationship between tweets and news by introducing an intermediate layer between the two streams. This layer is populated by what we call *entities*.

Definition 5 (Tweets-to-news model M) *Let $\mathcal{N}$ be a stream of news, $\mathcal{T}$ a stream of tweets, and $\mathcal{Z} = \{z_0, z_1, \ldots\}$ a set of entities. We model the relationship between tweets and news as a $|\mathcal{T}| \times |\mathcal{N}|$ matrix M, where $M(i, j)$ is the relatedness of tweet t_i to news n_j, and it is computed as*

$$M = T \cdot N,$$

where

- *T is a $|\mathcal{T}| \times |\mathcal{Z}|$ row-wise normalized matrix with $T(i, j)$ representing the relatedness of tweet t_i to entity z_j;*
- *N is a $|\mathcal{Z}| \times |\mathcal{N}|$ column-wise normalized matrix with $N(i, j)$ representing the relatedness of entity z_i to news n_j.*

The set of entities $\mathcal{Z}$ introduces a middle layer between the stream of news and the stream of tweets that allows us to generalize our analysis. First, this layer allows to overcome

any vocabulary mismatch problem between the two streams, since the streams are mapped onto the entity space. Second, rather than monitoring the relevance of a specific news or a tweet, we propose to measure the relevance of an entity.

There are a number of techniques that can be used to extract entities from news and tweets. A naïve approach is to let each term in the dictionary play the role of an entity. In this case $T(t, z)$ can be estimated as the number of occurrences of the term z in the tweet t, or as a `tf·idf` score, and similarly for N. An alternative approach is to use probabilistic latent semantic indexing and map tweets and news onto a set of latent topics [13].

In this work we follow a third approach: we use an existing entity-extraction system. In particular we use the `Spectrum` system, which was proposed by Paranjpe [20]. Given any fragment of text, the `Spectrum` system identifies entities related to Wikipedia articles. Therefore, we assume that $\mathcal{Z}$ consists of the set of all titles of Wikipedia articles. This choice has some interesting advantages. First, once an entity is detected, it is easy to propose a meaningful *label* to the user. Second, it allows to include additional external knowledge into the ranking, such as geographic position, categorization, number of recent edits by Wikipedia users, and many others. Although we choose to use `Spectrum`, we note that our model is independent of the specific entity extraction technique employed.

Example. Consider the following tweet by user KimAKelly: "Miss Liberty is closed until further notice." The words "Miss Liberty" are mapped to the entity/Wikipedia page *Statue of Liberty*, which is an interesting topic, due to the just announced renovation. This allows to rank high news regarding the Statue of Liberty, e.g. "Statue of Liberty to Close for One Year after 125th Anniversary" by Fox news. Potentially, since the Wikipedia page is geo-referenced, it is possible to boost the ranking of the news for users living nearby, or being interested in the topic/entity *New York*. □

The three matrices S^*, A, M can be used to measure the relatedness of the news in $\mathcal{N}$ to the tweets posted by a user or her social neighborhood.

Definition 6 (Content-based relatedness Γ) *Given the tweet authorship matrix A and the tweets-to-news model M for a stream of news $\mathcal{N}$ and a stream of tweets $\mathcal{T}$ authored by users $\mathcal{U}$, the relatedness between $\mathcal{U}$ and $\mathcal{N}$ is defined as*

$$\Gamma = A \cdot M,$$

where Γ is a $|\mathcal{U}|\times|\mathcal{N}|$ matrix, where $\Gamma(u_i, n_j)$ is the relevance of news n_j for user u_i.

According to the definition of Γ, a news article n_j is relevant for user u_i, if the news discusses entities that have been discussed in the past by the user in her own tweets.

Definition 7 (Social-based relatedness Σ) *Given the tweet authorship matrix A, the tweets-to-news model M, and the social network S for a stream of news $\mathcal{N}$ and a stream of tweets $\mathcal{T}$ authored by users $\mathcal{U}$, the relatedness between $\mathcal{U}$ and $\mathcal{N}$ is defined as*

$$\Sigma = S^* \cdot A \cdot M,$$

where Σ is a $|\mathcal{U}| \times |\mathcal{N}|$ matrix, where $\Sigma(u_i, n_j)$ is the relevance of news n_j for user u_i.

In the case of social-based relatedness Σ, the relevance of a news is computed by taking into account the tweets authored by neighboring users.

The matrices Γ and Σ measure content-based similarity, with no reference to popularity or freshness of tweets, news articles, and entities. In order to provide timely recommendations and catch up with trending news, we introduce a popularity component, which combines the "hotness" of entities in the news stream and the tweet stream.

Definition 8 (Entity-based news popularity Π) *Given a stream of news $\mathcal{N}$, a set of entities $\mathcal{Z}$, and their relatedness matrix N, the popularity of $\mathcal{N}$ is defined as*

$$\Pi = Z \cdot N,$$

where Z is a row-wise vector of length $|\mathcal{Z}|$ and $Z(i)$ is a measure of the popularity of entity z_i. The resulting Π is a row-wise vector of length $|\mathcal{N}|$, where $\Pi(j)$ measures the popularity of the news article n_j.

The vector Z holds the popularity of each entity z_i. The counts of the popularity vector Z need to be updated as new entities of interest arise in the news stream $\mathcal{N}$ and the tweet stream $\mathcal{T}$. An important aspect in updating the popularity counts is to take into account recency: new entities of interest should dominate the popularity counts of older entities. In this paper, we choose to update the popularity counts using an exponential decay rule. We discuss the details in Section 2.1. However, note that the popularity update is independent of our recommendation model, and any other decaying function can be used.

Finally, we propose a ranking function for recommending news articles to users. The ranking function is linear combination of the scoring components described above.

Definition 9 (Recommendation ranking $R_\tau(u, n)$) *Given the components Σ_τ, Γ_τ and Π_τ, resulting form a stream of news $\mathcal{N}$ and a stream of tweets $\mathcal{T}$ authored by users $\mathcal{U}$ up to time τ, the recommendation score of a news article $n \in \mathcal{N}$ for a user $u \in \mathcal{U}$ at time τ is defined as*

$$R_\tau(u, n) \;=\; \alpha \cdot \Sigma_\tau(u, n) \;+ \beta \cdot \Gamma_\tau(u, n) \;+ \gamma \cdot \Pi_\tau(n),$$

where α, β, γ are coefficients that specify the relative weight of the components.

At any given time, the recommender system produces a set of news recommendation by ranking a set of candidate news, e.g., the most recent ones, according to the ranking function R. To motivate the proposed ranking function we note similarities with popular recommendation techniques. When $\beta = \gamma = 0$, the ranking function R resembles collaborative filtering, where user similarity is computed on the basis of their social circles. When $\alpha = \gamma = 0$, the function R implements a content-based recommender system, where a user is profiled by the bag-of-entities occurring in the tweets of the user. Finally, when $\alpha = \beta = 0$, the most popular items recommended, regardless of the user profile.

Note that Σ, Γ, Π and R are all time dependent. At any given time τ, the social network and the set of authored tweets vary, thus affecting Σ and Γ. More importantly, some entities may abruptly become popular, and therefore of interest to many user. This time dependency is modeled by Π. While the changes in Σ and Γ derive directly from the

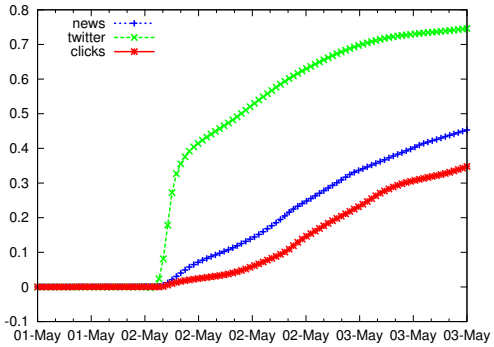


Figure 2: *Osama Bin Laden* trends on twitter and news streams.

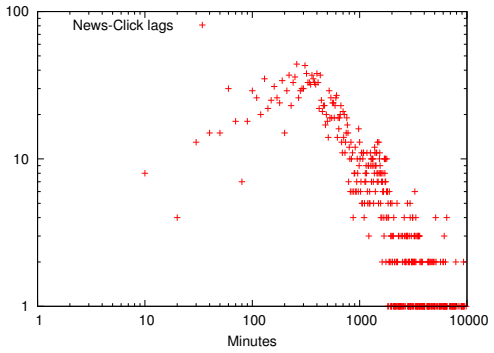


Figure 3: News clicks delay distribution.

tweet stream $\mathcal{T}$ and the social network S, the update of Π is crucial, and plays a fundamental role in the recommendation system that we describe in the next section.

2.1 Entity popularity

We complete the description of our model by discussing the update of the entity popularity counts. We motivate our approach by empirically observing how the user interest for particular entities decays over time.

In Figure 2 we show the cumulative distribution of occurrences and news clicks for the same entity in Figure 1(a) shown in the introduction. By using cumulative distribution, the fact the entity appears much earlier in twitter than in the news becomes more evident. If we consider the number of clicks on news as a surrogate of user interest, we notice that with a delay of about one day the entity receives a great deal of attention. This delay is probably due to the fact that the users have not been informed about the event yet. On the other hand, we observe that after two days the number of clicks drop, possibly the interest of users has been saturated or diverted to other events.

It turns out that the example above is not an exception, but it describes well the typical behavior of users with respect to clicking news articles. Figure 3 shows the distribution of the delay between the time that news articles are published the when they are clicked by users. The figure considers all the news articles and all the clicks in our dataset (see Section 4.1). We see that a very small number of news is clicked within the first hour from their publication. On the other hand, 76.7% of the clicks happen within 1 day (1,440 minutes) and 90.1% within 2 days (2,880 minutes).

From these empirical observations, we draw the following conclusions.

1. The increase in the number of tweets and news related to a given entity can be used to predict the increase of interest of the users.

2. News become stale after two days.

The first observation motivates us to inject a popularity component in our recommendation model. The second observation suggests updating popularity counts using a decaying function. We choose an exponentially-decaying function, which has the advantage of allowing to count efficiently frequent items in the data stream model, in order to monitor entity popularity in high-speed data streams [9].

Definition 10 (Popularity-update rule) *Given a stream of news $\mathcal{N}$ and a stream of tweets $\mathcal{T}$ at time τ, the popularity vector Z is computed at the end of every time window of fixed width ω as follows*

$$Z_\tau = \lambda Z_{\tau-1} + w_\mathcal{T} H_\mathcal{T} + w_\mathcal{N} H_\mathcal{N}.$$

The vectors $H_\mathcal{T}$ and $H_\mathcal{N}$ are estimates of the expected number of mentions of the entity occurring in tweets and news, respectively, during the latest time window. They are also called hotness coefficients. *The weights $w_\mathcal{T}$ and $w_\mathcal{N}$ measure the relative impact of news and tweets to the popularity count, and $\lambda < 1$ is an exponential forgetting factor.*

The popularity-update rule has the effect of promptly detecting entities becoming suddenly popular, and spreading this popularity in the subsequent time windows. According to our experimental evidence, we fix λ so that a signal is becomes negligible after two days. We set both coefficients $w_\mathcal{T}$ and $w_\mathcal{N}$ to 0.5 to give equal weight to tweets and news.

As shown by Asur et al. [4], the number of tweets for a trending topic grows linearly over time. Thus, we compute the expected number of mentions of the entity in tweets and news $H_\mathcal{T}$ and $H_\mathcal{N}$ by using the first order derivative of their cumulative counts, measured at the last time window $\tau - 1$.

3. LEARNING ALGORITHM

The next step is to estimate the parameters of the relevance model that we developed in the previous section. The parameters consist of the coefficients α, β, γ used to adjust the relative weight of the components of the ranking function R_τ. Another parameter is the damping factor used in the computation of social influence, but we empirically observed that it does not influence the results very much, so we used a default value of $\sigma = 0.85$. We consider only direct neighbors by setting the maximum hop distance $d = 1$.

Our approach is to learn the parameters of the model by using training data obtained from action logs. In particular, we use click-log data from Yahoo! toolbar as a source of user feedback. The Yahoo! toolbar anonymously collects clicks of several millions of users on the Web, including clicks on news. Our working hypothesis is that a high-quality news recommender ranks high the news that will be clicked by users. We actually formulate our recommendation task according to the learning-to-rank framework [16].

Consider a user $u \in \mathcal{U}$ and a news article $n \in \mathcal{N}$ with publication timestamp $\tau(n) \leq \tau$, where τ is the current time. We say that the news article n should be ranked in the i-th position of the recommendation list for user u, if it will be the i-th article to be clicked in the future by user u.

This formulation allows to define precedence constraints on the ranking function R_τ. Let $c(n)$ be the time at which the news n is clicked, and let $c(n) = +\infty$ if n is never clicked. At time τ, for two news n_i, n_j with $\tau(n_i) \leq \tau$ and $\tau(n_j) \leq \tau$, we obtain the following constraint:

$$\text{If } \tau \leq c(n_i) < c(n_j) \text{ then } R_\tau(u, n_i) > R_\tau(u, n_j). \quad (1)$$

The click stream from Yahoo! toolbar identifies a large number of constraints according to Equation (1) that the optimal ranking function must satisfy. As for the learning-to-rank problem [16], finding the optimal ranking function is an **NP**-hard problem. Additionally, considering that some of the constraints could be contradictory, a feasible solution may not exist. As usual, the learning problem is translated to a Ranking-SVM optimization problem.

Problem 2 (Recommendation Ranking Optimization)

$$\begin{aligned}
&\textit{Minimize: } V\left(\overrightarrow{\omega} \equiv \langle \alpha, \beta, \gamma \rangle, \overrightarrow{\xi}\right) = \frac{1}{2}\|\overrightarrow{\omega}\|^2 + C\sum \xi_{ij\tau} \\
&\textit{Subject to: } R_\tau(u, n_i) > R_\tau(u, n_j) + 1 - \xi_{ij\tau} \\
&\qquad \textit{for all } \tau, n_i \in \mathcal{N}, n_j \in \mathcal{N} \textit{ such that} \\
&\qquad\quad \tau(n_i) \leq \tau, \tau(n_j) \leq \tau, \tau \leq c(n_i) < c(n_j) \\
&\qquad \xi_{ij\tau} \geq 0
\end{aligned}$$

As shown by Joachims [16], this optimization problem can be solved via classification SVM. In the following sections, we show how to generate the training set of the SVM classifier in order to keep a reasonably low amount of training instances, so as to speed-up convergence and support the scalability of the solution.

3.1 Constraint selection

The formulation of Problem 2 includes a potentially huge number of constraints. Every click occurring after time τ on a news n, generates a new constraints involving n and every other non-clicked news published before time τ. Such a large number of constraints also includes relationships on "stale" news articles, e.g., a non-clicked news article published weeks or months before τ. Since clicks are the informative signals driving the learning process, it is important to select pairs of clicked news articles that eliminate biases, such as the one caused by stale news articles. Clearly, the more constraints are taken into consideration during the learning process, the more robust is the final model. On the other hand, increasing the number of constraints affects the complexity of the minimization algorithm. We propose the following strategy in order to select only the most interesting constraints and thus simplify the optimization problem.

First, we evaluate the ranking function only at the time instants when a click happens. This selection does not actually change the set of constraints of Problem 2, but it helps in the generation of the constraints by focussing on specific time instants. If the user u clicks at the news article n_i at time $c(n_i)$, then the news article n_i must be the most relevant at that time. The following must hold.

$$\begin{aligned}
R_{c(n_i)}(u, n_i) > R_{c(n_i)}(u, n_j) + 1 - \xi_{ijc(n_i)} \\
\text{for all } n_j \in \mathcal{N} \text{ such that } \tau(n_j) \leq c(n_i). \quad (2)
\end{aligned}$$

Whenever a click occurs at time $c(n_i)$, we add a set of constraint to the ranking function such that the clicked news article gets the largest score among any other news article published before time $c(\tau)$.

Second, we restrict the number of news articles to be compared with the clicked one. If a news article was published a long time before the click, then it can be easily filtered out, as it would not help the learning process. As we have shown in Section 2.1, users lose interest into news articles after a time interval $\widehat{\tau}$ of two days. We can make use of this threshold into Equation (2) as follows:

$$\begin{aligned}
R_{c(n_i)}(u, n_i) > R_{c(n_i)}(u, n_j) + 1 - \xi_{ijc(n_i)} \\
\text{for all } n_j \in \mathcal{N}\left[c(n_i) - \widehat{\tau}, c(n_i)\right] \quad (3)
\end{aligned}$$

where $\mathcal{N}\left[c(n_i) - \widehat{\tau}, c(n_i)\right]$ is the set of news published between time $c(n_i) - \widehat{\tau}$ and $c(n_i)$.

Finally, we further restrict $\mathcal{N}\left[c(n_i) - \widehat{\tau}, c(n_i)\right]$ by considering only those news articles that are relevant according to at least one of the three score components Σ_τ, Γ_τ, Π_τ. Let $\mathsf{Top}(k, \chi, \tau_a, \tau_b)$ be the set of k news articles with largest rank in the set $\mathcal{N}\left[\tau_a, \tau_b\right]$ according to the score component χ. We include into the optimization Problem 2 only the constraints:

$$\begin{aligned}
&R_{c(n_i)}(u, n_i) > R_{c(n_i)}(u, n_j) + 1 - \xi_{ijc(n_i)} \\
&\text{for all } n_j \text{ s.t. } n_j \in \mathsf{Top}\left(k, \Sigma_{c(n_i)}, c(n_i) - \widehat{\tau}, c(n_i)\right) \text{ or} \\
&\qquad\qquad n_j \in \mathsf{Top}\left(k, \Gamma_{c(n_i)}, c(n_i) - \widehat{\tau}, c(n_i)\right) \text{ or} \\
&\qquad\qquad n_j \in \mathsf{Top}\left(k, \Pi_{c(n_i)}, c(n_i) - \widehat{\tau}, c(n_i).\right) \quad (4)
\end{aligned}$$

By setting $k = 10$ we are able to reduce the number of constraints from more than 25 million to approximately 250 thousand, significantly reducing the training time.

3.2 Additional features

The system obtained by learning the model parameters using the SVM-Rank method is named `T.Rex`, and forms our main recommender. Additionally, we attempt to improve the accuracy of `T.Rex` by incorporating more features. To build this improved recommender we use exactly the same learning framework: we collect more features from the same training dataset and we learn their importance using the SVM-Rank algorithm. We choose three additional features: *age*, *hotness* and *click count*, which we describe below.

The *age* of a news article n is the time elapsed between the current time and the time n was published, that is, $\tau - \tau(n)$.

The *hotness* of a news article is a set of features extracted from the vectors $H_\mathcal{T}$ and $H_\mathcal{N}$, which keep the popularity counts for the entities in our model. For each news article n we compute the average and standard deviation of the vectors $H_\mathcal{T}$ and $H_\mathcal{N}$ over all entities extracted from article n. This process gives us four hotness features per news article.

Finally, the *click count* of a news article is simply the number of times that the article has been clicked by any users in the system up to time τ.

The system that we obtain by training our ranking function on these additional features is called `T.Rex`+.

4. EXPERIMENTS

4.1 Datasets

To build our recommendation system we need the following sources of information: twitter stream, news stream, the social network of users, and click-through data. We extract this information from three different data sources: twitter, Yahoo! news, and Yahoo! toolbar, respectively.

Twitter: We obtain tweets from twitter's public API by crawling users' timelines. We collect all tweets posted during May 2011 by our 3,214 target users (identified as described below). We also collect a random sample of tweets to track entity popularity across all twitter. We extract entities from tweets using the `Spectrum` method described by Paranjpe [20]. Overall we obtain about 1 million tweets in English, for which we are able to extract entities. Figure 4(a) shows the distribution of entities in the twitter dataset. The curve has a truncated power law shape.

Yahoo! news: We collect all news articles aggregated in the English site of Yahoo! news during May 2011. From the news articles we extract entities by using again the `Spectrum` algorithm. Overall we have about 28.5 million news articles, from which we keep only the articles that contain at least one entity contained in one of the tweets. In total we obtain about 40 thousand news articles. Figure 4(b) shows the distribution of entities in the Yahoo! news dataset. The shape of the distribution is similar to the one of twitter.

Yahoo! toolbar: We collect click information from the Yahoo! toolbar logs. The Yahoo! toolbar is a browser add-on that provides a set of browsing functionalities. The logs contain information about the browsing behavior of the users who have installed the toolbar, such as, user cookie id, url, referral url, event type, and so on. We collect all toolbar data occurring in May 2011. We use the first 80% in chronological order of the toolbar clicks to train our system.

Using a simple heuristic we identify a small set of users for whom we can link their toolbar cookie id, with their twitter user id. The heuristic is to identify which twitter account a user is visiting more often, discarding celebrity accounts and non-bijective mappings. The underlying assumption is that users visit more often their own accounts. In total we identify of set U_0 of 3,214 test users. The dataset used in this paper is the projection of all the data sources on the users of the set U_0, that is, for all the users in U_0 we collect all their tweets and all their clicks on Yahoo! news. Additionally, by employing snowball sampling on twitter we obtain the set of users U_1 who are followed by the set of users U_0. Then, by collecting all the tweets of the users in U_1, we form the social component for our set of users of interest U_0.

4.2 Test set

Evaluating a recommendation strategy is a complex task. The ideal evaluation method is to deploy a live system and gather click-through statistics. While such a deployment gives the best accuracy, it is also very expensive. An alternative evaluation method is via user studies. However, getting judgements from human experts does not scale well to a large number of recommendations. Furthermore, user studies cannot be automated, and they are impractical if more than one strategy or many parameters need to be tested.

For these reasons, we propose an automated method to evaluate our recommendation algorithm. The proposed evaluation method exploits the available click data collected by the Yahoo! toolbar, and it is similar in spirit with the learning process. Given a stream of news and a stream of tweets for the user, we identify an event that a user clicks at a news article n_*. Assume that such a click occurs at time $\tau = c(n_*)$. Suppose the user just logged in the system at time τ. Then the recommendation strategy should rank the news article n_* as high as possible.

To create a test instance, we collect all news articles that have been published within the time interval $[\tau - \tau^*, \tau]$, and select a subset as described by Equation (4): we pick the news with largest scores according to the components of content, social, popularity, and number of clicks until time τ. In total we generate a pool of k = 1,000 candidate news. All these news articles are ranked using our ranking function R_τ, and we then examine what is the ranking of the article n_* in the list.

The last 20%, in chronological order, of the Yahoo! toolbar data is used to test the `T.Rex` and `T.Rex+` algorithms, as well as all the baselines.

4.3 Evaluation measures

Evaluating a ranking of items is a standard problem in information retrieval. In our setting we have only one correct answer per ranking — the news article n_* clicked by the user — and we care about the position of this item in the ranking, which we evaluate using the mean reciprocal rank (MRR) measure [24]:

$$\text{MRR} = \frac{1}{|Q|} \sum_{i=1}^{Q} \frac{1}{r(n_*^i)},$$

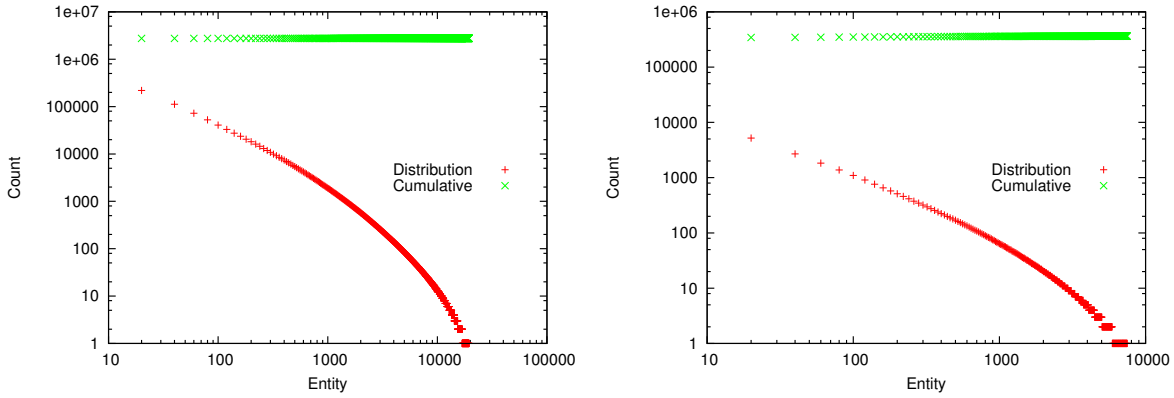
where $r(n_*^i)$ is the rank of the news article n_*^i in the i-th event of the evaluation and Q is the set of all tests. The MRR measure is commonly used to evaluate question-answering (QA) systems, where only the first correct answer matters. MRR gives higher scores to correct results that are higher in the ranking and reduces the importance of correct predictions in the lower parts of the ranking.

We also compute the precision of our system at different levels. For our task we define the *precision-at-k* (P@k) as the fraction of rankings in which the clicked news is ranked in the top-k positions. We compute P@k for k = 1, 5, and 10.

There are cases in which the sparsity of the data prevents our system from providing recommendations. Therefore, we also evaluate the coverage of our recommendations. We define the coverage as the fraction of clicks for which we were able to provide some recommendation.

Finally, we note that when the user clicks on a news, the user is actually expressing interest in the entities mentioned in the news. We can argue that if a user clicks on a report on Fukushima from CNN, the user might equally be interested in a report on the same topic from BBC. In this case, we would like to predict the entities that the user is interested in, rather than the news articles themselves.

We use Jaccard overlap between the sets of entities mentioned in two news articles to measure the similarity between the articles themselves. We compute the similarity between each news article in the ranking and the clicked news. Then, we assign a relevance level to each news by binning the computed similarity into five levels. We use DCG [14] to measure the performance of the strategies on this multiple level



(a) Twitter.

(b) News.

Figure 4: Distribution of entities

Algorithm	MRR	P@1	P@5	P@10	Coverage
`Recency`	0.020	0.002	0.018	0.036	1.000
`ClickCount`	0.059	0.024	0.086	0.135	1.000
`Social`	0.017	0.002	0.018	0.036	0.606
`Content`	0.107	0.029	0.171	0.286	0.158
`Popularity`	0.008	0.003	0.005	0.012	1.000
`T.Rex`	0.107	0.073	0.130	0.168	1.000
`T.Rex+`	0.109	0.062	0.146	0.189	1.000

Table 2: MRR, precision and coverage.

relevance task:

$$\mathrm{DCG}[i] = \begin{cases} G[i] & \text{if } i = 1; \\ \mathrm{DCG}[i-1] + \frac{G[i]}{log_2 i} & \text{if } i > 1, \end{cases}$$

where $G[i]$ is the relevance of the document at position i in our ranking. We compute the DCG for the top 20 results averaged over all the rankings produced. If a strategy is not able to produce a ranking, we pad the ranking with non-relevant documents.

4.4 Baselines

We compare our system with other five ranking baselines:

`Recency`: it ranks news articles by time of publication (most recent first);

`ClickCount`: it ranks news articles by click count (highest count first);

`Social`: it ranks news articles by using our algorithm with $\beta = \gamma = 0$;

`Content`: it ranks news articles by using our algorithm with $\alpha = \gamma = 0$;

`Popularity`: it ranks news articles by using our algorithm with $\alpha = \beta = 0$.

4.5 Results

We report MRR, precision and coverage results in Table 2. We observe that the two variants of our model, `T.Rex` and `T.Rex+`, have overall the best results.

The `T.Rex+` system has the highest MRR of all the alternatives. This result means that our model has a good overall performance across the dataset. `Content` has also a very high MRR. Unfortunately, the coverage level achieved by the `Content` strategy is very low. This issue is mainly caused by the sparsity of the user profiles. It is well know that most of twitter users belong to the "silent majority," and do not tweet very much.

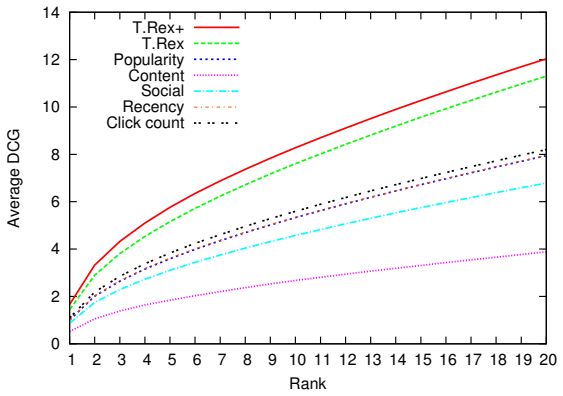


Figure 5: Average discounted cumulated gain on related entities on top 20 ranks.

The `Social` strategy is affected by the same problem, albeit to a much lesser extent. The reason for this difference is that `Social` draws from a large social neighborhood of user profiles, instead of just one. So it has more chances to provide a recommendation. The quality of the recommendation is however quite low, probably because the social-based profile only is not able to catch the specific user interests.

It is worth noting that in almost 20% of the cases, `T.Rex+` was able to rank the clicked news in the top 10 results. Ranking by the `ClickCount` strategy is quite efficient despite being very simple. Interestingly, adding the `ClickCount` feature to our model does not improve the results significantly. In fact, the difference in P@10 between `T.Rex` and `T.Rex+` is only about 2%. This suggests that the information contained in the click counts is already captured by our model, or cannot be effectively exploited by a linear combination. This issue deserves further investigation.

Figure 5 shows the DCG measure for the top 20 positions

in the rankings. From the figure we can see that T.Rex and T.Rex+ are able to suggest more related news than the other strategies. This result is expected, as our system is designed to exploit the concept of entities.

5. RELATED WORK

Recommender systems can be roughly divided in two large categories: content-based and collaborative filtering [12].

The large-scale system for generating personalized news recommendations proposed by Das et al. [10] falls in the second category. The authors exploit a linear combination of three different approaches: minhashing-based clustering of users, probabilistic latent semantic indexing of users and news, and news co-visitation count. All of the three approaches take advantage the user click-through data only, and therefore they are content agnostic. Even tough the system can update its recommendation immediately after a new click is observed, we have seen that click information arrives with a significant delay, and therefore it may fail in detecting emerging topics early.

Thus, for addressing emerging topics the idea of using twitter to provide *fresh* recommendations has recently become very appealing. A number of studies attempts to understand the twitter social network, the information-spreading process, and to discover emerging topics of interesest over such a network [6, 7, 15].

Chen et al. [8] propose a URL-recommender system from URLs posted in twitter. A user study shows that both the user content-based profile, and the user social neighborhood plays a role, but, the most important factor in the recommendation performance is given by the social ranking, i.e., the number of times a URL is mentioned among the neighborhood of a given user. The work presented by Esparza et al. [11] exploits data from a micro-blogging movie review service similar to twitter. The user profile is built on the basis of the user's posts. Similarly, a movie profile generated from the posts associated to the movie. The proposed prototype resembles a content-based recommender system, where users are matched against recommended items.

A small user study by Teevan et al. [22] reports that 49% of users search while looking for information related to news, or to topics gaining popularity, and in general to "to keep up with events." The analysis of a larger crawl of twitter by Kwak et al. [17] shows that about 85% of the twitter posts are about headlines or persistent news. The abundance of news-related content thus makes twitter the ideal source of information for the news recommendation task.

Akcora et al. [3] use twitter to detect abrupt opinion changes. Based on an *emotion-word corpus*, the proposed algorithm detects opinion changes, which can be related to publication of some interesting news. No actual automatic linking to news is produced. The system proposed by Phelan et al. [21] is a content-based approach that uses tweets to rank news using tf·idf. A given a set of tweets, either public or of a friend, is used to build a user profile, which is matched against the news coming from a set of user-defined RSS feeds. Mccreadie et al. [18] show that, even if useful, URL links present in blog posts may not be sufficient to identify interesting news due to their sparsity. Their approach exploits user posts as if they were votes for the news of a given day. The association between news and posts is achieved by exploiting a divergence from randomness model. They show that a Gaussian weighting scheme can profitably be used to predict the importance of a news on a given day, given the posts of a few previous days.

Our approach pursues the same direction, with the goal of exploiting twitter posts to predict news of interest. To this end, rather than analyzing the raw text of a tweet, we chose to extract the topics discussed in tweets. Indeed, twitter highlights in its web interfaces the so called *trending topics*, i.e., set of words occurring frequently in recent tweets. Asur et al. [4] crawled all the tweets containing the keywords identifying a twitter trending topic in the 20 minutes before the topic is detected by twitter. Authors found that the popularity of a topic can be described as a multiplicative growth process with noise. From this, it can be derived that the cumulative number of tweets related to a trending topic increases linearly with time. Another interesting study on a large crawl of twitter has been conducted by Kwak et al. [17], where a number of features and phenomena across the twitter social network have been analyzed. We highlight a few interesting findings. After the initial break-up, the cumulative number of tweets of a trending topic increases linearly, as suggested by Asur et al. [4], and independently of the number of users. Almost 80% of the trending topics have a single activity period, i.e., they occur in a single burst, and 93% of such activity periods last less than 10 days. Finally, once a tweet is published, half of its *re-tweets* occur within an hour and 75% within one day. Also, once re-tweeted, tweets quickly spread four hops away. These studies confirm the fast information spreading occurring on twitter.

In our work, we did not exploit the twitter *trending topics* for two reasons. First, because we need to discover interesting topics even before they are recognized as trending by twitter. Second, our system provides personalized recommendations based on users interests, while *trending topics* capture general interests and are not suitable for modeling a single user.

A basic approach for building topic-based user profiles from tweets is proposed by Michelson and Macskassy [19]. Each capitalized non-stopword is considered an entity. The entity is used as a query to Wikipedia and the categories of the retrieved page are used to update the list of topics of interest for the user who authored the tweet.

Abel et al. [1] propose a more sophisticated user model to support news recommendation for twitter users. They explore different ways of modeling use profiles by using hashtags, topics or entities and conclude that entity based modeling gives the best results. They employ a simple recommender algorithm that uses cosine similarity between user profiles and tweets. The authors recommend tweets containing news URLs and re-tweets are used as ground truth.

Similarly, we used the Spectrum system [20] to map every single tweet to a bag of entities, each corresponding to a Wikipedia page. In addition to the user-authored posts, we used her social network to further enrich the user profile. We apply the same entity-extraction process to the incoming stream of news, with the goal of overcoming the vocabulary mismatch problem by mapping both tweets and news in the same entity-based coordinate system.

Our proposed model borrows from the aforementioned works by exploiting a blend of content-based and social-based profile enrichment. But in addition to the state-of-the-art recommender systems, it is able to discover emerging trends on twitter by measuring their popularity and taking into account aging effects.

6. CONCLUSIONS AND FUTURE WORK

We presented a new method for making personalized recommendations of news articles, which leverages information found in real-time Web. We use data from twitter and Yahoo! news in order to build user profiles that model user interests, content found in the social neighborhood of users, and topic popularity. These signals are combined using a learning-to-rank approach with training data extracted from the Yahoo! toolbar logs. The resulting systems, `T.Rex` and its variant `T.Rex+`, are light-weight online recommenders that are able to predict with good accuracy the news articles clicked by the users and rank them higher.

A number of research questions deserve further investigation. First, we are interested to explore the effects of a different entity space, for instance, performing LSI-type of decompositions on the existing entity space, or using a higher level representation, for example, moving from wikipedia article titles to broad-level categories such as "US News," "movies," and "cooking." We also intend to test if the accuracy of the system can be improved by clustering the users and learning different parameters for each cluster. Note that such clusters do not need to be topical but behavioral. For example, one cluster may represent the users who have broad set of interests and gets influenced very little from their social neighborhood, and so on. Finally, we plan to investigate the effect of using other notions of social neighborhoods, such as geographic locations at a city, state, or country level.

Acknowledgements. Claudio Lucchese and Gianmarco De Francisci Morales were partially supported by the EU-FP7-250527 (Assets) and the POR-FESR 2007-2013 (VISITO Tuscany) projects. Aristides Gionis and Gianmarco De Francisci Morales were partially supported by the Torres Quevedo Program of the Spanish Ministry of Science and Innovation, co-funded by the European Social Fund, and by the Spanish Centre for the Development of Industrial Technology under the CENIT program, project CEN-20101037, "Social Media" (http://www.cenitsocialmedia.es/).

References

[1] F. Abel, Q. Gao, G. Houben, and K. Tao. Analyzing user modeling on twitter for personalized news recommendations. In *19th International Conference on User Modeling, Adaption and Personalization*, 2011.

[2] G. Adomavicius and A. Tuzhilin. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17, 2005.

[3] C. Akcora, M. Bayir, M. Demirbas, and H. Ferhatosmanoglu. Identifying breakpoints in public opinion. In *1st Workshop on Social Media Analytics*, 2010.

[4] S. Asur, B. Huberman, G. Szabo, and C. Wang. Trends in social media: Persistence and decay. *Arxiv preprint arXiv:1102.1402*, 2011.

[5] R. Baeza-Yates, P. Boldi, and C. Castillo. Generalizing pagerank: Damping functions for link-based ranking algorithms. In *29th International Conference on Research and Development in Information Retrieval*, 2006.

[6] E. Bakshy, J. M. Hofman, W. A. Mason, and D. J. Watts. Everyone's an influencer: quantifying influence on twitter. In *4th International Conference on Web Search and Data Mining*, 2011.

[7] M. Cataldi, L. Di Caro, and C. Schifanella. Emerging topic detection on twitter based on temporal and social terms evaluation. In *10th International Workshop on Multimedia Data Mining*, 2010.

[8] J. Chen, R. Nairn, L. Nelson, M. Bernstein, and E. Chi. Short and tweet: experiments on recommending content from information streams. In *28th International Conference on Human Factors in Computing Systems*, 2010.

[9] G. Cormode, F. Korn, and S. Tirthapura. Exponentially decayed aggregates on data streams. In *24th International Conference on Data Engineering*, 2008.

[10] A. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: scalable online collaborative filtering. In *16th International Conference on World Wide Web*, 2007.

[11] S. G. Esparza, M. O'Mahony, and B. Smyth. On the real-time web as a source of recommendation knowledge. *4th ACM Conference on Recommender Systems*, 2010.

[12] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35, 1992.

[13] T. Hofmann. Probabilistic latent semantic indexing. *22nd International Conference on Research and Development in Information Retrieval*, 1999.

[14] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transaction Information Systems*, 20, 2002.

[15] A. Java, X. Song, T. Finin, and B. Tseng. Why we twitter: understanding microblogging usage and communities. In *9th WebKDD and 1st SNA-KDD workshop on Web Mining and Social Network Analysis*, 2007.

[16] T. Joachims. Optimizing search engines using clickthrough data. In *8th International Conference on Knowledge Discovery and Data Mining*, 2002.

[17] H. Kwak, C. Lee, H. Park, and S. Moon. What is twitter, a social network or a news media? *19th International Conference on World Wide Web*, 2010.

[18] R. Mccreadie, C. Macdonald, and I. Ounis. News article ranking: leveraging the wisdom of bloggers. In *9th International Conference on Adaptivity, Personalization and Fusion of Heterogeneous Information*, 2010.

[19] M. Michelson and S. Macskassy. Discovering users' topics of interest on twitter: a first look. In *4th workshop on Analytics for noisy unstructured text data*, 2010.

[20] D. Paranjpe. Learning document aboutness from implicit user feedback and document structure. In *18th ACM Conference on Information and Knowledge Management*, 2009.

[21] O. Phelan, K. McCarthy, M. Bennett, and B. Smyth. Terms of a Feather : Content-Based News Recommendation and Discovery Using Twitter. *Advances in Information Retrieval*, 6611(07), 2011.

[22] J. Teevan, D. Ramage, and M. R. Morris. #twittersearch: a comparison of microblog search and web search. In *4th International Conference on Web Search and Data Mining*, 2011.

[23] A. Toffler. *Future shock*. Random House Publishing Group, 1984.

[24] E. M. Voorhees. The TREC-8 Question Answering Track Report. In *Text REtrieval Conference*, 1999.