Activity Ranking in LinkedIn Feed

Deepak Agarwal, Bee-Chung Chen, Rupesh Gupta,
Joshua Hartman, Qi He, Anand Iyer, Sumanth Kolar, Yiming Ma,
Pannaga Shivaswamy, Ajit Singh and Liang Zhang
LinkedIn Corporation
Mountain View, CA, USA
{dagarwal,bchen,rugupta,jhartmain,qhe}@linkedin.com
{aiyer,skolar,yma,pshivasw,ajsingh,lizhang}@linkedin.com

ABSTRACT

Users on an online social network site generate a large number of heterogeneous activities, ranging from connecting with other users, to sharing content, to updating their profiles. The set of activities within a user's network neighborhood forms a stream of updates for the user's consumption. In this paper, we report our experience with the problem of ranking activities in the LinkedIn homepage feed. In particular, we provide a taxonomy of social network activities, describe a system architecture (with a number of key components open-sourced) that supports fast iteration in model development, demonstrate a number of key factors for effective ranking, and report experimental results from extensive online bucket tests.

Categories and Subject Descriptors

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

General Terms

Design; Experimentation

Keywords

Activity Ranking; Relevance; Large Scale Learning

1. INTRODUCTION

Professional networking site LinkedIn and other social networking sites provide users an easy way to connect with other users and keep track of various updates from them. An important mechanism that helps each user keep track of such updates is through a feed of activities that is centered around her egocentric networks (her connections, people she follows, companies she follows, etc). Some examples of activities generated by a user's first degree network include job changes or profile updates, sharing of content, likes or comments on a piece of information, joining a group or connecting with another user. Since the volume of social network activities is usually large and each user has limited time to consume information in his/her feed, it is important to rank activities according their "relevance" to the user, while keeping the feed fresh and diverse.

KDD'14, August 24–27, 2014, New York, NY, USA.

ACM 978-1-4503-2956-9/14/08 ...$15.00.
http://dx.doi.org/10.1145/2623330.2623362.

The primary goal of the feed is to maximize long-term user engagement, manifested by a user visiting the site more often. However, optimizing the visit frequency of a user is difficult in practice due to long feedback loops. Proxies that are readily measurable with little feedback delay are often optimized in practice. While there are several such proxies, the most popular approach is based on the probability of a user clicking on an activity. Such a click probability is also called click-through rate (CTR).

Challenges: Ranking activities in social network feeds is challenging because of the following reasons.

- *Scalability*: The volume of activities to be ranked is high. The number of user visits is large. The data to be used for training CTR prediction models is also large. A ranking system needs to scale along all these dimensions.
- *Personalization*: There is wide variation in egocentric networks (neighborhoods) of users. Users also have different information needs and feed consumption patterns, and may prefer different types of activities. A good ranking system needs to rank activities according to each user's individual preference and provide support to test many potential signals for personalization through fast iteration of machine-learned models.

This paper describes our experience working with the production system at LinkedIn. We describe the approaches we took to address these challenges and the lessons learned. We would like to point out at the very outset that this paper is not about presentation of fundamentally new machine learning methods, but rather about practical challenges faced when deploying machine learning methods in a production environment and our attempts to address them. When studying new data mining and machine learning methods, testing out-of-sample predictive accuracy of machine-learned models on retrospective data is usually the main focus . Although useful, it does not provide the whole story of the challenges involved in deploying models in production.

Before we discuss our ranking system, we first describe a couple of potential straw man approaches that are practiced in various industrial settings when dealing with feed recommendation problems. We did not find these to be satisfactory in our problem setting.

Approach 1 – Reverse chronological ordering of feed activities: Ranking based on reverse chronological ordering (recency) of activities leads to a fresh but not necessarily a relevant feed. We conducted an online bucket test for three weeks in 2013 comparing pure recency ordering with relevance-based ranking (using our model that served the majority of users at that time), and found the CTR of relevance-based ranking to be 43% higher than that of or-

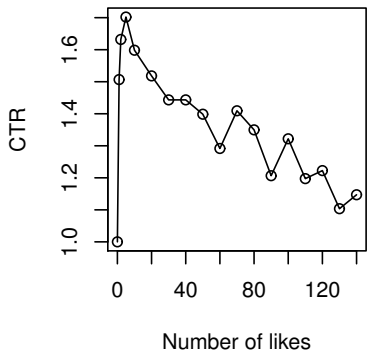


Figure 1: CTR as a function of number of likes.

dering by recency.[1] By the end of 2013, we further improved our relevance model and obtained an additional 15% gain. We believe that a well-designed relevance-based ranking system can significantly outperform recency-based ranking. In our scenario, even after achieving 60% lift (in CTR), there is still more room left for improvement.

Approach 2 – Ranking by social popularity: On LinkedIn, users can like (by clicking on the "like button") or comment on activities. Social popularity can be defined as the number of *likes* (i.e., "like button" clicks) or comments. It may seem at the first glance that ranking activities according to their social popularity is good enough. This is not true in our scenario. Firstly we note that a non-negligible fraction of activities do not receive any like or comment on LinkedIn, despite being shown to a non-negligible set of users. For those that do receive likes or comments, the CTR is not a monotonically increasing function of social popularity as shown in Figure 1. This does not imply that social popularity is not a useful signal, but that it is extremely sparse and requires systematic modeling for leverage.

Contributions: In this paper, we make the following contributions:

- We introduce a taxonomy of activities in social network feeds in Section 3.1.
- We present our system architecture and modeling process (as of 2013) that supports fast model iteration (Section 4). Our main feature storage system (Voldemort[2]) and activity indexing system (Sensei DB [3]) have already been open-sourced. We are also in the process of open-sourcing our large-scale model training library.
- We illustrate the user behavior when interacting with social network feeds based on "randomized" data in Section 3. Randomization helps reduce the bias induced by the ranking methods used during data collection.
- We report on extensive experiments based on online bucket tests (A/B experiments) in Section 5. In particular, we illustrate the effect of freshness, the benefit of impression discounting, and the gains brought in by personalization.

[1]This experiment only considers each user's first feed visit in a session. Subsequent visits in a session are served through other methods.

[2]http://www.project-voldemort.com/voldemort/

[3]http://www.senseidb.com/

Terminology: In the rest of this paper, the terms *activity* and *item* are used interchangeably both referring to a professional network activity. The term *impression* is used to refer to an item's appearance in a user's LinkedIn feed. Because users primarily respond to an activity in the feed through clicks (including clicks on links to see more information and clicks on buttons to share, like or submit a comment on the activity), we combine all these responses and denote it as the click-through rate (CTR) and use it as our primary measure to optimize in this paper.

2. RELATED WORK

Ranking activities from a user's interpersonal network has always been important in social networks. One line of work is to rank homogeneous tweet feeds in Twitter. Chen *et. al.* conducted a set of user studies for information stream recommendation on Twitter. Their purpose was to manually study how various features result in various user preferences to the Tweet feeds. They studied content sources, topic interest of users, and social voting on items in [8]; and thread length, topic relevance and tie-strength in [7], respectively. Their work paved the way for the future automated studies. For example, [1, 11] enriched user profiles with their activities on Twitter for a personalized news recommendation system; and [6] produced different rankers for different sub-communities of a user's social network, where these rankers are described as topics so that the user can rank his/her social feeds by specific topics. As this line of work mainly focused on homogeneous feeds, they are essentially close to the earlier personalized news feed recommendation work [10], except that they used additional social network features like social voting, social tie strength etc. to model the activity relevance. Furthermore, this line of work did not conduct large-scale study.

More recently, researchers started to pay attention to the heterogeneous nature of the network activity ranking problem. Starting from small-scale data, [18] identified different types of user activities from a few users' tweet feeds in Twitter; and [9] investigated the importance of diversity on the Twitter item recommendation and found that content was more relevant to users when it was highly homogeneous or highly heterogeneous. These works inferred heterogeneous activity types from Twitter's text feeds.

Subsequently, researchers started to directly study the heterogeneous data. Berkovsky *et. al.* computed the relevance of the feed items using user-user relationship strengths and user-action interest scores for the social feed personalization in the online Total Wellbeing Diet (TWD) portal [3] and an experimental eHealth portal [2] respectively. These personalized streams attracted more user attention compared to the chronologically ordered feed lists. Paek *et. al.* [19] attempted to understand how people judge the importance of their newsfeed in Facebook by asking some Facebook users to rate the importance of their newsfeed posts as well as their friends'. Classifiers were learned to identify predictive features for the newsfeed and friend relevance. Bourke *et. al.* [4] improved social stream relevance by leveraging features from messages, content source, and the users for Facebook. Soh *et. al.* [20] recommended Facebook social feeds to users by exploiting their response behavior in the past. Based on SocialBlue's news feeds, [12] computed the relevance of network activity feeds from the observed interactions of the individuals in the past. They diversified features into four types: user-user, user-action, user-object and age of activities. Some researchers also investigated the heterogeneous network activity stream personalization problem for enterprise network activity streams [13, 14]. They concluded that the activity stream based user profile is more productive than entity-based user profile for personalizing the stream. Despite the boom of this line in re-

search, all the above work on heterogeneous activity feed ranking has not been evaluated on top of large-scale production systems.

The closest related work is [15], where various machine learning techniques including linear models on features and latent factor models (matrix factorization and tensor factorization) were applied to activity ranking in LinkedIn feed. Their focus was on offline modeling. Here, we present an overview of the end-to-end system and online bucket test results.

3. DATA ANALYSIS

In this section, we study several important aspects of activity ranking based on randomized data. In particular, we assigned a small fraction of users to the *random bucket*. For each user in the random bucket, we shuffle (uniformly at random) the top-100 activities that could appear in the user's feed and present the randomized feed to the user. For details, see Section 4.2. Such randomized data helps to reduce the *serving bias* caused by the ranking algorithm used during the data collection period and the *positional bias* caused by potential high likelihood of placing certain kinds of activities at certain positions in the feed.

We start by introducing different types of activities in the feed. We then show some interesting characteristics of our data. They motivate our relevance modeling described later.

3.1 Taxonomy of Activities

In this section, we present a taxonomy of different types of activities in a social network. In general, each activity can be represented as a triple: (actor, verb, object). For example, member A connects to member B, member A shares article C, and member A updates her profile picture D. Each user has two roles in a social network. On the one hand, a user is an *actor* who generates activities for other users to consume. On the other hand, a user is also a *viewer*, who receives a stream of activities generated by the actors in her network. The problem we study is how to rank this activity stream for each viewer. We note that, in addition to users, there may be other kinds of actors in the network. For example, in the LinkedIn network, companies, schools and content channels are actors as well, which can share different kinds of information.

The *type* of an activity, also called an *activity type*, is a triple of (actor type, verb type, object type), e.g., (member, connect, member), (member, share, article) or (member, profile-update, picture). We broadly classify activities in social networks into the following five categories and also show examples of activities on LinkedIn in Table 1.

1. Connection activities: These are the activities that add new edges in the social network. We can further classify these activities into:

- *Symmetric connection*: For example, the event of a member connecting to another member adds an undirected edge (or more precisely, a bidirectional edge) to the LinkedIn professional network.
- *Asymmetric connection*: For example, the event of a member starting to follow another member or a company (in order to receive what the followee shares) adds a directed edge to the LinkedIn network; the followee will not receive anything from the follower. Another example is the event of a member joining a group (which connects the member to the group).

2. Informational activities: These are the activities where entities in the network pass information to others. In the LinkedIn network, members and companies can share messages, articles, pictures, discussions (in groups) or jobs.

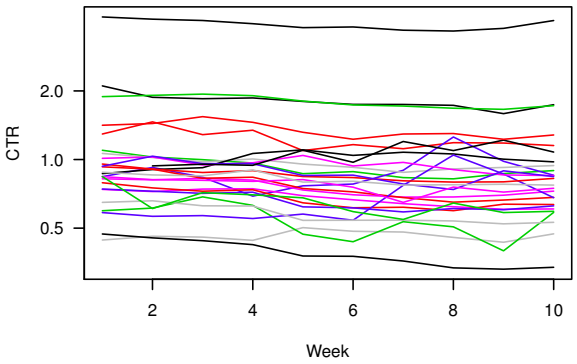


Figure 2: Relative CTRs of different activity types over a 10-week period. The y-axis is log-scaled.

3. Profile activities: These are the activities generated due to users' profile changes. Examples of such profile changes on LinkedIn include updates of profile pictures, contact information, job positions, etc.

4. Opinion activities: These are activities where users express their opinions on different kinds of objects (e.g., articles, pictures, discussions, etc.). Two kinds of opinions are common on social network sites: *like* and *comment*, where the former takes a user little effort to express his/her opinion, while the latter requires a user to put his/her opinions into words.

5. Site-specific activities: The above four categories of activities generally appear on many social network websites. However, each website may also have some activities specific to its purposes or needs. For example, LinkedIn is a professional social network and has job-related activities: job anniversaries, endorsements of users' skills and job recommendations.

Characteristics of CTR of activity types: Users' response to different activity types vary. In Figure 2, we show the CTRs of different activity types relative to the average CTR of all types for users in the random bucket. Each curve represents the CTR of an activity type over a 10-week period. Due to confidentiality reasons, we do not label the curves. However, the key message from this plot is that a few activity types have much higher CTRs than others. Also note that the CTRs of some activity types do not change much over time, while the CTRs of some other types can have large temporal variations.

3.2 Freshness

In this section, we show how freshness of activities affects click-through rate. We consider two aspects of freshness. First, the age of an activity since its creation. Second, the number of times a user has seen a particular item in the past. Note that these two notions of freshness can differ significantly for different users. For example, a heavy user might see an activity several times within the first hour of its lifetime. In contrast, a light user who did not visit the feed for a week would not have seen any activity less than 7 days old.

3.2.1 Effect of Activity Age

To measure the effect of age of activity on CTR, we analyzed the impression and click data collected from the random bucket (described in Section 4.2) over a period of six weeks. Even though the random shuffling mechanism used in the random bucket helps to

Table 1: Taxonomy of activities on LinkedIn

Category	Actor Type	Verb Type	Object Type
Connection	member	connect or follow	member or company
	member	join	group
Informational	member or company	share	article, picture, message, discussion, group, job
Profile	member	profile-update	picture, address, phone, snapshot, job-change
Opinion	member	like or comment	article, picture, message, discussion, group, job
Site-Specific	member	anniversary	job-anniversary
	member	endorse or endorse-by	member
	member	recommend or recommend-by	member

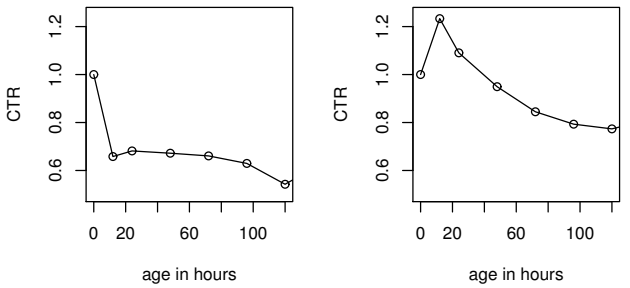


Figure 3: Effect of activity age on CTR for information sharing (left) and for profile picture updates (right). The CTRs have been normalized so that the CTR at age 0 is one.

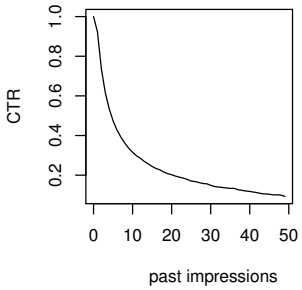


Figure 4: Effect of repeated impressions on CTR.

remove serving and positional biases, the number of times an item has been seen by a user in the past can still confound the analysis. To account for any potential bias due to user fatigue from repeated impressions, we only considered the first impression of an item for each user. The age of an item was coarsely divided into a few buckets. For example, if the age of an item was between zero and twelve hours, it was placed in the zero bucket. If the age was between 12 and 24, it was placed in the bucket corresponding to age 12 and so on. For each of these buckets, we estimated the CTR based on the items that were presented to users.

The CTR curves of two different activity types are shown in Figure 3. Due to proprietary reasons, for each activity type, the y-axis have been normalized based on the CTR of the first bucket. The plots show that the decay of CTR with age can look different for different types of activities. In the case of profile picture updates, there is a spike in CTR a few hours after the activity is created which then drops down subsequently. In contrast, in the case of information sharing there is a sharp drop in CTR initially followed by a much slower decay from there on.

3.2.2 Effect of Repeated Impressions

To determine the variation in CTR with repeated impressions, we also use the random bucket data. For each activity, we counted the number of past impressions of the activity for each user. If an item never appeared in a user's feed before, the number of past impressions for that (user, item) pair would be zero.

The behavior of CTR with the number of previous impressions of the same activity is shown in Figure 4. In this plot, the x-axis shows the number of times an activity was seen before and the y-axis shows the corresponding CTR. The plot has been normalized by the CTR of the first impression. It is obvious from the plot that the CTR of an activity decreases as it is seen multiple times. This curve suggests the need for discounting the predicted CTR based on the number of previous impressions of an activity.

3.3 Diversity of the Feed

LinkedIn feed consists of diverse types of activities as introduced in Section 3.1. In this section, we analyze how CTR changes with respect to the diversity in the feed using the random bucket data. For each feed that was presented to a user, we consider the ID of the actor of the activity at position 10 and count the number of times activities from the same actor appeared in positions 1 to 9. When this count is high, this corresponds to a scenario where most activities in the feed have the same actor. In other words, a high count reflects lack of actor diversity in the feed. Similarly, we considered the counts of the same verb type or same object type in positions 1 to 9. To avoid any bias cased by users with very few connections (thus, it is unlikely to have diverse feeds for them), we restricted this analysis to only those users who had at least 50 connections. The decay of CTR with respect to repeated actor ID, verb type and object type is shown in Figure 5. In all the plots, the y-axis have been normalized so that the CTR in the case of no repetition (i.e., #repetitions = 0) is one. It can be seen that for all the three different types of repetitions, there is a rapid initial decay in CTR.

3.4 Connection Relationship

Since members on LinkedIn are connected via a professional network, we can study how the connection relationship between a viewer and an actor affects whether the viewer would click on an activity of the actor. A few examples of connection relationship between a viewer and an actor are in the following.

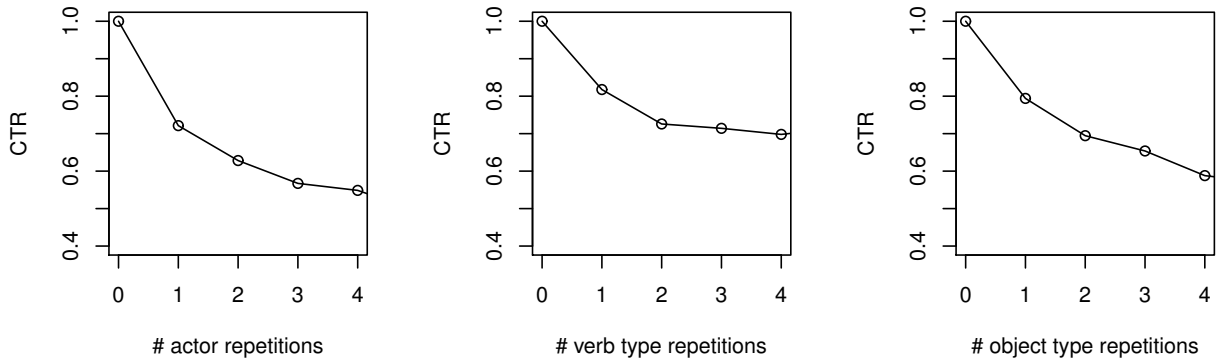


Figure 5: Effect of (lack of) diversity on CTR for actor ID, verb type and object type.

Table 2: CTR lift when the viewer and the actor have the same characteristic compared to the case where they do not.

Viewer-Actor Relationship	CTR Lift
Same Company	66%
Same Job Function	25%
Same Industry	33%
Same Geo Region	24%

- Demographic similarities based on age, gender and education, etc.
- Experience similarities based on the common companies, job positions, etc.
- Skill similarities based on the common skills and similar skills.
- Geo similarities based on the locations of the viewer and the actor at different resolutions (e.g., city, state and country).
- Social network similarities based on connections of the viewer and the actor.

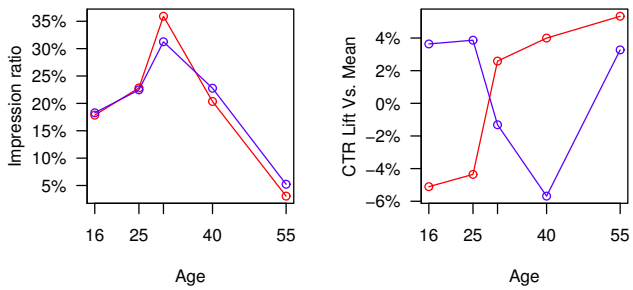
Effect of connection relationship: We now present some observations from our data with respect to a couple of these similarity measures by looking at whether the viewer and the actor belong to the same company, have the same job function, are in the same industry or in the same geo-region. Table 2 shows that when the viewer and the actor have similar characteristics, there is a higher chance that the viewer would click on an activity of the actor. However, such behavior also depends on the type of activity. For example, consider whether the viewer and actor are in the same company or not. Averaged over all activity types, the CTR of "same company" is 66% higher than the CTR of "different companies". However, Table 3 shows that it is not the case for job-change type of activities. A viewer clicks more often on a job-change of an actor in a different company than a job-change of an actor in the same company, perhaps due to the high likelihood that the viewer already knew her connections in her company changed jobs.

3.5 Platform Dependence

From our analysis, we observe that users' interaction with activities differ significantly across the mobile and desktop platforms. We first looked at the impression distribution across different age groups of the viewer. It is evident from Figure 6a that the distribution of impressions is similar across age groups on the two platforms. We then looked at the CTR by age groups on these two platforms. This is shown in Figure 6b. Due to proprietary reasons, the plot only shows the deviation of CTR for an age group compared to the overall CTR on that platform. Surprisingly, although the impressions distribution is similar on the two platforms, different age groups respond differently on mobile and desktop. This example shows differences in user behavior across different age groups on different platforms. It is important to analyze users' behavioral differences on different platforms and build models that capture the behavior specific to each platform. As another example, we will show in Section 5 that a mobile specific model performs significantly better than simply using a desktop model to rank activities on our mobile feed.

Table 3: CTR lift when the viewer and the actor are in the same company for different types of activities.

Activity Type	CTR Lift
(member, post, message)	207%
(member, share, article)	111%
(member, connect, member)	72%
(member, profile-update, job-change)	-10%



(a) Impression distributions (b) CTR lift

Figure 6: Behavior of different age groups of viewers on mobile (red) vs. desktop (blue)

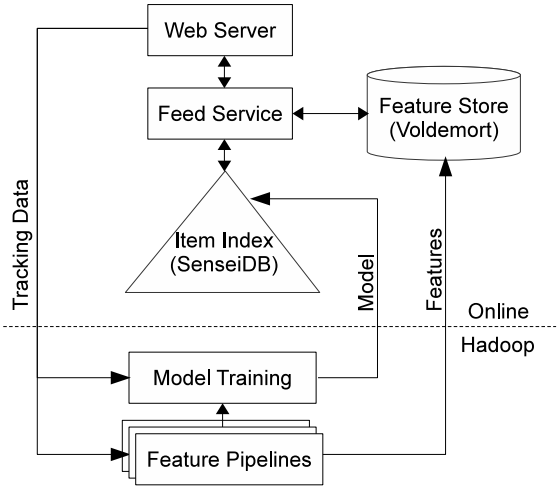


Figure 7: System architecture (simplified).

4. RELEVANCE SYSTEM

In this section, we first give an overview of our system as of 2013 and then outline our model training process which includes data collection, large-scale logistic regression and an offline replay methodology for model evaluation. The primary goal of relevance modeling is to accurately predict the CTR of a given user-item pair, i.e., the probability of the user clicking on the item. In additional to CTR prediction, it is also important to incorporate mechanisms to ensure freshness and diversity. These will be discussed in Section 4.5.

4.1 System Overview

At a high level our system consists of online components for serving users' feed requests, and also offline components for generating features and training models on Hadoop. Figure 7 shows the main components of our system and their interaction with one another. We would like to mention that two of these important components, the feature store (Voldemort) which provides large-scale feature storage and retrieval, and the item index (SenseiDB) which indexes items and ranks them according to a scoring model, have already been open-sourced and we are in the process of open-sourcing the model training component.

When a user logs in to LinkedIn, the web-server sends a request for the feed to the feed service. Activities generated on the LinkedIn professional network are stored in SenseiDB. When the feed service receives the ID of the viewer, it retrieves the appropriate model for the viewer (depending on the online bucket test running at that time) and sends the model to SenseiDB to score the candidate activities for the viewer stored in SenseiDB. The SenseiDB stores item specific features (for example: activity type, creation time etc). Some of our models incorporate additional features that are either not item specific or are derived through additional computation. For example, we use a score for the affinity of the viewer to each activity type in our models. Such features are generated by separate feature pipelines on Hadoop, and pushed to a Voldemort store which is a distributed key-value store. The feed service can then access these features efficiently from the Voldemort store. The feature pipelines generate features from the tracking data which is a log of impressions (with basic features of activities) and clicks. The features produced by the feature pipelines are also used in the training process which generates models for use in SenseiDB. More details on model training, features used and evaluation are provided next.

4.2 Training Data Collection

Collection of training data for our application can be a non-trivial exercise. In the absence of a good ranking model, a seemingly natural option is to rank activities in a reverse chronological order and present to the viewers. The response information (click/no-click) thus collected could then comprise the training dataset. However, this method of training data collection can severely suffer from serving bias in the following way. Usually viewers' attention is drawn primarily to the first few results in the feed. So if the serving scheme never serves activities of certain types to a viewer, the viewer cannot give his/her feedback on such activities. At LinkedIn, the frequencies at where different types of activities are generated are quite different. For example there are many more member connect member updates in the candidate set of activities than some other of the other types. This implies that a reverse chronologically sorted stream of a typical viewer would comprise of a large number of connection activities and other infrequent activities, like job changes, will rarely, if ever, get impressed on the viewer although they might be more relevant. This leads to data sparsity problem for a number of activity types.

The most common approach for eliminating serving bias is random serving, where activities in the candidate pool are ranked randomly. However, this scheme is likely to cause very poor user experience due to the following two reasons. First, many irrelevant and stale activities might get promoted to the top positions. Second, the set of activities presented at the first few positions would change dramatically upon each page reload. Moreover, the feed would still be dominated by a few types of activities that are more common than the others.

We overcome the aforementioned problems in the following way. We first use a ranking algorithm to rank activities. Then, the top-k results are permuted uniformly at random to prepare the final feed for the viewer. This scheme removes the serving bias from the training data to a large extent, without completely sacrificing user experience. Our training data consists of activities that were presented using this mechanism. The activities on which a viewer clicked are considered positive examples and those with which the viewer did not interact are considered negative examples. Such training data comes at the cost of reduced user experience due to randomization. Since model training/re-training is a continuous process which entails persistent availability of fresh training data, this cost must to be minimized. This is done by serving this randomized feed to only a small fraction of our viewers, and frequently changing this bucket of viewers. Also, we keep updating the underlying ranking model (which produces the top-k results for randomization) to the current best model. This ensures an improving user experience even for those viewers who happen to fall in the random serving bucket.

4.3 Model and Features

A score is required to rank the activities in the candidate pool. As discussed earlier, we use the predicted CTR as the score. To predict the CTR of each activity, we learn a logistic regression model with ℓ_2 regularization. Let y denote the click on the activity described by feature vector $\mathbf{x}$ then the corresponding Bernoulli random variable $\mathbb{Y}$ is modeled as

$$E[\mathbb{Y}] = \frac{1}{1 + \exp(-\boldsymbol{\theta}^\top \mathbf{x})} \quad (1)$$

where $\boldsymbol{\theta}$ is the parameter vector which we want to learn. Once $\boldsymbol{\theta}$ has been estimated from the training data, for any new example

$\mathbf{x}_{\text{new}}$ we simply use the mean of the Bernoulli distribution as the predicted CTR, i.e.,

$$\text{CTR}_{\text{predict}} = P(\{\mathbb{Y} = 1\}) = E[\mathbb{Y}] = \frac{1}{1 + \exp(-\boldsymbol{\theta}^\top \mathbf{x}_{\text{new}})} \quad (2)$$

We train a logistic regression model with a number of features, such as features extracted from the viewer's and actor's profile, features qualifying the connection strength and similarity between the viewer and the actor, features representing the age of an activity, features quantifying the affinity between the viewer and the type of an activity and the affinity between the viewer and the actor. Note that some of the features are generated using other models — logistic regression can be thought of as a tool to combine such base models. We also include interactions among the aforementioned features.

4.4 Model Training and Offline Evaluation

We train the logistic regression model using a Hadoop based distributed logistic regression algorithm. Our implementation of logistic regression is based on the Alternating Direction Method of Multipliers (ADMM) [5] which is a popular method for solving a convex optimization problem in a distributed fashion. We are in the process of making our implementation open source so that others facing similar challenges can benefit from it. The ADMM algorithm partitions the set of training examples into several blocks. Each block is assigned to a machine which solves (in parallel) a convex optimization problem independently on the training examples in that block. Solutions from different partitions are collected and a consensus solution is formed. The consensus solution is then sent to individual machines which update their solutions again. This process is repeated until convergence. More details can be found in [5]. This method is scalable and it allows us to train our models with a large number of training examples collected from our random bucket.

After we train a model, we need to evaluate the model before deploying it in production. For this, we use a replay methodology for unbiased offline evaluation of online serving schemes [17, 16]. We use the offline replay methodology since it is a theoretically sound way of evaluation. Typically we consider a few weeks of data from the random bucket (described in Section 4.2), starting from a point in time after the period over which the training data was collected. For replay, the activities that were presented at serve time in the random bucket are reordered using the candidate model obtained via training described above. After reordering, we only consider impressions that occur at exactly the same position as they appeared at serve time. We call these matched impressions and count the total number of clicks on such matched impressions. We call the total number of clicks on matched impressions as reward. We typically consider reward at the first feed position and the reward at the first three feed positions when comparing our models. Although the absolute value of reward may not be very meaningful, the relative rewards obtained from two different models are typically indicative of which model would perform better in production.

4.5 Reranking

Our model training described above focused only on accurately modeling the CTR of each user-activity pair. However, we pointed out in Section 3 that the click-through rate can also be influenced by aspects such as diversity and freshness.

Although relevance and diversity can be modeled jointly, such approaches typically require greedily picking the next activity based on the current set of items, which can be computationally prohibitive in a real online system serving millions of users. Hence we use a simple mechanism to enforce diversity in the feed. In this mechanism, we first score and sort all the candidate activities using the ranking model. We then go through the ranked list and count the number of times a particular actor or object appeared on the list before each item. The score of each item is discounted based on this count in a way that more prior occurrences of the same actor or object leads to a stronger discounting. In our experiments, we simply fix this reranking mechanism for all our models.

It is also important to keep the feed fresh. However, simply learning age based and impression count based coefficients from the data may not provide desired level of feed freshness. Since we also do not want to give a completely time sorted feed, we perform an additional exponential decay on the score based on the age of activities. The half lives of this decay are set in accordance with product requirements (which is influenced by user experience studies). Similarly, repeated impressions of the same activity is also discouraged by introducing an additional decay factor that takes into account past impression counts of the activity. These reranking modules allow an easy way of tuning the feed to match required characteristics.

5. EXPERIMENTS

In this section we describe several online bucket tests (A/B experiments) on the LinkedIn feed and a few offline replay experiment results. Most of our experiments are based on a large number of real users who visited LinkedIn. LinkedIn currently has more than 300 million members geographically spread across the world. A large fraction of visitors see the LinkedIn feed.

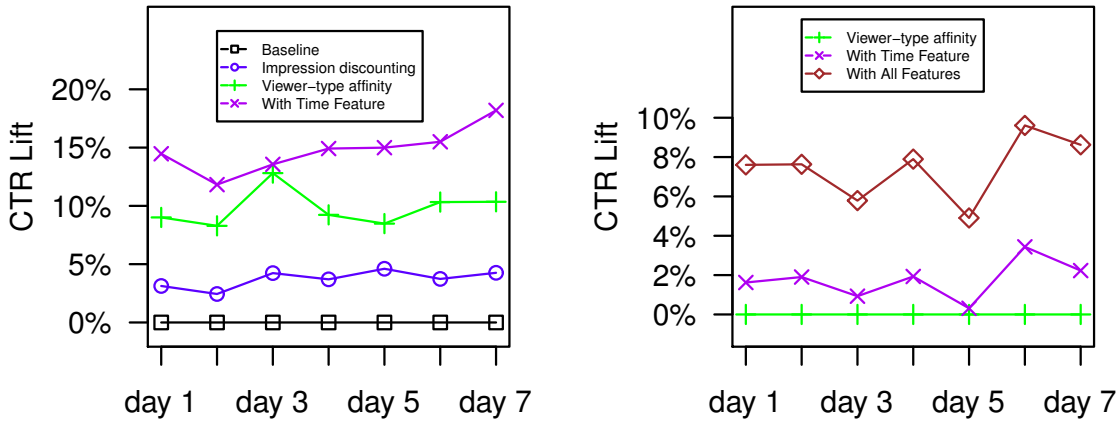
We continuously test new features and new models in our production system. We typically launch a new feature to a small fraction of users and if the feature works well compared to existing models, we slowly ramp the feature to more users. At any point in time, we are testing several models in production. It is impossible to present results from all the models that were in bucket tests. In this section, we present several interesting experiments focusing on just one aspect at a time. We also present some results where we combined a number of useful features in our production system.

5.1 Effectiveness of Replay Analysis

We described our offline replay evaluation metric in Section 4.4. In this section, we show the offline replay results and the corresponding online replay results for two models. We considered a baseline model in production that was constructed from the features mentioned in Section 4.3. We refer to this as Model A. We then added features derived from networks of viewer and actor and learned a new model using our training pipeline. We denote this model as Model B. In offline replay analysis, Model B showed a lift of 4.96% at the first feed position over Model A. When we ran these models in production for two weeks, the lifts that we observed for model B in those two weeks were 3.6% and 5.0% compared to model A. Although the absolute value of the performance improvement in the first week is slightly different from the offline lift for most of the models we tried in production, we have found that replay provides a sound way for deciding whether a model is a promising candidate for deployment. In the rest of this paper, we provide bucket test results from models deployed online rather than focusing on offline analyses.

5.2 Desktop Bucket Test Results

In this section, we show the online bucket test results using several new features that we incorporated into our modeling. First, we show the effect of adding them independently and finally we show the effect when we combined all the features.



(a) Individual features

(b) All features

Figure 8: Desktop online CTR lift.

Effect of freshness.

As we discussed in Section 4.5, we control the freshness of the feed using an exponential decay on the score based on the age of the activity. To show how different segments of users behave differently to different levels of freshness, we consider two models that we had in our bucket tests. The two models were exactly the same except for the half life period on the exponential decay. Model A had an exponential decay with a half life of four days (which means the score predicted by the model halves every four days), while model B had an exponential decay with a half life of two days. Although the overall CTR for model B was slightly higher than that of model A, the results are more interesting when we segment users and look at the lifts on the individual segments. Users were divided into four segments (light, moderate, regular, heavy) based on the number of visits to LinkedIn feed. For heavy users, model B had a lift of about 25%, while for regular users model B had a smaller lift of about 7%. However, for moderate and light users, model A had a lift of 2.3 % and 6.6% respectively over model B. These results intuitively make sense. Visitors who come to LinkedIn more often expect to see fresher content, whereas visitors who visit LinkedIn less often are happy to consume older content.

Impression Discounting.

In the previous paragraph, we described bucket tests with age based freshness as the criterion. We now consider the effect based on repeated impressions of items. As we described in Section 3.2.2, we first estimated how the CTR decays with repeated impressions of activities. One challenge for impression discounting is that we have to store the number of past impressions to every activity that is served to a user. This requires a large key value store, where key is the ID of an activity and value is the number of past impressions of that activity for each viewer. We make use of a Voldemort store which was described in Section 4.1. We store the number of past impressions for several millions of activities such that these counts are available at runtime on our production systems. We then applied impression discounting to our best model at that time. By applying impression discounting, there was an improvement in CTR of about 2.2% during a period of one week. However, for our heaviest user segment (who visit our feed more than 10 times on an average per day), there was an improvement of over 5%.

Model Personalization.

In this section, we show results from couple of model improvements that we achieved through personalization. Our first approach was to develop an affinity score between viewers and activity types. As we described in Section 3.1, there are several activity types on the LinkedIn feed. Since different LinkedIn members can have different affinity towards different types of activities, we model the affinity between users and activity types. This affinity score was devised based on historic data of interaction between viewers and activity types. The basic idea of this score is to segment users based on their features (such as industry, job seniority, company size, language etc.). For each segment, we look at those user's historic data to collect how often they interacted with different activity types. From all this information, we compute an affinity score which is then used as a feature in modeling the click-through rate prediction via logistic regression. A second approach was to use a personalized age based decay for every user depending on time based features. This involved utilizing the last visit time of a user to LinkedIn. This feature effectively personalizes the feed such that heavy users tend to get fresher content. We note that the model with the time based feature also included impression discounting when we tested in production.

We trained models that consisted of each of the above feature individually combined with the features in our baseline model. The improvement from adding viewer activity type affinity features and personalized decay (with impression discounting) were 6.5% and 11.5% respectively over a period of one week. The results from our bucket tests over the days of the week are also shown in Figure 8a. While there are some variations over days, the overall trend clearly indicates performance lifts from personalization and impression discounting.

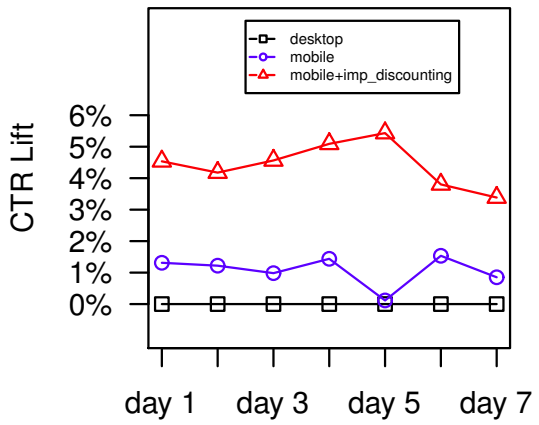


Figure 9: Mobile CTR lift over 7 days.

Combining All The Features.

Previous paragraphs showed how adding each feature individually improved user engagement. We then trained a model jointly with all the features described in the previous paragraphs (i.e., impression discounting, affinity feature and time based features) combined with the features in our baseline model. Since we usually discontinue inferior models in our production system, the result in Figure 8b only compares the all-inclusive model against the best performing models from Figure 8a which were active at the same time. As evident from Figure 8b, over a period of one week, the combined model ("With All Features" in figure) achieved a notable improvement of 7.3% over the model including only viewer activity type affinity ("Viewer-type affinity"), and 5.1% over the model including only personalized decay and impression discounting ("With Time Features").

5.3 Mobile Bucket Test Results

To understand the effectiveness of using the mobile specific models, we performed a bucket test for a period of 7 days. The results are shown in Figure 9. We first partitioned our mobile user base into 3 buckets. We then use the CTR performance of the best desktop model (trained on data from our desktop feeds) to serve traffic on mobile as the baseline (the curve labeled "desktop" in Figure 9). We also had a model which was trained using data collected from the random bucket. The mobile specific model (labeled "mobile") consistently outperformed the desktop model. A mobile model with impression discounting (labeled "mobile+imp_discount") had a significant lift over the 7 day's period.

6. CONCLUSION

We introduced the problem of ranking activities in LinkedIn feed. We showed the diverse nature of our feed through a taxonomy of activities. We analyzed various characteristics of our data and argued that a relevance system should take into account many of these characteristics to be successful in production. We then described our system architecture and showed how we collect training data, generate features, continuously train models using a distributed algorithm and evaluate those models offline before launching them in production. Finally, we showed several bucket test results which show how personalization and other modeling efforts help to improve the system.

Here are some lessons that we learned during the course of this work:

- Good ranking model are found by continuously testing of different models with different features. it is important to have a modeling framework that allows fast iteration over model training, deployment and experimentation.
- While learning from data can show impressive gains in the performance of a system, the models learned from data may not have all the characteristics we desire. For example, we learned from data that CTR decays over time. However, the learned decay factors do not always generate a fresh feed for each user visit. Other mechanisms are sometimes required to ensure certain desired user experience.
- Different platforms (such as mobile vs desktop) have different characteristics. Simply porting a desktop model to serve mobile feeds may not necessarily work well.
- Personalizing our models for user's tastes and behavior often gives large improvements in performance.

7. REFERENCES

[1] F. Abel, Q. Gao, G.-J. Houben, and K. Tao. Analyzing user modeling on twitter for personalized news recommendations. In *UMAP'11 Proceedings of the 19th international conference on User modeling, adaption, and personalization*, pages 1–12, 2011.

[2] S. Berkovsky, J. Freyne, S. Kimani, and G. Smith. Selecting items of relevance in social network feeds. In *UMAP'11 Proceedings of the 19th international conference on User modeling, adaption, and personalization*, pages 329–334, 2011.

[3] S. Berkovsky, J. Freyne, and G. Smith. Personalized network updates: Increasing social interactions and contributions in social networks. In *UMAP'12 Proceedings of the 20th international conference on User Modeling, Adaptation, and Personalization*, pages 1–13, 2012.

[4] S. Bourke, M. P. O'Mahony, R. Rafter, and B. Smyth. Ranking in information streams. In *Proceedings of the companion publication of the 2013 international conference on Intelligent user interfaces companion*, pages 99–100, 2013.

[5] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. Technical report, Stanford University, 2010.

[6] M. Burgess, A. Mazzia, E. Adar, and M. Cafarella. Leveraging noisy lists for social feed ranking. In *AAAI*, 2013.

[7] J. Chen, R. Nairn, and E. H. Chi. Speak little and well: Recommending conversations in online social streams. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 217–226, 2011.

[8] J. Chen, R. Nairn, L. Nelson, M. Bernstein, and E. H. Chi. Short and tweet: Experiments on recommending content from information streams. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1185–1194, 2010.

[9] M. D. Choudhury, S. Counts, and M. Czerwinski. Identifying relevant social media content: Leveraging information diversity and user cognition. In *Proceedings of the 22nd ACM conference on Hypertext and hypermedia*, pages 161–170, 2011.

[10] W. Chu and S.-T. Park. Personalized recommendation on dynamic content using predictive bilinear models. In *WWW*, 2009.

[11] G. D. Francisci, A. Gionis, and C. Lucchese. From chatter to headlines: Harnessing the real-time web for personalized news recommendation. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 153–162, 2012.

[12] J. Freyne, S. Berkovsky, E. M. Daly, and W. Geyer. Social networking feeds: Recommending items of interest. In *RecSys*, 2010.

[13] I. Guy, I. Ronen, and A. Raviv. Personalized activity streams: Sifting through the "river of news". In *RecSys*, pages 181–188, 2011.

[14] I. Guy, T. Steier, M. Barnea, I. Ronen, and T. Daniel. Swimming against the streamz: search and analytics over the enterprise activity stream. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 1587–1591, 2012.

[15] L. Hong, R. Bekkerman, J. Adler, and B. D. Davison. Learning to rank social update streams. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 651–660, 2012.

[16] J. Langford, A. Strehl, and J. Wortman. Exploration scavenging. In *Proceedings of the 25th International Conference on Machine learning*, 2008.

[17] L. Li, W. Chu, J. Langford, and X. Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 297–306. ACM, 2011.

[18] M. Naaman, J. Boase, and C.-H. Lai. Is it really about me? message content in social awareness streams. In *Proceedings of the 2010 ACM conference on Computer supported cooperative work*, pages 189–192, 2010.

[19] T. Paek, M. Gamon, S. Counts, D. M. Chickering, and A. Dhesi. Predicting the importance of newsfeed posts and social network friends. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10)*, 2010.

[20] P.-H. Soh, Y.-C. Lin, and M.-S. Chen. Recommendation for online social feeds by exploiting user response behavior. In *WWW*, 2013.