Co-Factorization Machines: Modeling User Interests and Predicting Individual Decisions in Twitter

Liangjie Hong Aziz S. Doumith Brian D. Davison
Dept. of Computer Science and Engineering, Lehigh University, Bethlehem, PA, USA
{lih307, asd309, davison}@cse.lehigh.edu

ABSTRACT

Users of popular services like Twitter and Facebook are often simultaneously overwhelmed with the amount of information delivered via their social connections and miss out on much content that they might have liked to see, even though it was distributed outside of their social circle. Both issues serve as difficulties to the users and drawbacks to the services.

Social media service providers can benefit from understanding user interests and how they interact with the service, potentially predicting their behaviors in the future. In this paper, we address the problem of simultaneously predicting user decisions and modeling users' interests in social media by analyzing rich information gathered from Twitter. The task differs from conventional recommender systems as the cold-start problem is ubiquitous, and rich features, including textual content, need to be considered. We build predictive models for user decisions in Twitter by proposing Co-Factorization Machines (CoFM), an extension of a state-of-the-art recommendation model, to handle multiple aspects of the dataset at the same time. Additionally, we discuss and compare ranking-based loss functions in the context of recommender systems, providing the first view of how they vary from each other and perform in real tasks. We explore an extensive set of features and conduct experiments on a real-world dataset, concluding that CoFM with ranking-based loss functions is superior to state-of-the-art methods and yields interpretable latent factors.

Categories and Subject Descriptors: H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; H.4 [**Information Systems Applications**]: Miscellaneous

Keywords: Latent Factor Models; Recommender Systems; Collaborative Filtering; Twitter

1. INTRODUCTION

In the current online social media ecosystem, users are able to connect and communicate with each other utilizing rich multimedia content, including text, images, video, and audio. These communication streams allow users to be informed instantly of the latest updates from their social connections. As a result, the aggregated social content has become a powerful tool for monitoring critical information in various situations, such as natural disasters and political upheavals.

Although services like Twitter and Facebook provide platforms for users to obtain fresh information, two issues prevent them from being sufficiently relevant, causing deterioration of user experience and engagement. First of all, when facing a large amount of content from their social connections, users simply cannot consume it in an effective and efficient way, leading to the problem of *information overload*. On the other hand, information for a user is usually limited in scope to the user's social connections. Thus, it is difficult for a user to obtain information distributed outside of their circle, even though it might match their interests, leading to the problem of *information shortage*. Users may spend a significant amount of time filtering and searching for relevant information in social media. From the perspective of service providers, it is also very important to understand how users interact with the systems through a variety of actions such as re-posting (retweets), replying and commenting. It is also indispensable to track what in which users are interested, induced from the content requested and generated by them. Therefore, social media services can filter and recommend content to users based on their the history of previous interactions and interests.

The task of understanding users' behaviors and their interests has a number of challenges. First, although the number of items (updates, tweets, etc.) generated by users in such services may be huge, few of them impact a particular user, making the interaction data sparse. Second, new users and new content items flow into the system continuously. Thus, the "cold start" problem tends to be severe in these social platforms, compared to traditional information systems. In addition, a tremendous amount of content is rich yet noisy. Simple information retrieval or topical modeling techniques may not be sufficient to capture users' interests.

The problem tackled in this paper has strong links to research in recommender systems and collaborative filtering. From the perspective of recommender systems, the task can be cast as building a list of relevant content items to users based their social connections and interests. Thus, many collaborative filtering techniques are applicable to the task. However, on the other hand, as we mentioned, social content systems are much more dynamic than traditional recommender systems: many new items are pushed into the system every second. Therefore, recommender systems need to be adapted to this novel situation.

State-of-the-art collaborative filtering models are based on latent factor models (LFM), partially due to their superior performance in the Netflix competition [17]. However, the basic assumption for standard LFM is to exploit a user-item interaction matrix and cannot handle arbitrary features easily. Although some of newly proposed frameworks such as [1, 7], based on LFM, can consider features, fundamental modeling assumptions prevent them from handling high-order interaction data (e.g., tensors). In addition, current extensions to LFM [2, 25, 28] which incorporate rich text information are usually cumbersome, requiring complicated inference algorithms which cannot scale to large datasets. Moreover, researchers in collaborative filtering are realizing that pointwise-based measurement may no longer be appropriate and so a handful of ranking-based metrics are proposed. However, no work to date has compared them systematically on real world datasets.

In this work, we study the problem of modeling users' behaviors by focusing on one particular decisions, retweets, in Twitter and try to understand users' interests. Our method can be easily extended to model multiple types of users' decisions as well. We use a state-of-the-art recommendation model, Factorization Machines FM [22], to model user decisions and user-generated content simultaneously. Our contributions can be summarized as follows:

- We propose Co-Factorization Machines (CoFM), which deal with two (multiple) aspects of the dataset where each aspect is a separate FM. This type of model can easily predict user decisions while modeling user interests through content at the same time.
- We apply Factorization Machines to text data with constraints. Thus, the resulting method can mimic state-of-the-art topic models and yet benefit from the efficiency of a simpler form of modeling.
- For user decision modeling, we compare a number of ranking-based loss functions. We introduce the newly proposed WARP loss [26], which has been successfully applied in text and image retrieval tasks (e.g., [31, 32]), into the context of recommendation.
- We apply our proposed methods to the problem of modeling personal decision making in Twitter and explore a wide range of features, revealing which types of features contribute to the predictive modeling and how content features help with the prediction.

We next review several directions of related work. In Section 3, we review FM: its basic settings and learning procedure. In Section 4, we formalize CoFM with different strategies of shared latent features. In Section 5, we compare and discuss a variety of loss functions. In Section 6, we describe features used in our model. In Section 7, we report the experiments with the discussions of datasets and baselines used. We conclude the paper in Section 8.

2. RELATED WORK

In this section, we review three lines of relevant research work: 1) collaborative filtering and ranking, 2) collaborative filtering with content integration, and 3) Twitter user and content modeling. We link them with our tasks and discuss the novelty of our work as well.

Recommender systems which utilize LFM have gained significant attention because they were used by the winning team of the Netflix Prize. However, simple LFM cannot easily be coupled with additional information (features) in other recommendation scenarios. Recently, researchers have explored how traditional LFM can be enhanced by exploiting rich features generated by users. Three main paradigms are proposed for this purpose. The first paradigm is "Regression Based Factor Models" (RBFM) and its extensions, proposed by Agarwal, Chen and colleagues [1, 2, 3, 36], which have been successfully used in a variety of recommendation scenarios, such as social networks [33, 34], professional networks and content recommendation. The basic idea behind RBFM is to replace zero-mean Gaussian distributions usually used in a simple LFM with regression-based means. Thus, RBFM adds another layer of regression on top of LFM, which can incorporate different types of features effectively. However, RBFM can only handle 2-order data interactions and thus high-order data structures (e.g., tensors) cannot be modeled. In addition, the proposed method, training with Monte Carlo EM, is inefficient and cannot scale to large scale datasets easily. The second paradigm is called "Factorization Machines" (FM), proposed by Rendle et al. [22]. FM can handle, in theory, arbitrary orders of interactions between variables and naturally deal with features. However, the existing formalism of FM has not been explored with topical modeling of content and pairwise preferences learning has not been discussed in the context of FM. The last paradigm is called "Feature-based matrix factorization" (FBMF), proposed by Chen et al. [7], which is to combine LFM with linear regression. As noted by Rendle [22], similar to RBFM, FBMF cannot handle high-order interactions either. Additionally, FBMF can be viewed as a special case of FM.

Traditional collaborative filtering methods are trained and evaluated on pointwise-based measures, such as Root-Mean-Square-Error (RMSE) and Mean-Square-Error (MSE), essentially measuring how accurate each single prediction is, regardless of how items are presented to users. Although the use of RMSE gained popularity through the Netflix prize competition, ranking-based measures might be more appropriate than pointwise-based measures since users are presented a ranked list of items calculated by recommendation systems. Thus, it would be more natural to optimize the ranked list directly. Some recent advances in recommender systems lie in this direction. For instance, Weimer et al. [29] extended the maximum margin matrix factorization method (MMMF) by optimizing a surrogate loss function approximating the NDCG ranking measure, a ranking-based metric commonly used in the Information Retrieval (IR) community. However, the method proposed in the paper is complicated and arguably hard to scale to large datasets. A margin ranking criterion, an ordinal loss, is introduced from the field of IR to collaborative filtering by Weimer et al. [30] in the context of MMMF, which is a direct extension of the hinge loss for Support Vector Machines. This max-margin loss essentially minimizes AUC, which is the area under the ROC curve [4, 31]. A smooth version of hinge loss which is also to minimize AUC, called Bayesian personalized ranking, is proposed by Rendle et al. [23] and has yielded superior performance in tag recommendation. Recently, Balakrishnan and Chopra [5] proposed a two-stage procedure for collaborative ranking. Their proposal is to first train a matrix factorization model for users and items and treat latent factors as features to feed into a standard learning-to-rank framework. Koren and Sill [18] proposed a method to embed ordinal regression into matrix factorization by predicting a full distribution over all ranks. An interesting point is that this method is indeed a pointwise method. With a slightly different approach, Yang et al. [34] studied user behaviors when browsing a list of items. The proposed framework includes two loss functions that are comparable to multinomial logic model and a margin ranking criterion but have more intuitive explanations. Though different ranking-based loss are proposed, they are never compared.

Some recent developments in collaborative filtering have demonstrated the power to integrate rich content from articles and scientific papers with user decisions to provide better recommendation results. For instance, Agarwal and Chen [2] extended RBFM with a latent Dirichlet allocation prior for latent factor models. A similar approach was investigated by Shan and Banerjee [25] as well. Wang and Blei [28] proposed a method to combine matrix factorization with probabilistic topic modeling for recommending scientific papers. The method cannot easily leverage additional features. One significant advantage of these joint modeling methods is that latent factors obtained through content modeling can reveal interpretable meanings to latent spaces and thus provide a unique way to uncover some hidden structures of the data. However, all these methods require complicated inference algorithms which are not easily to scale to large datasets.

Recommender systems have been built specifically for Twitter. For instance, Kim and Shim [16] argued that Twitter does not offer to find the most interesting tweet messages for users. The authors proposed a probabilistic model derived from Probabilistic Latent Semantic Analysis (PLSA) for collaborative filtering to recommend potential followers to users in Twitter. The method does not consider any explicit features at all. Due to the fact that Twitter users form a information network, researchers have tried to use undirected graphical models to model such networks. For example, Yang et al. [35] used a factor graph to model the spread of tweets. Lin et al. [20] proposed a generic joint inference framework for topic diffusion and evolution in social network communities based on Gaussian Random Fields, which also cannot integrate with rich features. Similarly, Peng et al. [21] proposed a method based on Conditional Random Fields (CRF) to predict how likely a tweet will be retweeted by a user. The proposed method suffered from the difficulty to efficiently perform inference on graph-like CRFs. Duan et al. [9] studied how learning to rank approaches can be used in ranking tweets. They explored a number of features and used 20 query terms as input to train a RankSVM as the model. In the present work, we do not have explicit queries while modeling user decisions and user-generated content. Hong et al. [14] and Uysal and Croft [27] trained classifiers to predict whether a tweet will be retweeted. However, the classifiers they trained are universal for all users and hence cannot provide personalized results. Recently, Chen et al. [6] utilized FBMF with a wide range of features to recommend tweets for users on Twitter. However, the proposed method cannot provide much insight on how content contributes to users' decisions and only one type of ranking loss function is used without comparisons. In all, these methods either do not handle arbitrary features or do not obtain summarized content (topics) from Twitter messages, preventing us from further understanding how users' decisions are made. In this work, we will perform these two tasks simultaneously.

3. MODELING TWITTER BY FM

In this section, we first review the basic settings of FM with a discussion of how it can be used for different types of responses. Then, we detail how FM can fit into the task of modeling Twitter data, which is a predictive modeling for user responses and a topic coding model for content.

3.1 Factorization Machines

Here we briefly review FM, a state-of-the-art framework for latent factor models with rich features. For a detailed description, please refer to Rendle [22]. We start from a design matrix $\mathbf{X} \in \mathbb{R}^{D \times P}$, where the ith row $\mathbf{x}_i \in \mathbb{R}^P$ denotes one data instance with P real-valued variables and where y_i is a response for the data instance. We use s_i to represent our estimation of y_i based on $\mathbf{x}_i$. In many tasks, the goal is to make the discrepancy between s_i and y_i as small as possible. The factorization machine (FM) model of order $d = 2$ for $f(y_i \,|\, \mathbf{x}_i, \mathbf{\Theta}) = s_i$ can be defined as:

$$s_i = \beta_0 + \sum_{j=1}^{P} \boldsymbol{\beta}_j \mathbf{x}_j + \sum_{j=1}^{P} \sum_{j'=j+1}^{P} \mathbf{x}_{i,j}\mathbf{x}_{i,j'} \sum_{k=1}^{K} \boldsymbol{\theta}_{j,k}\boldsymbol{\theta}_{j',k} \quad (1)$$

where k is the dimensionality of the factorization and the model parameters $\mathbf{\Theta} = \{\beta_0, \boldsymbol{\beta}, \boldsymbol{\theta}\}$ are: $\beta_0 \in \mathbb{R}$, $\boldsymbol{\beta} \in \mathbb{R}^P$ and $\boldsymbol{\theta} \in \mathbb{R}^{P \times K}$. The form of FM (Equation 1) is very general and can be used in many applications. In order to cope with different tasks, we can define an exponential family distribution over $P(y_i \,|\, s_i)$ similar to that utilized in [19], as $P(y_i \,|\, s_i) = h(y_i, \mathbf{\Phi}) \exp\Big[\eta(\lambda; s_i)^T T(y_i) - A\Big(\eta(\lambda; s_i)\Big)\Big]$ where λ is the **natural** parameter of the family. For instance:

- **Gaussian response**: For normal rating prediction problems, y_i can be treated as a real-valued response drawn from a Gaussian distribution. Thus, we scale s_i with a known variance. $\eta = s_i/\sigma^2$, $A = s_i^2/2\sigma^2 = \sigma^2\eta^2/2$ and $h = \frac{1}{\sqrt{2\pi\sigma}} \exp(-\frac{1}{2\sigma^2}y_i^2)$ where σ^2 is the variance. Thus, the expectation $\mathbb{E}[T(y)] = s$.
- **Poisson response**: For word counts, y_i can be treated as the indicator of word index, drawn from a Poisson distribution. Thus, $\eta = s_i$, $A = \exp(s_i)$ and $h = \frac{1}{y_i!}$. The expectation $\mathbb{E}[T(y)] = \exp(s) = \lambda$.
- **Binary (Bernoulli) response**: For binary decisions, y_i is usually treated as 0 or 1. Thus, $\eta = s_i$, $A = \log(1 + \exp(s_i))$. The expectation $\mathbb{E}[T(y)] = \frac{\exp(s)}{1+\exp(s)}$.

One important aspect of FM is that the model can mimic the structure of many state-of-the-art models like matrix factorization, pairwise interaction tensor factorization, SVD++ and neighborhood models in one unified framework, as demonstrated by Rendle [22].

FM can be learned in several ways, including maximum a posteriori (MAP) estimation and Bayesian inference. In this paper, we stick to MAP learning, which is to maximize the log likelihood through stochastic gradient descent (SGD) or alternating least squares. An equivalent view is to minimize a loss between the reconstructed response $f(y_i \,|\, \mathbf{x}_i, \mathbf{\Theta})$ and the true response y_i as $\arg\min_{\mathbf{\Theta}} \sum_i l(f(y_i \,|\, \mathbf{x}_i, \mathbf{\Theta}), y_i) + \mathcal{R}(\mathbf{\Theta})$ where l is a certain loss function and $\mathcal{R}$ is a regularization term for $\mathbf{\Theta}$.

3.2 Modeling User Decisions and Content

In this paper, we focus on extending FM to model Twitter data. We have two goals to achieve for understanding and modeling user behaviors in Twitter. First, we wish to uncover what kind of items with which users interact through various actions (e.g., retweets, replies and favorites) and what features contribute to the mechanism that causes certain pieces of information to be shared across social connections. Second, content is of great importance in Twitter and thus it is vital to discover topics in which users are interested and how these topics influence users' decisions. As mentioned earlier, we wish to predict users' decisions as accurately as possible while discerning topics from the huge amount of user-generated content at the same time.

For the first task, we focus on a binary response y_d for each tweet d: whether the tweet will be retweeted by a target user $u^{(t)}$. We use $u^{(a)}$ denote the author of the current tweet d. For each tweet, we use $\mathbf{x}_d$ to represent a list of explicit features for tweet d, which might include features about $u^{(t)}$ and $u^{(a)}$, and $\boldsymbol{\theta}_d$ to represent a list of latent features (factors) discovered through modeling. For the purpose of discussion, we compose a simple feature vector consisting of one categorical feature to indicate its author and one categorical feature to indicate the index of the tweet. Following the definition of FM (Equation 1), we can have an estimation of y_d based on $\mathbf{x}_d$ and $\boldsymbol{\theta}_d$ as $f(s_d \,|\, \mathbf{x}_d, \boldsymbol{\theta}_d)$:

$$s_d = \beta_0 + \boldsymbol{\beta}_u + \boldsymbol{\beta}_d + < \boldsymbol{\theta}_u, \boldsymbol{\theta}_d > \tag{2}$$

This is the exact form of traditional matrix factorization for collaborative filtering where the $\boldsymbol{\theta}$ are treated as latent factors for users and items (tweets). Since y_d is a binary response, we can also have a binary version as $y_d \sim \text{Bernoulli}\Big(\delta(s_d)\Big)$ where δ is the sigmoid function. For user decisions $\mathbf{y}$, one natural criteria for a reasonable model is to predict them as accurately as possible by specifying some error based loss functions (e.g., squared error loss). Alternatively, we can provide a ranked list of items for each user and measure how well these ranked lists approximate user actions. The latter view goes beyond pointwise evaluation and learning process for recommendation systems and has been studied extensively recently (e.g., [17, 34]). Later, we will further explore this idea.

For the second task, we will model terms in each tweet d. We denote s_{dv} to represent our estimation of the raw word count of term v in tweet d, w_{dv}, which is the response in this task. The vector $\mathbf{x}_{di}$ is used to represent features of term i in tweet d. In order to explain things more clearly, we change $\boldsymbol{\beta}$ for s_{dv} to $\boldsymbol{\alpha}$ and latent factors to $\boldsymbol{\phi}$. We use two simple indicator features here. More specifically, for each s_{dv}, we associate one categorical feature to indicate the term index and another one to indicate the tweet index. Following a similar argument, s_{dv} is a function of all features $f(s_{dv} \,|\, \mathbf{x}_{dv}, \boldsymbol{\phi}_{dv})$:

$$s_{dv} = \alpha_0 + \boldsymbol{\alpha}_d + \boldsymbol{\alpha}_v + < \boldsymbol{\phi}_v, \boldsymbol{\phi}_d > \tag{3}$$

The inner product between $\boldsymbol{\phi}_v$ and $\boldsymbol{\phi}_d$ is of interest. For $\boldsymbol{\phi}_d$, it is a K-dimensional vector and it can be treated as code for tweet d, playing a similar role as $P(z \,|\, \theta)$ in traditional topic models like probabilistic latent semantic analysis (PLSA) or latent Dirichlet allocation (LDA). The only difference is that $\boldsymbol{\phi}_d$ is not constrained to rest on the simplex. For $\boldsymbol{\phi}_v$, it is a K-dimensional vector for term v and it can be treated as a $P(z \,|\, v)$. Using $\boldsymbol{\phi}_v$ for all terms, we construct the following matrix:

$$\mathbf{M} \in \mathbb{R}^{K \times V} \quad \text{where} \;\; \mathbf{M}_{\cdot,v} = \boldsymbol{\phi}_v$$

where each column of $\mathbf{M}$ is set to $\boldsymbol{\phi}_v$. Therefore, each row of $\mathbf{M}$ can be treated as $P(v \,|\, z)$ **without** normalization. In order to recover a similar modeling power from topic models, we add the following constraints to the parameters:

$$\sum_v \boldsymbol{\phi}_{kv} = 1 \text{ for all } k, \boldsymbol{\phi}_{kv} \geq 0 \text{ ; and, } \boldsymbol{\phi}_{dk} \geq 0 \text{ ;}$$

Thus, we have a normalized matrix $\mathbf{M}$, resembling a conventional topic matrix. We can further restrict all $\boldsymbol{\alpha}$ to be non-negative, resulting in a non-negative decomposition of the term matrix. In such a setting, the content modeling behaves like a simpler topic model than its more complicated counterparts. If we view the terms in a tweet as count data, we can use the Poisson distribution and thus have $w_{di} \sim \text{Poisson}\Big(\exp(s_{di})\Big)$. On the other hand, since terms are sparse in tweets where one term will most likely appear only once in a single tweet, an alternative parametrization is to use the Bernoulli distribution $w_{di} \sim \text{Bernoulli}\Big(\delta(s_{di})\Big)$. Note that other explicit features, which are not discussed here, can be incorporated into the model easily.

Thus, we can have use FM to model two tasks separately. However, it might be better if we can jointly model these two aspects of the data together. In this paper, we propose several methods to simultaneously perform these two tasks.

4. CO-FACTORIZATION MACHINES

In many application scenarios, we may have multiple views. For instance, each tweet is associated with two types of important aspects to be modeled: 1) user action responses and 2) terms. In this subsection, we introduce Co-Factorization Machines (CoFM) to address the problem of modeling multiple aspects of data. Following the setting described in Equation 2 and Equation 3, we have two separate FMs to model two aspects of the **same** tweet where the two aspects are not linked together. Notice that we have learned two different latent representations of the same tweet: $\boldsymbol{\theta}_d$ and $\boldsymbol{\phi}_d$. Linking two factorization machines might be possible if these two latent representations of the tweet can be coupled in certain ways. Indeed, there exist several design choices to combine these two modeling processes. We present three paradigms below.

4.1 CoFM through shared features

One natural approach to link two latent representations of the **same** tweet is to treat one type of latent representation as a set of features and feed it into another modeling process. More specifically, we plug $\boldsymbol{\phi}_d$ into Equation 2 and have:

$$\begin{aligned} s_d &= \beta_0 + \boldsymbol{\beta}_u + \boldsymbol{\beta}_d + < \boldsymbol{\beta}_{\boldsymbol{\phi}_d}, \boldsymbol{\phi}_d > + < \boldsymbol{\theta}_u, \boldsymbol{\theta}_d > \\ &+ \sum_{k=1}^{K} \boldsymbol{\phi}_{d,k} < \boldsymbol{\theta}_u, \boldsymbol{\theta}_{\boldsymbol{\phi}_{d,k}} > + \sum_{k=1}^{K} \boldsymbol{\phi}_{d,k} < \boldsymbol{\theta}_d, \boldsymbol{\theta}_{\boldsymbol{\phi}_{d,k}} > \end{aligned} \tag{4}$$

Here, the third term $< \boldsymbol{\beta}_{\boldsymbol{\phi}_d}, \boldsymbol{\phi}_d >$ is a simple regression part by using latent factors obtained from content. The last two terms are of interest. Each latent factor in $\boldsymbol{\phi}_d$ is **re-weighted** by the interaction between its projection to the latent space of $\boldsymbol{\theta}$ and corresponding user/tweet latent factors. In other words, the last two terms can be re-written as:

$$\begin{aligned} &\sum_{k=1}^{K} \boldsymbol{\phi}_{d,k} < \boldsymbol{\theta}_u, \boldsymbol{\theta}_{\boldsymbol{\phi}_{d,k}} > + \sum_{k=1}^{K} \boldsymbol{\phi}_{d,k} < \boldsymbol{\theta}_d, \boldsymbol{\theta}_{\boldsymbol{\phi}_{d,k}} > = \\ &< \boldsymbol{\phi}_d, \boldsymbol{\omega}_{u,\boldsymbol{\phi}} > + < \boldsymbol{\phi}_{di}, \boldsymbol{\omega}_{d,\boldsymbol{\phi}} > \end{aligned}$$

where $\boldsymbol{\omega}$ are weights that each element is obtained from the interaction between the latent factors $\boldsymbol{\theta}$ and linear mapping vector $\boldsymbol{\theta}_{\boldsymbol{\phi}}$. This kind of mapping is similar in spirit that used by Gantner et al. [11]. The same process can be used for $w_{i,v}$ as well. If the shared feature mechanism is indeed a feature re-weighting scheme, another re-weighting approach might also be possible. Instead of using $\boldsymbol{\theta}_{\boldsymbol{\phi}}$ to map each dimension in $\boldsymbol{\phi}$ to $\boldsymbol{\theta}$, we treat $\boldsymbol{\phi}$ as the latent representation of a **missing** feature $\omega_{\boldsymbol{\phi}}$. The corresponding equation is:

$$\begin{aligned} s_d &= \beta_0 + \boldsymbol{\beta}_u + \boldsymbol{\beta}_d + \boldsymbol{\beta}_{\omega_{\boldsymbol{\phi}}} \omega_{\boldsymbol{\phi},d} + < \boldsymbol{\theta}_u, \boldsymbol{\theta}_d > \\ &+ \omega_{\boldsymbol{\phi}} < \boldsymbol{\theta}_u, \boldsymbol{\phi}_d > + \omega_{\boldsymbol{\phi}} < \boldsymbol{\theta}_d, \boldsymbol{\phi}_d > \end{aligned} \tag{5}$$

Under this formalism, $\omega_{\boldsymbol{\phi}}$ is a missing feature and can be treated as weights for interactions between $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$. Comparing Equation 5 and Equation 4, we can see that the second formalism has fewer parameters to be estimated and more intuitive motivations. In this paper, we use the second formalism.

4.2 CoFM through shared latent space

The methods introduced in the previous section assume that the latent representations obtained by different aspects of the model are different. A simpler approach is to assume that the latent factor of $\boldsymbol{\theta}_d$ is exactly the same as $\boldsymbol{\phi}_d$. Therefore, some parts of the latent factors of the same tweet are shared across different aspects. If we use $\boldsymbol{\eta}$ to indicate the shared latent factors, the two aspects under this formalism are as follows:

$$\begin{aligned} s_d &= \beta_0 + \boldsymbol{\beta}_u + \boldsymbol{\beta}_d + <\boldsymbol{\theta}_u, \boldsymbol{\eta}_d> \\ s_{d,v} &= \alpha_0 + \boldsymbol{\alpha}_d + \boldsymbol{\alpha}_v + <\boldsymbol{\phi}_v, \boldsymbol{\eta}_d> \end{aligned} \tag{6}$$

This formalism shares the idea of matrix co-factorization used in relational learning scenarios.

This approach would be convenient when multiple factors will be shared. For instance, for each term, we can add one more categorical feature to indicate the author of the tweet and therefore obtain a latent representation of its author through content modeling:

$$\begin{aligned} s_d &= \beta_0 + \boldsymbol{\beta}_u + \boldsymbol{\beta}_d + <\boldsymbol{\eta}_u, \boldsymbol{\eta}_d> \\ s_{d,v} &= \alpha_0 + \boldsymbol{\alpha}_d + \boldsymbol{\alpha}_v + \boldsymbol{\alpha}_u + <\boldsymbol{\phi}_v, \boldsymbol{\eta}_d> + <\boldsymbol{\phi}_v, \boldsymbol{\eta}_u> \\ &\quad + <\boldsymbol{\eta}_u, \boldsymbol{\eta}_d> \end{aligned}$$

where the factors for user u and tweet d are shared in two aspects.

4.3 CoFM via latent space regularization

The discussions in Sections 4.1 and 4.2 represent two variations of how to work with two latent representations of the tweet. One can regularize two such representations such that they do not reside too far away from each other. A simple approach is to impose the following regularization on the model:

$$\lambda_{\boldsymbol{\phi},\boldsymbol{\theta}} ||\boldsymbol{\phi}_d - \boldsymbol{\theta}_d||_F^2$$

where $\lambda_{\boldsymbol{\phi},\boldsymbol{\theta}}$ is a regularization parameter. Under this assumption, we can also view that one latent factor is drawn from the multivariate normal distribution with the mean as another latent factor:

$$\boldsymbol{\phi}_d \sim \mathrm{MVN}(\boldsymbol{\theta}_d, \lambda_{\boldsymbol{\phi},\boldsymbol{\theta}}^{-1}\mathbf{I})$$

This will recover the formalism from Wang and Blei [28]. An additional possibility is suggested in Agarwal and Chen [3] where a “global” representation is assumed. The “local” representation is drawn from the “global” representation by a multivariate normal distribution. Thus, a third latent representation will be introduced if this approach is used.

5. LEARNING WITH COFM

In this section we formalize the FM learning problem in an optimization framework. The discrepancy between the estimation s_i by FM/CoFM and the true value y_i can be measured by loss functions. Different choices of loss functions may lead to significant changes in performance as we will see in the experiments. Traditionally, pointwise error-based loss functions are widely used in latent factor models, which are widely used in recommender systems. Here, we discuss how different loss functions fit into the FM/CoFM framework and how the overall learning algorithm proceeds.

5.1 Optimization with content

For the task of modeling content in Twitter, two possible loss functions come from two assumptions (Poisson or Bernoulli distributions). We have the following optimization task:

$$\begin{aligned} &\text{Opt(C)} = \arg\min_{\boldsymbol{\Phi},\boldsymbol{\alpha}} \sum_{d=1}^{D}\sum_{v=1}^{V} l_C(w_{dv}, f(s_{dv} \,|\, \mathbf{x}_{dv}, \boldsymbol{\alpha}, \boldsymbol{\Phi})) \\ &+ \sum_{j=1}^{P_1} \lambda_{\boldsymbol{\alpha},j}\boldsymbol{\alpha}_j^2 + \sum_{j=1}^{P_1} \lambda_{\boldsymbol{\phi},j}||\boldsymbol{\phi}_j||_F^2 \\ &\text{s.t.:}\boldsymbol{\alpha} \geq 0,\ \boldsymbol{\phi}_{kv} \geq 0, \forall k, v;\ \boldsymbol{\phi}_k \in \mathcal{P}, \forall k \in K;\ \boldsymbol{\phi}_{dk} \geq 0, \forall d, k \end{aligned} \tag{7}$$

where P_1 is the number of features used for each term v in each tweet d and $\mathcal{P}$ is a $(K-1)$-simplex. We consider the following loss functions for this task:

- Log Poisson loss: $l^{LP}(w_{dv}, s_{dv}) = -w_{dv}\log s_{dv} + s_{dv}$. Minimizing this loss is actually equivalent to minimizing an unnormalized KL-divergence between observed counts w_{dv} and their reconstructions s_{dv}.
- Logistic loss: $l^{LG}(w_{dv}, s_{dv}) = \log[1 + \exp(-w_{dv}s_{dv})]$. Minimizing this loss is essentially performing logistic regression. Here, we only consider on/off of a term v in tweet d, dismissing its possible multiple occurrences.

The optimization problem in Equation 7 can be efficiently solved according to two facts: 1) Due to the property of *multilinearilty* [22], the model is linear with respect to each model parameter when others are fixed and, 2) Proposition 1 in [37] states that the optimal value of a single parameter when other are fixed is the maximum between zero and the value obtained by a non-constrained version of the same problem. Also, efficient methods (e.g., [10]) exist to project real-valued vectors onto the simplex. Therefore, this optimization problem is solvable.

5.2 Optimization with user decisions

For modeling user decisions, we also formalize the problem as an optimization problem as follows:

$$\begin{aligned} \text{Opt(U)} &= \arg\min_{\boldsymbol{\Theta},\boldsymbol{\beta}} \sum_{d=1}^{D} l_U(y_d, f(s_d \,|\, \mathbf{x}_d, \boldsymbol{\beta}, \boldsymbol{\Theta})) \\ &+ \sum_{j=1}^{P_2} \lambda_{\boldsymbol{\beta},j}\boldsymbol{\beta}_j^2 + \sum_{j=1}^{P_2} \lambda_{\boldsymbol{\theta},j}||\boldsymbol{\theta}_j||_F^2 \end{aligned} \tag{8}$$

where P_2 is the number of features used for each tweet d. Unlike content modeling, no constraints are put onto the parameter space. For pointwise loss functions, we consider:

- Squared error loss: $l^S(y_d, s_d) = (y_d - s_d)^2$, which is for regression problems with Gaussian responses.
- Logistic loss: l^{LG}, the same as the loss used in modeling content.
- Huber loss:

  $$l^H(y_d, s_d) = \begin{cases} \frac{1}{2}\max(0, 1 - y_d s_d)^2 & \text{if } y_d s_d > 0 \\ \frac{1}{2} - y_d s_d & \text{otherwise} \end{cases}$$

  This is the one-sided variant of Huber’s robust loss function. It is convex and continuously differentiable. The loss is mentioned in Yang et al. [33].

It has been demonstrated that pointwise loss functions may not be appropriate for recommender tasks when users choose items from a list of items prepared by the system [8]. In such cases, we wish to adopt more advanced loss functions which consider pairwise preferences. For each target user u, we can construct a set of tweets which are originated by other users and retweeted by u, denoted as $\mathcal{C}_u^+$. Note that these tweets could be originated by u’s friends or any other users who are not connected to u. On the contrary, we denote all other tweets from u’s friends which are not retweeted by u as $\mathcal{C}_u^-$. Therefore, for each user u, there exists a huge set of tweets which are outside of u’s network and are treated as unknown and not considered in the following loss functions. In this work, we focus on pairwise loss functions:

Margin ranking criterion (MRC):

$$l^M(x_1, x_2) = \sum_{x_1 \in \mathcal{C}_u^+}\sum_{x_2 \in \mathcal{C}_u^-} \max[0, 1 - f(x_1) + f(x_2)]$$

which considers all pairs of positive and negative labels, and assigns each a cost if the negative label is larger or within a "margin" of 1 from the positive label. Optimizing this loss is similar to optimizing the area under the curve of the receiver operating characteristic curve. That is, all pairwise violations are considered equally if they have the same margin violation, independent of their position in the list. For this reason the margin ranking loss might not optimize precision at k very accurately. This loss function is proposed in Herbrich et al. [12] and used in many IR tasks (e.g., [15, 4]).

Bayesian personalized ranking (`BPR`):

$$l^B(x_1, x_2) = \sum_{x_1 \in \mathcal{C}_u^+} \sum_{x_2 \in \mathcal{C}_u^-} -\log[\delta(f(x_1) - f(x_2))]$$

where δ is a sigmoid function. This loss is proposed in Rendle et al. [23] and has been used in tag recommendation (e.g., [24]) and yielded superior performance. This can be viewed as a smooth version of `MRC`.

Weighted Approximately Ranked Pairwise loss (`WARP`): This loss, proposed in Usunier et al. [26], has been successfully applied in image retrieval tasks [31] and IR tasks [32]. Here, we discuss its application in recommender systems. The idea of `WARP` is to focus more on the top of the ranked list where the top k positions are those we care about, comparing to `MRC` and `BPR` where no notion of ranked list is introduced. By using the precision at k measure, one can weigh the pairwise violations depending on their position in the ranked list. `WARP` is defined as an error function as follows:

$$\text{err}_{\text{WARP}} = \sum_{\mathbf{x}_i \in \mathcal{C}_u^+} L[rank(f(\mathbf{x}_i))] \tag{9}$$

where $rank(f(\mathbf{x}_i))$ is the rank of a positive labeled instance $\mathbf{x}_i \in \mathcal{C}_u^+$ given by $rank(f(\mathbf{x}_i)) = \sum_{\mathbf{x}' \in \mathcal{C}_u^-} \mathbb{I}[f(\mathbf{x}') \geq f(\mathbf{x}_i)]$ where $\mathbb{I}(x)$ is the indicator function, and $L(\cdot)$ transforms this rank into a loss: $L(r) = \sum_{j=1}^{r} \tau_j$, with $\tau_1 \geq \tau_2 \geq \cdots \geq 0$. The idea of the $rank$ function is to compute the violations where negative instances are ranked higher than the positive ones and the L function is to transform violations into a loss. Different choices of τ define different importance of the relative position of the positive examples in the ranked list. In particular:

- For $\tau_i = 1$ for all i we have the same AUC optimization as margin ranking criterion.
- For $\tau_1 = 1$ and $\tau_{i>1} = 0$ the precision at 1 is optimized.
- For $\tau_{i \leq k} = 1$ and $\tau_{i>k} = 0$ the precision at k is optimized.
- For $\tau_i = 1/i$ a smooth weighting over positions is given, where most weight is given to the top position, with rapidly decaying weight for lower positions. This is useful when one wants to optimize precision at k for a variety of different values of k at once.

In this work, we use $\tau_i = 1/i$. It is difficult to directly optimize `WARP` due to the discrete nature of indicator functions. In addition, since the number of negative instances is significantly larger than positive instances, the $rank$ function is inefficient to be calculated. Before we tackle these issues, the form of Equation 9 can be readily re-written as (see [31] for details):

$$\text{err}_{\text{WARP}} = \sum_{\mathbf{x}_i \in \mathcal{C}_u^+} \frac{L[rank(f(\mathbf{x}_i))] \sum_{\mathbf{x}' \in \mathcal{C}_u^-} \mathbb{I}[f(\mathbf{x}') \geq f(\mathbf{x}_i)]}{rank(f(\mathbf{x}_i))}$$

with the convention $0/0 = 0$ when the correct label y is top-ranked. We replace the indicator function by using the margin function $\max(0, 1 - f(\mathbf{x}_i) + f(\mathbf{x}'))$. In order to approximate the $rank$ function, for a given positive instance, one draws negative instances until the one which violates the indicator function. Thus, we approximate $rank(f(\mathbf{x}_i))$ by using $\left\lfloor \frac{D^- - 1}{N} \right\rfloor$ where $\lfloor \cdot \rfloor$ is the floor function, D^- is the number of items in $\mathcal{C}_u^-$ and N is the number of trials of sampling until a violating pair is found. The approximation only requires local knowledge of negative instances, making it easily to be calculated per user for our case.

Competitive softmax loss (`SOFTMAX`):

$$P(y_i = 1 \,|\, \mathcal{C}_u) = \frac{\exp(f(s_i \,|\, \mathbf{x}_i))}{\sum_{\mathbf{x}_i} \exp(f(s_i \,|\, \mathbf{x}_i))} \text{ for all } \mathbf{x}_i \in \mathcal{C}_u \tag{10}$$

This loss is introduced by Yang et al. [34] and is motivated by the idea that users are presented a list of items and they choose items based on the "utility" they will receive if the item is chosen. Here, we assume that the utility for item i consists of two components $s_i + e_i$ where s_i encodes the intrinsic interest of the item to the target user and e_i is a stochastic error term reflecting the uncertainty and complexity of the choice process. We choose s_i to be the outcome from the predictive model (e.g., `FM`, `CoFM`). If the error term e_i is independently and identically distributed as a Weibull distribution, the probability item i is chosen is exactly as Equation 10, which is essentially a multinomial logic model.

Competitive hinge loss (`HINGE`): Following a similar assumption that a user chooses items based on their utilities, we can formalize the problem of distinguishing positive instances from negative ones as a problem of classification. Therefore, the key idea of `HINGE` loss is that the utility difference between a positive item and negative items would be greater than **random errors**, namely:

$$\mathbb{I}(y_i == 1) f(s_i \,|\, \mathbf{x}_i) > \frac{1}{|\mathcal{C}_u^-|} \sum_{\mathbf{x}_i \in \mathcal{C}_u^-} \mathbb{I}(y_i == 0) f(s_i \,|\, \mathbf{x}_i)$$

With this spirit, a pairwise preference learning problem can be formalized as follows:

$$l^H = \min \sum_{t=1}^{|\mathcal{C}_u^+|} \xi_t$$

$$\text{s.t.: } f(s_t \,|\, \mathbf{x}_t) - \frac{1}{|\mathcal{C}_u^-|} \sum_{\mathbf{x}_i \in \mathcal{C}_u^-} f(s_i \,|\, \mathbf{x}_i) \geq 1 - \xi_t \text{ and } \xi_t \geq 0,$$

$$\forall \mathbf{x}_t \in \mathcal{C}_u^+$$

where ξ are introduced parameters to be optimized. This loss reflects the insight that user decisions are usually made by comparing alternatives and considering the differences in potential utilities. In other words, the marginal utility between user choice and the average of non-choices is maximized. This loss is also introduced in Yang et al. [34].

All of these ranking-based loss functions were proposed in different contexts and never compared in recommender systems. From the discussion above, it is clear that `WARP` is the only loss function considering the relative positions between positive instances. Meanwhile, both `SOFTMAX` and `HINGE` consider the set of positive and negative instances as a whole, rather than `MRC` and `BPR` only deal with pairwise preferences. In addition to what has been discussed here, other ranking based loss measures are also proposed. For instance, Koren and Sill proposed the OrdRec approach [18], which is based on a pointwise ordinal model. The idea of OrdRec is to predict a full probability distribution of the expected item ratings, rather than only a single score for an item. However, this is not desirable in our setting in that we only care about relevant items ranked at the top. Thus, it is not necessary (and even impossible in practice) to predict a full distribution over

Algorithm 1: The sketch of the algorithm to optimize Equation 11. This is one iteration over the whole dataset.

```
for $u = 1$ to $|U|$ do
    Optimize Opt(U) for $\theta_d$, $\theta_u$ and $\beta$:
        Perform stochastic gradient descent for pointwise loss
        functions or rank-based loss functions.
    Optimize Opt(C) for $\phi_d$ and $\alpha$:
        Perform stochastic gradient descent for log-Poisson
        loss or logistic loss
Optimize Opt(C) for $\mathbf{M}$
    Perform gradient descent to obtain the current optimal
    value for the topic matrix
```

all positions. Other researchers have tried to optimize NDCG directly. However, this is either done by approximation [29] or by a two-stage approach [5], which might be sub-optimal.

5.3 Summary

Putting things together, a CoFM framework for learning a model for user decisions and content understanding can be formalized as:

$$\text{Opt(CoFM)}: \ \text{Opt(U)} + \pi\text{Opt(C)} \tag{11}$$

where π is a parameter to balance two objective functions. By choosing different coupling strategies introduced in Section 4 and different loss functions, Opt(CoFM) can effectively perform predictive modeling and maximum likelihood estimation of content at the same time. We adopt a hybrid of SGD and coordinate descent to optimize Equation 11, which is sketched in Algorithm 1. We iterate the whole dataset by performing SGD for predictive modeling and content modeling while fixing the topic matrix. After one iteration, we optimize the topic matrix by gradient descent. Since we restrict $\mathbf{M} \in \mathcal{P}$, an efficient method [10] can achieve this task.

6. FEATURES

Here we discuss the features used in our models. These features utilize the content of tweets that users have generated. All of these features try to capture users' interests. Features are divided into five groups: 1) categorical features, 2) content profiles, 3) relevance scores, 4) latent topic model features, and 5) content meta features, where each group has multiple features.

Categorical Features: The key idea of FM/CoFM is to use both indicator features and explicit features together to obtain competitive performance in predictive tasks. For modeling user decisions, we use three categorical features: 1) target user id, 2) neighbor user id and 3) the tweet id. For content modeling, we use the term id and the tweet id as features.

Content Features: For "content profiles" we utilize features to characterize what users have posted and what their friends have posted. For each tweet i, let $\boldsymbol{w}_i$ be the term vector for this tweet. Let $u(i)$ be the author of the tweet i. For user u, we construct three user profiles as follows:

- **Content Profile**: Let $\text{CP}(u) = \sum_{i=1}^{D} \mathbb{I}[u(i) == u]\mathbf{w}_i$. This is essentially the concatenation of all term vectors generated by user u.
- **Neighborhood Profile**: Let $\text{NP}(u) = \sum_{u' \in N(u)} C(u')$ where $N(u)$ is a set of friend users of user u.
- **Retweet Profile**: Let $\text{RP}(u) = \sum_{i=1}^{D} \mathbb{I}[u(i) == u \wedge r(i) == 1]\mathbf{w}_i$ where $r(i)$ is a binary indicator for whether tweet i is a retweet or not.

These profile features will capture the interests of users at a fine-grained level. One drawback of these features is that they capture the long-term interests of users. For new tweets, they remain the same and would be less discriminative. Thus, we introduce the second group of features, characterizing how relevant a new tweet is against user profiles. Let $\mathcal{R}(\mathbf{w}_1, \mathbf{w}_2)$ be a relevance measurement between term vector $\mathbf{w}_1$ and $\mathbf{w}_2$. Thus, we have the following relevance scores:

- **Content Relevance**: $\mathcal{R}(\mathbf{w}_i, \text{CP}(u))$, measuring the similarity of the incoming tweet to the user's content profile.
- **Neighborhood Relevance**: $\mathcal{R}(\mathbf{w}_i, \text{NP}(u))$, measuring the similarity of the incoming tweet to the user's neighborhood profile.
- **Retweet Relevance**: $\mathcal{R}(\mathbf{w}_i, \text{RP}(u))$, measuring the similarity of the incoming tweet to the user's retweet profile.

We use dot product as the relevance measure although many other IR relevance scores could also apply. Both "content profiles" and "relevance scores" utilize term level information to determine the features. In addition to these features directly related to the content generated by the users, other meta information also might be useful:

- **Length of Tweet**: Number of characters used in tweet i.
- **Hash Tag Count**: Number of hash tags used in tweet i.
- **Hash Tag History**: How many times the hash tag appears in $u(i)$'s retweets.
- **URL Count**: The number of URLs used in tweet i.
- **URL Domain History**: How many times the URL domain appears in $u(i)$'s retweets.
- **Retweet Count**: The number of times tweet i has been retweeted so far.

Local Graph & User Features: These features potentially characterize how popular and how well connected a user is. Intuitively, a popular user who has many friends and followers can be actively passing information by retweeting messages.

- **Mention Count**: The number of times user u is mentioned in other tweets.
- **Friend Count**: The number of friends user u follows.
- **Follower Count**: The number of followers user u has.
- **Status Count**: The number of tweets user u has published.
- **Account Age**: Number of years user u appeared on Twitter.

User Relationship Features: Relation features refer to those features which represent the relationship between a target user u_i and his/her friend u_j.

- **Co-Friends Score**: This feature estimates the similarity of friend sets of the target user u_i and his/her friend u_j.
- **Co-Follow Score**: This feature estimates the similarity of follower sets of the target user u_i and his/her friend u_j.
- **Mention Score**: The number of times u_i mentions u_j.
- **Retweet Score**: The number of times u_i retweets u_j.
- **Mutual Friend**: Whether u_i and u_j are mutual friends.

The similarity measure used is the Jaccard coefficient.

Temporal Features: We estimate user u's activity level at time t as $h_u(t)$, which is calculated by the average number of tweets he/she published in a periodical time slot, e.g., every Monday. With the estimated response time Δt, the number of accumulated tweets can be written as: $r_u(\Delta t) = \sum_{j \in F(u)} \int_{t^w}^{t^w + \Delta t} h_j(t)\, dt$, as proposed by Peng et al. [21]. We calculate both activities using period of a day and a week.

In our case, all features are pre-calculated through a Hadoop cluster and can be processed efficiently.

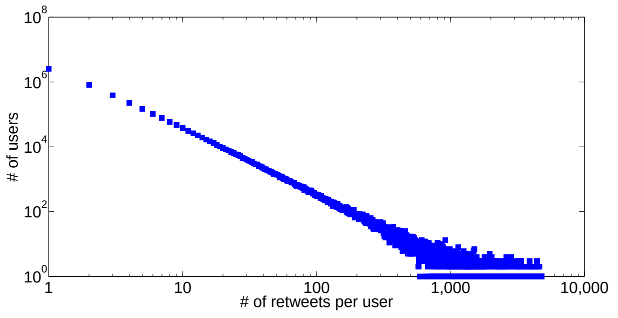

Figure 1: The sparsity of retweets per user.

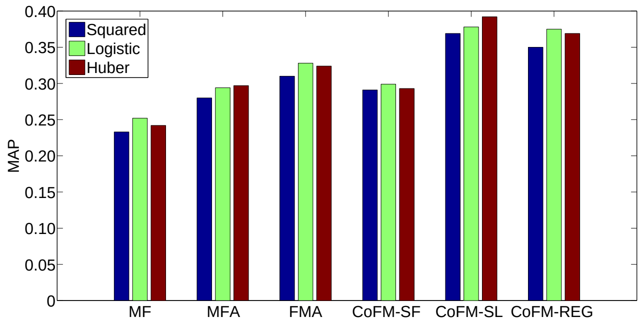

Figure 2: The comparison of pointwise loss functions.

7. EXPERIMENTS

To prepare our dataset, we first monitored the Twitter public stream for one month in June 2012 and extracted users who post at least ten tweets including at least one retweet during this time period, resulting in 765,386 target users with approximately 11M tweets. For all these target users, we 1) crawled all their historical posts and 2) traced who they retweeted from and crawled all their posts as well. In this fashion, we obtained 4,327,816 neighbor users with 27M tweets, resulting in a dataset that is significantly larger than any previous work for similar tasks. For each target user u, we treat all tweets from his/her neighbor users as incoming tweets. If a tweet d from incoming tweets are retweeted by user u, d is treated as a positive instance, and negative otherwise. We plot the unnormalized distribution of number of retweets per user in Figure 1, demonstrating that a great number of users only retweet a limited number of times while some users retweet thousands of times.

We adopt rank-based metrics to evaluate the effectiveness of different models. We borrow Mean Average Precision (MAP) from the IR community. We define "precision" at position k (Precision@k) of all incoming items for a particular user as $(\text{\# of retweets in top positions})/k$. Then an average measure across all top m positions (Average Precision) for user u is defined as $(\sum_{k=1}^{m} \text{Precision@k} \times l_k)/(\text{\# of retweets for ranked list of user } u)$ where l_k is a binary indicator whether the position k has been retweeted or not and m is the total number of positions evaluated. Note that Average Precision is evaluated per user. We can average it across all users, resulting in the MAP measure, as used similarly in [6, 13].

In order to mimic a realistic evaluation environment, we adopt a time-based evaluation process, significantly differing from Chen et al. [6] where a **fixed** ratio of training vs. testing dataset is used. (It is not clear whether the ratio is kept according to the time order.) In addition, we do not use cross validation as it violates the time order of data, yielding unfair advantages to certain models. Here, for each user, we split all incoming tweets into n consecutive time periods with **equal** number of tweets in each time period. We train models on one time period and test them on the next. This is a balance between cross validation and online training and testing. In our experiments, we set $n = 5$.

We compare several aspects of proposed methods and other state-of-the-art baselines (some details omitted for space):

- **Matrix factorization** (MF): Categorical features, the target user id and the tweet id are used to feed into FM, yielding a MF model with biases, which is a solid baseline in many collaborative filtering tasks.
- **Matrix factorization with attributes** (MFA): In addition to the categorical features used in MF, we add explicit features into the model, which essentially mimic the state-of-the-art latent factor models with features [1, 33, 7] as mentioned by Rendle [22].
- **Collaborative personalized tweet recommendation** (CPTR): This is the model proposed in [6], which is a variation of MFA where the item factors for a tweet are decomposed into term factors and neighboring user factors. This is a state-of-the-art method for the task. We re-implement their approach under the framework of FM.
- **Factorization machines with attributes** (FMA): In contrast to two categorical features, we add one more categorical feature, the neighbor user id, into the model, resulting in a pairwise tensor factorization model with "target user-item-neighbor user" interactions.
- CoFM **with shared features** (CoFM-SF): This is the model introduced in Section 4.1 with the same indicator features as FMA for user decisions. We use term id and tweet id as two categorical features for content modeling. The latent factors for the tweet are shared through the tweet id. Thus, we have a pairwise tensor interaction model for user decisions and a topic coding model for content.
- CoFM **with shared latent spaces** (CoFM-SL): This is the model introduced in Section 4.2. The feature setting is the same as CoFM-SF.
- CoFM **with latent space regularizations** (CoFM-REG): This is the model introduced in Section 4.3. The feature setting is the same as CoFM-SF.

For the sake of simplicity, we fix all regularization parameters to 1 and tune π, the balance parameter of predictive modeling and content modeling in Equation 11. We report the performance of $\pi = 0.3$, which is the best in our experiments. We also tune K, the dimensionality of latent factors in FM/CoFM, from 10 to 250. We report the performance of $K = 150$, which is the best in the experiments. For CPTR, we fix $K = 200$, which is used in [6]. For content modeling, we do not observe significant differences between using log-Poisson loss and logistic loss. Thus, for generality, we report the results based on log-Poisson loss.

Predictive Results: We compare the predictive power of different models. First, we demonstrate how loss functions affect performance, starting from pointwise loss functions. For MF, MFA and FMA, we compare using three pointwise loss functions: squared error loss, logistic loss and Huber loss. For CoFM-SF, CoFM-SL and CoFM-REG, we use the same three loss functions for predictive modeling while fixing log-Poisson loss for content modeling. Since CPTR is fixed to pairwise learning, we exclude it from the experiment. The result is shown in Figure 2. The first observation is that the performance of MF which only uses user-item interactions

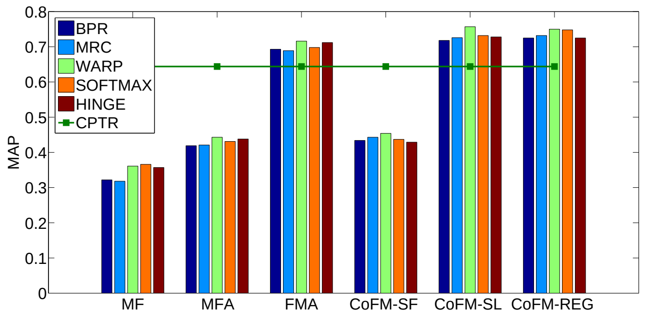

Figure 3: The comparison of ranking-based loss functions.

Table 1: Examples of topics are shown. The terms are top ranked terms in each topic. The topic names shown in bold are given by the authors.

Entertainment
album music lady artist video listen itunes apple produced movies #bieber bieber new songs
Finance
percent billion bank financial debt banks euro crisis rates greece bailout spain economy
Politics
party election budget tax president million obama money pay bill federal increase cuts

is significantly worse than the ones utilizing explicit features. The second observation is that logistic loss and Huber loss is consistently better than squared error loss. This might be due to the reason that we only have binary responses (retweets), similar to what is reported in Yang et al. [33]. The third observation is that CoFM−SL and CoFM−REG are noticeably better than all other methods. This validates our discussion in Section 4 that these two paradigms can be viewed as variants of many successful co-factorization models where the predictive aspect can benefit from the content modeling aspect. On the other hand, CoFM−SF performs poorly and even cannot match the performance of FMA. We conjecture that this is because the data is too sparse such that additional parameters induced by CoFM−SF cannot be effectively learned. We also observe that FMA performs better than MFA, indicating that ternary interactions "target user-item-neighbor user" can indeed capture the dynamics between users on Twitter, compared to "target user-item" binary interactions.

In addition to pointwise loss functions, we also compare performance of different models with ranking-based loss functions, as shown in Figure 3. The green line in the bar chart is the performance of CPTR since it is trained with BPR. Comparing the results to pointwise loss functions, the overall performance is significant higher, indicating that ranking-based loss functions are indeed better for the task. Also, the discrepancy between different models becomes larger, compared to pointwise loss functions. For instance, FMA, CoFM−SL and CoFM−REG are much better than the others, where all three are above CPTR significantly. In addition, CoFM−SL and CoFM−REG are consistently $3\% - 4\%$ better (depending on the specific ranking-based loss function) than FMA in absolute MAP scores across 5 split of data. Overall, WARP achieves competitive performance consistently for all models.

Content Modeling: We explore how topics are learned through the modeling. From the formalism in Section 5.1, the matrix $\mathbf{M}$

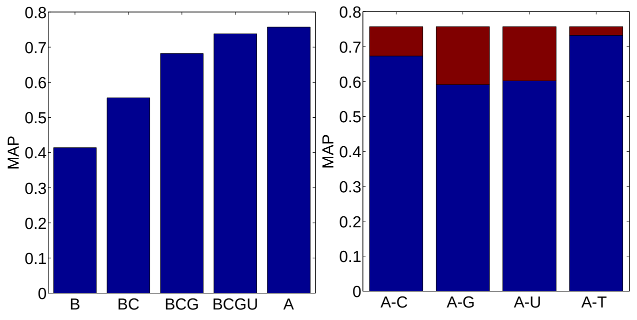

Figure 4: The impact of different groups of features. The effect of "add on" is shown on the left and the effect of "take out" is on the right. For both figures, "A", "B", "C", "G", "U" and "T" stand for "All", "Base model", "Content feature", "Graph feature", "User feature" and "Temporal feature" respectively.

can be interpreted as a topic matrix as in standard PLSA/LDA. Thus, we can describe topics as in other topic models by ranking terms in probabilities. This is superior to CPTR [6] where term factors are not in the simplex. We show some example topics in Table 1. We can see that these topics are easily recognized and have the benefit of normal topic models while we do not have cumbersome Bayesian style formalism and expensive inference algorithms in the model. Note, however, that content modeling is not only for explanatory analysis—it is indeed helpful for prediction tasks. From Figures 2 and 3, we can see that CoFM which utilizes content modeling has better performance in general, and especially for CoFM−SL and CoFM−REG which can outperform state-of-the-art methods significantly.

Feature Analysis: We study how different types of features contribute to the predictive power of the model. Instead of using methods like χ^2 to calculate the correlation between feature values with respect to classification labels, we adopt the following two straightforward methods. First, we start from a base model CoFM−SL using WARP without any explicit features, which is the best model from previous experiments, and then add one group of features consecutively. This method, denoted as "add on", shows the contribution of each group of features as it is added into the model. The second method, denoted as "take out", starts with a complete model and removes one group of features to see how performance drops accordingly. The results for "add on" and "take out" are shown in Figure 4. For "add on", it is clear that each group of features contributes to the final performance of the model and "Temporal" features have the least marginal gain. The most gains come from "Content" features and "Local Graph" features. This observation is consistent with [14]. For "take out", the red square in each bar in the figure represents the performance deduction for the corresponding feature group. Again, "Temporal" features have the least impact on performance while removal of "Local Graph" features hurts performance much more than that of "Content" features. From both "add on" and "take out", it seems that "Local Graph" plays an important role in the performance, followed by "Content" features. This suggests that social connections are important in determining retweets as well as content factors.

8. CONCLUSION

Users of social media services are often simultaneously overwhelmed with the amount of information delivered via their social connections and miss out on content that they might have liked to see. Both issues serve as difficulties to the users and drawbacks to the services. These services can benefit from understanding user interests and how they interact with the service, potentially predicting their behaviors in the future. We propose Co-Factorization Machines (`CoFM`) to address the problem of simultaneously predicting user decisions and modeling content in social media by analyzing rich information gathered from Twitter. The task differs from conventional recommender systems as the cold-start problem is ubiquitous, and rich features, including textual content, need to be considered. Additionally, we discuss and compare ranking-based loss functions in the context of recommender systems, shedding light on how they vary from each other and perform in real tasks, providing the first work in this direction. We explore a large number of features and conduct experiments on a real-world dataset, concluding that `CoFM` with ranking-based loss functions is superior to state-of-the-art methods and yields interpretable latent factors.

Acknowledgements

This material is based upon work supported in part by the National Science Foundation under Grant Number IIS-0803605.

9. REFERENCES

[1] D. Agarwal and B.-C. Chen. Regression-based latent factor models. In *KDD*, pages 19–28, 2009.

[2] D. Agarwal and B.-C. Chen. fLDA: matrix factorization through latent dirichlet allocation. In *WSDM*, pages 91–100, 2010.

[3] D. Agarwal, B.-C. Chen, and B. Long. Localized factor models for multi-context recommendation. In *KDD*, pages 609–617, 2011.

[4] B. Bai, J. Weston, D. Grangier, R. Collobert, K. Sadamasa, Y. Qi, C. Cortes, and M. Mohri. Polynomial semantic indexing. In *NIPS*, pages 64–72, 2009.

[5] S. Balakrishnan and S. Chopra. Collaborative ranking. In *WSDM*, pages 143–152, 2012.

[6] K. Chen, T. Chen, G. Zheng, O. Jin, E. Yao, and Y. Yu. Collaborative persoanlized tweet recommendation. In *SIGIR*, 2012.

[7] T. Chen, Z. Zheng, Q. Lu, W. Zhang, and Y. Yu. Feature-based matrix factorization. Technical report, Apex Data & Knowledge Management Lab, Shanghai Jiao Tong University, 2011.

[8] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *RecSys*, pages 39–46, 2010.

[9] Y. Duan, L. Jiang, T. Qin, M. Zhou, and H.-Y. Shum. An empirical study on learning to rank of tweets. In *COLING*, pages 295–303, 2010.

[10] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the l1-ball for learning in high dimensions. In *ICML*, pages 272–279, 2008.

[11] Z. Gantner, L. Drumond, C. Freudenthaler, S. Rendle, and L. Schmidt-Thieme. Learning attribute-to-feature mappings for cold-start recommendations. In *ICDM*, pages 176–185, 2010.

[12] R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. In *Advances in Large Margin Classifiers*, pages 115–132. MIT Press, 2000.

[13] L. Hong, R. Bekkerman, J. Adler, and B. D. Davison. Learning to rank social update streams. In *SIGIR*, pages 651–660, 2012.

[14] L. Hong, O. Dan, and B. D. Davison. Predicting popular messages in twitter. In *WWW Companion*, pages 57–58, 2011.

[15] T. Joachims. Optimizing search engines using clickthrough data. In *KDD*, pages 133–142, 2002.

[16] Y. Kim and K. Shim. Twitobi: A recommendation system for twitter using probabilistic modeling. In *ICDM*, pages 340 –349, 2011.

[17] Y. Koren. Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 4(1):1:1–1:24, Jan. 2010.

[18] Y. Koren and J. Sill. OrdRec: an ordinal model for predicting personalized item rating distributions. In *RecSys*, pages 117–124, 2011.

[19] H. Lee, R. Raina, A. Teichman, and A. Y. Ng. Exponential family sparse coding with applications to self-taught learning. In *IJCAI*, pages 1113–1119, 2009.

[20] C. Lin, Q. Mei, J. Han, Y. Jiang, and M. Danilevsky. The joint inference of topic diffusion and evolution in social communities. In *ICDM*, pages 378 –387, dec. 2011.

[21] H.-K. Peng, J. Zhu, D. Piao, R. Yan, and Y. Zhang. Retweet modeling using conditional random fields. In *ICDM Workshops*, pages 336–343, 2011.

[22] S. Rendle. Factorization machines with libFM. *ACM TIST*, 3(3):57:1–57:22, May 2012.

[23] S. Rendle, C. Freudenthaler, Z. Gantner, and S.-T. Lars. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*, pages 452–461, 2009.

[24] S. Rendle and L. Schmidt-Thieme. Pairwise interaction tensor factorization for personalized tag recommendation. In *WSDM*, pages 81–90, 2010.

[25] H. Shan and A. Banerjee. Generalized probabilistic matrix factorizations for collaborative filtering. In *ICDM*, pages 1025–1030, 2010.

[26] N. Usunier, D. Buffoni, and P. Gallinari. Ranking with ordered weighted pairwise classification. In *ICML*, pages 1057–1064, 2009.

[27] I. Uysal and W. B. Croft. User oriented tweet ranking: a filtering approach to microblogs. In *CIKM*, pages 2261–2264, 2011.

[28] C. Wang and D. M. Blei. Collaborative topic modeling for recommending scientific articles. In *KDD*, pages 448–456, 2011.

[29] M. Weimer, A. Karatzoglou, Q. V. Le, and A. Smola. CofiRank - maximum margin matrix factorization for collaborative ranking. In *NIPS*, pages 1593–1600, 2007.

[30] M. Weimer, A. Karatzoglou, and A. Smola. Adaptive collaborative filtering. In *RecSys*, pages 275–282, 2008.

[31] J. Weston, S. Bengio, and N. Usunier. Large scale image annotation: learning to rank with joint word-image embeddings. *Machine Learning*, 81(1):21–35, Oct. 2010.

[32] J. Weston, C. Wang, R. Weiss, and A. Berenzweig. Latent collaborative retrieval. In *ICML*, 2012.

[33] S.-H. Yang, B. Long, A. Smola, N. Sadagopan, Z. Zheng, and H. Zha. Like like alike: joint friendship and interest propagation in social networks. In *WWW*, pages 537–546, 2011.

[34] S.-H. Yang, B. Long, A. J. Smola, H. Zha, and Z. Zheng. Collaborative competitive filtering: learning recommender using context of user choice. In *SIGIR*, pages 295–304, 2011.

[35] Z. Yang, J. Guo, K. Cai, J. Tang, J. Li, L. Zhang, and Z. Su. Understanding retweeting behaviors in social networks. In *CIKM*, pages 1633–1636, 2010.

[36] L. Zhang, D. Agarwal, and B.-C. Chen. Generalizing matrix factorization through flexible regression priors. In *RecSys*, pages 13–20, 2011.

[37] J. Zhu and E. P. Xing. Sparse topical coding. In *UAI*, pages 831–838, 2011.