GPOP: Scalable Group-level Popularity Prediction for Online Content in Social Networks

Minh X. Hoang†, Xuan-Hong Dang†, Xiang Wu‡, Zhenyu Yan‡, Ambuj K. Singh†
† University of California, Santa Barbara; ‡ Adobe Systems
†{mhoang, xdang, ambuj}@cs.ucsb.edu; ‡ {xianwu, wyan}@adobe.com

ABSTRACT

Predicting the popularity of online content in social networks is important in many applications, ranging from ad campaign designing, web content caching and prefetching, to web-search result ranking. Earlier studies target this problem by learning models that either generalize behaviors of the entire network population or capture behaviors of each individual user. In this paper, we claim that a novel approach based on group-level popularity is necessary and more practical, given the observation that users naturally organize themselves into clusters and that users within a cluster react to online content in a uniform manner. We develop a novel framework by first grouping users into cohesive clusters, and then adopt tensor decomposition to make predictions. In order to minimize the impact of noisy data and be more flexible in capturing changes in users' interests, our framework exploits both the network topology and interaction among users in learning a robust user clustering. The PARAFAC tensor decomposition is adapted to work with hierarchical constraint over user groups, and we show that optimizing this constrained function via gradient descent achieves faster convergence and leads to more stable solutions. Extensive experimental results over two social networks demonstrate that our framework is scalable, finds meaningful user groups, and significantly outperforms eight baseline methods in terms of prediction accuracy.

Keywords

Content prediction; Graph clustering; Tensor decomposition

1. INTRODUCTION

With the enormous amount of data generated on the Internet today, predicting the popularity of online content, such as videos, news articles, or posts on social networks, is increasingly important in many different applications. In particular, *given the set of users who have reacted (i.e., commented, liked, shared or retweeted) to a content from time t_0 (when it was created) to time t_1, can we predict how many and which users will react to it until time $t_2 > t_1$?* If this question can be efficiently answered, we can filter information for users to cope with data overload, or prefetch web content to reduce latency and improve user experience. We can also design more effective ad campaigns to increase product popularity and maximize profit.

DOI: 10.1145/1235

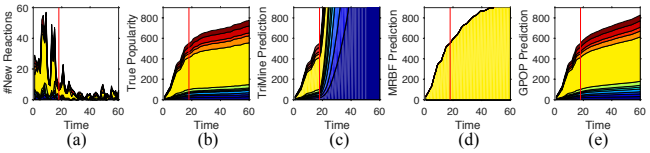


Figure 1: An example project from Behance.net where each color stripe represents one user group: (a) Number of new reactions per 4-hour time periods; (b) Cumulative number of reactions; Predictions of popularity made at time $t_1 = 18$ (marked by the vertical red line) for (c) user level [23] (then aggregated by groups), (d) population level [27], and (e) group level (our solution).

Several studies have been done to predict the popularity of various online network-contents, and they can be generally grouped into two categories: (i) *user-level popularity* [22, 23, 36–38] that predicts (at a low level) which users will react to a content; (ii) *population-level popularity* [10, 11, 20, 27, 29] that predicts (at a high level) how many users in total will react to a content. While each approach is reasonable to use in certain situations, we claim that a *group-level popularity* approach, which predicts the popularity within user groups, is more practical given the noise and the intrinsic heterogeneity in the network data. The user-level information cascades, on one hand, are often susceptible to missing data, sensitive to users' emotions, and also often costly to learn [7]. The population-level popularity, on the other hand, is only able to provide a very coarse view, losing most essential information on user behaviors, and thus lacks flexibility in tailoring information for different users' interests. We observe from many social networks (specifically Twitter.com and Behance.net in this paper) that users naturally organize themselves into groups, reflecting their interests, communities or locations. Within a group, users are fairly consistent in how they react to content. Thus, predicting contents at the group-level popularity essentially manifests this natural fact and further provides a great trade-off between the cost in model learning and prediction quality. Compared to the user-level, a group-level popularity is much less noisy and more compact, while it is more detailed and cohesive than that at the population level. In addition, a group-level incurs a significantly smaller computational cost than the user-level predictions. Thus, for many applications like

macro-level ad campaign designing, or web content caching and prefetching, a group-level popularity prediction is more practical and beneficial than the other two alternatives.

Example 1. Behance.net is a social network where users share their creative projects for others to see. The popularity of a project at a timestamp t can be defined as the total number of users who have pressed the "appreciate" button on this project. In Fig. 1a, we show the number of new reactions (appreciations) at each timestamp (a period of 4 hours each) for one project, while Fig. 1b shows its cummulative form. Each color stripe corresponds to one cohesive group of users (based on our later proposed solution). It can be seen that most users who appreciated this project are from one group corresponding to the yellow stripe in Fig. 1a-b, suggesting that the interests and behaviors of users are similar within each group and different across different groups. Finally, given the observation over this project from time 0 to 18 (first 3 days), we predict its popularity from time 19 to 60 (the next 7 days). Fig. 1c-e respectively show the prediction results of user-level [23], population-level [27], and group-level popularity. Clearly, our group-level solution makes the best predictions for both the number of appreciating users in total and within each individual group.

We develop in this paper a novel framework to predict the group-level of online content: *Given the set of users reacting to a content from time t_0 to time t_1, how many users in each group will react to that content until time $t_2 > t_1$?* We first group users into robust and cohesive clusters, and then perform tensor decomposition coupled with a hierarchical structure among groups to make predictions. Improvement in either of these two steps will lead to an improvement in the overall prediction accuracy. The proposed framework not only ensures scalability in dealing with large-scale social networks but also promises a high prediction accuracy. In order to minimize the impact of noise and be more flexible in capturing the changes in user interests, we exploit both the network topology among users and their interaction activities in learning a robust partition over all users. The PARAFAC tensor decomposition is further adapted to work with the hierarchical constraint over user groups, and we show that optimizing this constrained function via gradient descent achieves faster convergence and leads to more stable solution, as compared to other matrix factorizations. Here are our contributions:

- We propose to combine users' historical activities and the network structure into a network-constrained popularity graph. By clustering this graph, we put users into robust and meaningful groups that capture the evolution of online content's popularity over time.
- We propose to add a novel hierarchical constraint into a coupled tensor decomposition to simultaneously predict popularity at the group and population levels. We further prune data using top-k similarity queries to improve accuracy and reduce computational cost.
- We evaluate our framework, which we name GPOP (Group-level POpularity Prediction), on two real-world datasets collected from Twitter and Behance social networks: GPOP scales linearly with its parameters, and outperforms all baseline methods significantly in terms of accuracy. Our code and data are available online[1].

2. PROBLEM DEFINITIONS

[1]Code and data: http://cs.ucsb.edu/~mhoang/gpop.tar.gz

Let us denote $G = (V, E)$ a network where $V = \{v_1, v_2, \ldots, v_n\}$ is the set of nodes, each representing a user, and $E \subseteq V \times V$ is the set of edges representing the (undirected) connections among users. Let p_i be a content (e.g. a hashtag in Twitter) being broadcast in the network. We define the following essential concepts and problems in the paper.

Definition 2.1 (User-level popularity) *A user-level popularity of a content p_i at time t is defined by the vector $s_{it} = (\mathcal{S}_{it1}, \ldots, \mathcal{S}_{itn})$, where $\mathcal{S}_{itj} \in [0, +\infty)$ is the number of times user v_j has reacted to content p_i after the first t timestamps since p_i was created. We call $\mathcal{S}_{itj}$ the state of user v_j at timestamp t w.r.t. content p_i.*

In Definition 2.1, the popularity is a *non-decreasing quantity* over time: $\mathcal{S}_{itj} \leq \mathcal{S}_{it'j}\ \forall i, j$ and $t' > t$. If users are no longer interested in that content, its popularity will stay the same. We define the popularity to be the cumulative number of reactions instead of the number of new reactions at each timestamp since the latter in practice is very noisy, as visualized in Fig. 1a. Additionally, all timestamps are *relative* to the creation time of each content.

Problem 1 (User Clustering) *Given a network G, a set of contents $P = \{p_1, p_2, \ldots, p_m\}$, and the number of clusters l, find a partition of the users $C = \{C_1, C_2, \ldots, C_l\}$ that reflects the spread of the popularity of contents in P, where $C_j \cap C_{j'} = \emptyset$, $\cup_{j=1}^{l} C_j = V$.*

Definition 2.2 (Group-level popularity) *Given C as an optimal solution for Problem 1, the group-level popularity of a content p_i at timestamp t is the vector $x_{it} = (\mathcal{X}_{it1}, \ldots, \mathcal{X}_{itl})$, where $\mathcal{X}_{itj}$ is the total number of times users in group C_j have reacted to p_i after the first t timestamps since p_i was created, that is, $\mathcal{X}_{itj} = \sum_{v_h \in C_j} \mathcal{S}_{ith}$. For brevity, we call $\mathcal{X}_{itj}$ the state of group C_j at timestamp t w.r.t. p_i. Finally, in case $C = \{V\}$, we have the population-level popularity.*

Problem 2 (Group-level Popularity Prediction) *Given a network G, a set of historical contents $P = \{p_1, p_2, \ldots, p_m\}$ (each content was observed over q timestamps), a set of user groups $C = \{C_1, C_2, \ldots, C_l\}$, and the group-level popularity of a new content p_{m+1} during its first t_1 timestamps ($t_1 < q$), predict the group-level popularity of p_{m+1} during time period $[t_1 + 1, q]$, that is, given $\{x_{m+1,1}, x_{m+1,2}, \ldots, x_{m+1,t_1}\}$, predict $\{x_{m+1,t_1+1}, \ldots, x_{m+1,q}\}$.*

We solve Problem 1 in Section 3 and Problem 2 in Section 4.

Example 2. For a project in Behance, we want to predict how many users in total and in each user group would appreciate it. Fig. 2a shows the group-level popularity of the same project in Fig. 1b (normalized by the total number of appreciating users at time t_1). Fig. 2c shows our popularity prediction for this project using its top-3 similar projects in Fig 2d. Our prediction is very close to the ground truth.

3. USER CLUSTERING

For Problem 1, we first discuss the four goals, G1-G4, of clustering users, and then propose how to achieve the goals.

3.1 Clustering goals

G1: Group users with similar interests and behaviors. First, the behaviors and interests of users should be similar within the same group and different across different groups, leading to meaningful and useful groups for

real-world applications. For example, an ad campaign may choose to target only a few relevant groups, knowing that the users in these groups will react to the advertised products instead of wasting money on other irrelevant groups. Similarly, web content can be prefetched in batches to only groups of users that are more likely to react to that content, avoiding unnecessary bandwidth and storage cost.

G2: Capture future changes in user interests. Using past user behaviors to cluster users is prone to overfitting: the obtained groups are good for historical contents, but may fail to capture a change of user interests on unobserved future content for two reasons. First, the spread of a content in a network is highly random and noisy, especially at the user level [7]. Fitting the clusters too tightly to the historical data would thus capture this noise. Second, many users are not active and react to few contents. Such users can be put in any group without incurring a significant cost in the clustering objective, making the cluster membership more random and less powerful in modeling future events.

G3: Capture the paths of information spread. There are two main ways a user gets exposed to a new content, which may in turn trigger him/her to react to that content: (i) via the network structure, or (ii) via an external source. For example, on Twitter, a user can learn about a new hashtag either (i) by reading his/her friends' tweets, or (ii) via other websites. Similarly, in Behance.net, a user will be exposed to a new project either (i) if his/her friends have performed some actions on that project, or (ii) if s/he actively searches for the project using some side information.

G4: Avoid imbalanced user groups. In clustering users, we may obtain groups of largely varying sizes while still optimizing some clustering objective. For example, simply minimizing the edge cut in a graph among users may lead to one group with almost all users and many tiny groups with only a few users. Such a partition of the users is hardly useful for reducing cost in a targeted ad campaign or group-based web-content caching/prefetching. Thus, we propose to find groups with comparable sizes.

3.2 Clustering network-constrained popularity graph

For goal G1, users v_i and $v_{i'}$ should be more likely to be in the same group if they tend to react to the same contents at the same times, i.e., $\exists t, p_j$ s.t. $\mathcal{S}_{itj} > 0$ and $\mathcal{S}_{i'tj} > 0$. This is equivalent to clustering the following popularity graph $G^{\mathcal{S}}$ into separate groups of vertices to minimize the edge cut.

Definition 3.1 (Popularity graph) Given a user set $V = \{v_1, \ldots, v_n\}$, and the user-level popularity $\mathcal{S}$ over q timestamps of m historical contents $P = \{p_1, \ldots, p_m\}$, denote the popularity vertex set $S = \{s_{11}, \ldots, s_{it}, \ldots, s_{mq}\}$ as the set of all combinations of m contents and q timestamps. The popularity graph $G^{\mathcal{S}} = (V^{\mathcal{S}}, E^{\mathcal{S}}, N^{\mathcal{S}}, W^{\mathcal{S}})$ is then defined as a weighted undirected bipartite graph (see Fig. 3a) with vertex set $V^{\mathcal{S}} = V \cup S$, edge set $E^{\mathcal{S}} = \{(v_j, s_{it}) | \mathcal{S}_{itj} > 0\}$, vertex weights $N^{\mathcal{S}}$, and edge weights $W^{\mathcal{S}}$, such that $\forall v_j, v_{j'} \in V; s_{it}, s_{i't'} \in S$: $N^{\mathcal{S}}_{v_j} = 1; N^{\mathcal{S}}_{s_{it}} = 0$; $W^{\mathcal{S}}_{v_j, s_{it}} = W^{\mathcal{S}}_{s_{it}, v_j} = \mathcal{S}_{itj}$; $W^{\mathcal{S}}_{v_j, v_{j'}} = 0$; and $W^{\mathcal{S}}_{s_{it}, s_{i't'}} = 0$.

$G^{\mathcal{S}}$ captures past user behaviors but it does not help us with goals G2 and G3. Fortunately, even if future user interests are different from those obtained from historical data, a new content must still spread via the network structure G unless users actively approach this content via some external sources. Thus, the network structure can be included in the clustering framework to deal with this change. In other words, we claim that clustering the following network-constrained popularity graph G^*, which is the union of G and $G^{\mathcal{S}}$, will help us satisfy both goals G2 and G3.

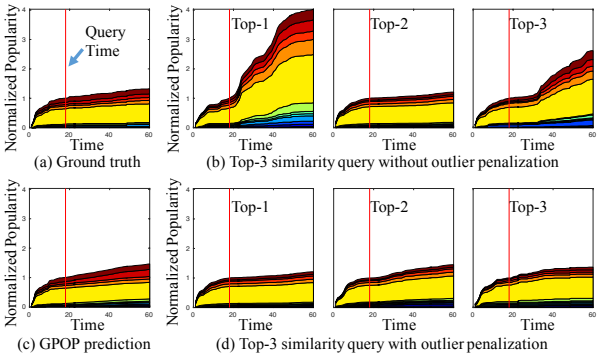


Figure 2: Example top-3 similarity query and prediction after the first 3 days (marked by the vertical line at $t_1 = 18$) for Behance. Popularity is normalized by the population-level popularity at t_1.

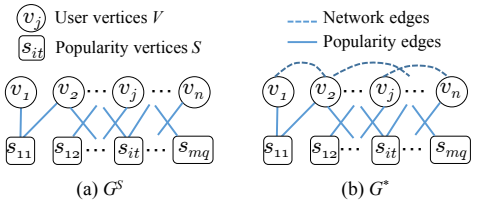


Figure 3: (a) Popularity graph $G^{\mathcal{S}}$ and (b) network-constrained popularity graph G^*

Definition 3.2 (Network-constrained popularity graph) Given a graph $G = (V, E)$ and its popularity graph $G^{\mathcal{S}} = (V^{\mathcal{S}}, E^{\mathcal{S}}, N^{\mathcal{S}}, W^{\mathcal{S}})$, we define the network-constrained popularity graph (Fig. 3b) as $G^* = (V^*, E^*, N^*, W^*)$ with vertex set $V^* = V^{\mathcal{S}}$, edge set $E^* = E^{\mathcal{S}} \cup E$, vertex weights $N^* = N^{\mathcal{S}}$, and edge weights W^*, such that $\forall v_j, v_{j'} \in V$; $s_{it}, s_{i't'} \in S$: $W^*_{v_j, s_{it}} = W^*_{s_{it}, v_j} = \mathcal{S}_{itj}$; $W^*_{v_j, v_{j'}} = \max(A_{jj'}, A_{j'j})$; and $W^*_{s_{it}, s_{i't'}} = 0$, where A is the adjacency matrix of G.

By using G^*, the cluster membership of an inactive user can be decided more effectively: s/he is more likely to be in the same cluster with her/his friends, rather than some random users that are very far away in G but coincidentally active at the same time.

Finally, goal G4 will be satisfied if we cluster G^* with a balancing criteria: we would like to obtain a partition $C = \{C_1, \ldots, C_l\}$ of V, such that $|C_j| \approx |C_{j'}|\ \forall C_j, C_{j'} \in C$. However, in clustering G^*, we actually obtain a partition $C^* = \{C^*_1, \ldots, C^*_l\}$ of V^*, such that $C^*_j \cap C^*_{j'} = \emptyset$, $\cup_j C^*_j = V^*$. We can easily convert C^* into C by choosing $C_j = C^*_j \cap V\ \forall 1 \le j \le l$. Moreover, define the weight $w(C^*_j)$ of each group C^*_j as the sum of all vertex weights in C^*_j, then:

$$w(C^*_j) = \textstyle\sum_{v_i \in C^*_j \cap V} N^*_{v_i} + \sum_{s_{it} \in C^*_j \cap S} N^*_{s_{it}} = |C_j|$$

Thus, the balancing criteria on C can be translated into a balancing criteria on C^*, i.e., $w(C^*_i) \approx w(C^*_j)\ \forall C^*_i, C^*_j \in C^*$.

Clustering objectives: Based on the above intuitions, we cluster G^* with two objectives to satisfy all goals G1-G4:

1. Weighted edge cut minimization:

$$\min_{C^*} \textstyle\sum_{u \in C^*_i, v \in C^*_j, i \ne j} W^*_{u,v}$$

Algorithm 1 Find-User-Groups

Input: Network $G = (V, E)$. Number of user groups l
Historical contents $P = \{p_1, \ldots, p_m\}$. Imbalance factor β
Output: $C = \{C_1, \ldots, C_l\}$
1: $\mathcal{S} \leftarrow$ Network state tensor of P
2: $G^* \leftarrow$ Network-constrained popularity graph$(G, \mathcal{S})$
3: $C^* \leftarrow$ Part-Graph-K-way(G^*, l, β)
4: $C \leftarrow \{C_j^* \cap V | j = 1, \ldots, l\}$
5: **return** C

2. Group balancing: $\frac{w(C_j^*)}{n/l} < 1 + \beta \ \forall C_j^* \in C^*$, where $\beta > 0$ is a predefined imbalance factor.

The first objective assures that each user group is homogeneous since it tries to minimize the amount of weighted edges between different groups. The second objective guarantees that group sizes do not deviate too far from the attainable average size n/l, where l is the desired number of groups.

Clustering algorithm: We use the multilevel k-way partitioning algorithm [17] (Part-Graph-K-way) for graph clustering since it is scalable and supports our two objectives. The final clustering procedure is summarized in Algorithm 1.

Example 3. Fig. 4a shows the average group sizes over 5-fold cross validation clustering for our Behance data: the groups clearly have similar sizes. Fig. 4b shows the group-level popularity of one example project in the testing set (one fold) given three different partitions of users obtained by clustering G^*, G, and $G^{\mathcal{S}}$ on the training data (the other four folds). Clearly, combining G and $G^{\mathcal{S}}$ to get G^* makes the obtained user groups more homogeneous on testing data.

4. HIERARCHICAL PREDICTION

For Problem 2, we find the top-k similar contents at group and population levels for p_{m+1}, and then perform coupled tensor decomposition on these top-k contents with a novel hierarchical constraint to predict p_{m+1}'s future popularity.

4.1 A baseline approach

Once the groups $C = \{C_1, \ldots, C_l\}$ are defined, we can create a *group-level polularity tensor* $\mathcal{X}$ for $P \cup \{p_{m+1}\}$ as shown in the left hand side of Fig. 5, where $\mathcal{X}_{itj}$ is defined as in Definition 2.2, and $\mathcal{X}_{m+1,t,j}$ is missing for all $t > t_1$. We can perform tensor completion to fill in these missing values and predict p_{m+1}. In particular, we decompose $\mathcal{X}$ into three matrices $D \in \mathbb{R}^{(m+1)\times R}$, $J \in \mathbb{R}^{q\times R}$, $F \in \mathbb{R}^{l\times R}$ using PARAFAC [19] such that, for all observed entries:

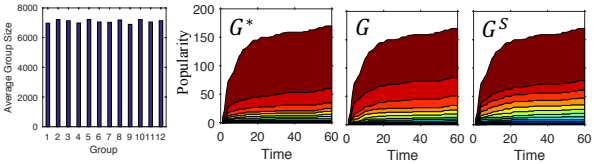
$$\mathcal{X}_{itj} = \sum_{r=1}^{R} D_{ir} J_{tr} F_{jr} \quad (1)$$

or equivalently, $\mathcal{X} = [[D, J, F]]$, where D is the factor matrix for contents in $P \cup \{p_{m+1}\}$, J is the factor matrix for q timestamps, F is the factor matrix for l groups, and R is the number of latent dimensions. To learn D, J, and F, we minimize this objective function using gradient descent [4]:

$$\mathcal{L} = \tfrac{1}{2}\|\mathcal{M} * (\mathcal{X} - [[D, J, F]])\|_F^2 + \tfrac{\lambda}{2}(\|D\|_F^2 + \|J\|_F^2 + \|F\|_F^2) \quad (2)$$

where "$*$" is the element-wise tensor product, $\|.\|_F$ is the Frobenius norm, $\lambda > 0$ is a regularization factor to avoid overfitting, and $\mathcal{M}$ is a mask tensor of the same size as $\mathcal{X}$, indicating observed entries in $\mathcal{X}$, i.e., $\mathcal{M}_{itj} = 0$ iff $i = m+1, t_1 < t \le q$, and $\mathcal{M}_{itj} = 1$ otherwise.

Drawbacks: (i) P can be large and contains contents vastly different from p_{m+1}, causing unnecessary computational cost and degrading accuracy. (ii) $\mathcal{M}$ is a dense tensor, causing huge memory and computational cost if $\mathcal{X}$ is large.



(a) Group size (b) Popularity with groups from different graphs

Figure 4: Behance data: (a) Group sizes (for G^*, 5-fold cross validation); (b) Group-level popularity of an example project.

4.2 Finding Top-k Similar Contents

Due to the drawbacks in Section 4.1, we claim that including only the top-k contents in P that are similar to p_{m+1} in tensor $\mathcal{X}$ makes $\mathcal{X}$ smaller and more relevant.

In Equation 1, predicting p_{m+1} is equivalent to learning the $(m+1)$th row of D. If J, F and the first m rows of D are fixed, this task is equivalent to representing row $(m+1)$th as a linear combination of the first m rows in D. Thus, the best candidate for predicting the $(m+1)$th row is some row i_1 in D, if it exists, such that $D_{m+1,r} = \beta D_{i_1,r}\ \forall 1 \le r \le R$ for some $\beta \in \mathbb{R}$. This is because we then need to learn only a single parameter β to make a perfect prediction for p_{m+1}:

$$\mathcal{X}_{m+1,t,j} = \sum_{r=1}^{R} \beta D_{i_1 r} J_{tr} F_{jr} = \beta \mathcal{X}_{i_1 tj}$$

for $\forall 1 \le t \le q; 1 \le j \le l$. Therefore, we propose to find top-k similar contents for p_{m+1} in a normalized space: given the training time period $[1, t_1]$, we normalize each content in $\mathcal{X}$ by its population-level popularity at time t_1 as follows:

$$\tilde{\mathcal{X}}_{itj} = \mathcal{X}_{itj} / \textstyle\sum_j \mathcal{X}_{it_1 j} \quad \forall i, t, j \quad (3)$$

We then define the distance at timestamp T between p_{m+1} and another content p_i as the Euclidean distance:

$$\delta_T(p_i, p_{m+1}) = \sqrt{\textstyle\sum_{t=1,\ldots,T; j=1,\ldots,l} (\tilde{\mathcal{X}}_{itj} - \tilde{\mathcal{X}}_{m+1,t,j})^2} \quad (4)$$

The top-k similar contents are then the contents with the smallest distance to p_{m+1} at time t_1.

Outliers: The top-k contents may be similar to p_{m+1} at time t_1, but very different from p_{m+1} in the future due to some unforeseeable events after t_1. For example, on Behance.net, a project could be promoted by the website and becomes popular even though it was barely noticed before. Similarly, in Twitter, some real-world events outside the social network may boost the usage of some hashtags suddenly. Therefore, we reduce the impact of such a historical outlier by including an outlierness score defined as the average distance at time q between it and the rest of the historical contents. The new distance δ_T^{out} at time T is defined as:

$$\delta_T^{out}(p_i, p_{m+1}) = \tfrac{\delta_T(p_i, p_{m+1})}{m-1} \times \textstyle\sum_{j=1,\ldots,m; j\neq i} \delta_q(p_i, p_j) \quad (5)$$

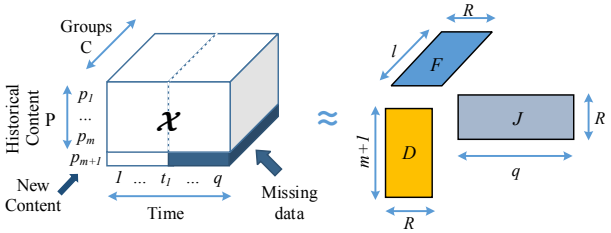


Figure 5: Predict missing values in group-level popularity tensor $\mathcal{X}$ using PARAFAC [19]. All timestamps are relative to the creation time of each content.

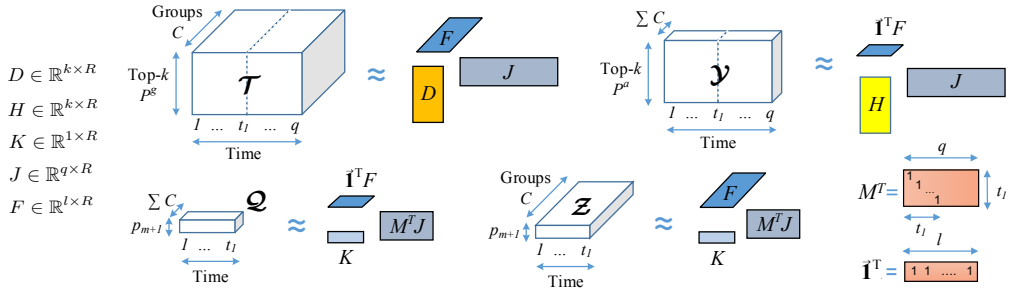


Figure 6: Hierarchical prediction using coupled tensor factorization. P^g (and P^a) are the top-k contents with similar group-level (and population-level) popularity as p_{m+1} over the time period $[1, t_1]$. $M^T J$ contains the first t_1 rows of J; $\vec{1}^T F$ is the sum of rows of F.

Example 4. Fig. 2(b, d) show an example top-k query using δ and δ^{out}. Since δ does not penalize outliers, it returns the top-k contents (Fig. 2b) that are very different from the querying content (Fig. 2a) after time t_1. Whereas, δ^{out} gives us significantly better top-k results (Fig. 2d).

Algorithm 2 summarizes the procedure for finding top-k similar contents given historical data and user groups.

4.3 Tensor-based Hierarchical Prediction

Since the population level is less noisy, we borrow its strength to make better predictions for the group level by combining them in a hierarchical prediction framework.

Algorithm 3 summarizes how to hierarchically predict a new content p_{m+1}. First, in Lines 1-2, we use Algorithm 2 to find its top-k similar contents in P during time $[1, t_1]$ at group level (P^g) and population level (P^a). The latter is a special case of the former, where $C = \{V\}$. In line 3, we next build four tensors $\mathcal{T}$, $\mathcal{Y}$, $\mathcal{Z}$, $\mathcal{Q}$ as shown in Fig. 6:

$$\mathcal{T} \in \mathbb{R}^{k \times q \times l}; \mathcal{Y} \in \mathbb{R}^{k \times q \times 1}; \mathcal{Z} \in \mathbb{R}^{1 \times t_1 \times l}; \mathcal{Q} \in \mathbb{R}^{1 \times t_1 \times 1}$$

$$\mathcal{T}_{itj} = \sum_{v_h \in C_j} \mathcal{S}_{ith} \ \forall p_i \in P^g; 1 \le t \le q; 1 \le j \le l \quad (6)$$

$$\mathcal{Y}_{it1} = \sum_{v \in V} \mathcal{S}_{ith} \ \forall p_i \in P^a; 1 \le t \le q \quad (7)$$

$$\mathcal{Z}_{1tj} = \sum_{v_h \in C_j} \mathcal{S}_{m+1,t,h} \ \forall 1 \le t \le t_1; 1 \le j \le l \quad (8)$$

$$\mathcal{Q}_{1t1} = \sum_{v_h \in V} \mathcal{S}_{m+1,t,h} \ \forall 1 \le t \le t_1 \quad (9)$$

$\mathcal{T}$ and $\mathcal{Y}$ store the group-level and population-level popularity of contents in P^g and P^a respectively. $\mathcal{Z}$ and $\mathcal{Q}$ store the group-level and population-level popularity of p_{m+1} during time $[1, t_1]$ respectively. In line 4, we decompose these tensors into five factor matrices (Fig. 6) as detailed here:

$$D \in \mathbb{R}^{k \times R}; H \in \mathbb{R}^{k \times R}; K \in \mathbb{R}^{1 \times R}; J \in \mathbb{R}^{q \times R}; F \in \mathbb{R}^{l \times R}$$

$$\mathcal{T} \approx [[D, J, F]] = \mathcal{T}^* \quad (10)$$

$$\mathcal{Y} \approx [[H, J, \vec{1}^T F]] = \mathcal{Y}^* \quad (11)$$

$$\mathcal{Z} \approx [[K, M^T J, F]] = \mathcal{Z}^* \quad (12)$$

$$\mathcal{Q} \approx [[K, M^T J, \vec{1}^T F]] = \mathcal{Q}^* \quad (13)$$

where R is the chosen number of latent dimensions; $\vec{1}$ is an all-one column vector with l elements; and M is a $q \times t_1$ mask matrix to extract the first t_1 rows of matrix J, i.e.,

$$M_{ii} = 1 \ \forall i \text{ and } M_{ij} = 0 \ \forall i \neq j \quad (14)$$

To predict p_{m+1} (Lines 5-10), we use K, F, and the last $q - t_1$ rows of J (corresponding to time period $[t_1 + 1, q]$).

Intuitively, the rows of D and H represent the contents in P^g and P^a respectively; the rows of J represent the q timestamps; K has only one row representing the new content p_{m+1}; and the rows of F represent l user groups in C. Additionally, $\vec{1}^T F$ is the sum of the rows in F, representing the population-level popularity; while $M^T J$ are the first t_1 rows of J, representing the observed time period $[1, t_1]$.

Equations 10 and 11 capture the latent representations at the group and population levels using the historical contents respectively; whereas Equations 12 and 13 map p_{m+1} to the same latent space as that of the historical contents by sharing the factor matrices for time J and groups F. Since data for p_{m+1} is incomplete, only the observed part $M^T J$ of J is shared. Finally, by sharing factor matrix F in these four equations, we effectively learn a *hierarchical model* at both the group and population levels simultaneously.

Why coupled tensor decomposition? While $\mathcal{Y}$ and $\mathcal{Z}$ can be represented as matrices, and $\mathcal{Q}$ can be represented as a vector instead of tensors, we choose to use tensors in our formulation because of two reasons. First, PARAFAC decompositions are often unique, leading to more stable results and faster convergence compared to other matrix factorizations (which are often not unique, except for SVD) [19]. Second, the hierarchical structure of the user groups are naturally reflected in the decomposition when $\mathcal{Y}$, $\mathcal{Z}$ and $\mathcal{Q}$ are represented as tensors. In particular, $\mathcal{Y}$ and $\mathcal{Q}$ are simply the collapsed versions of $\mathcal{T}$ and $\mathcal{Z}$ along the group (3^{rd})

Algorithm 2 Top-k

Input: Historical contents $P = \{p_1, p_2, \ldots, p_m\}$
Partially observed content p_{m+1}
Maximum observed timestamp $t_1 < q$
User groups $C = \{C_1, C_2, ...\}$. Number of top items k
Output: Top-k content IDs
1: Construct tensor $\mathcal{X}$ for $P \cup \{p_{m+1}\}$ & C (Definition 2.2)
2: $\tilde{\mathcal{X}} \leftarrow$ Normalize $\mathcal{X}$ at time t_1 using Equation 3
3: **for** $i := 1$ to m **do**
4: $\quad d_i \leftarrow \delta^{out}_{t_1}(p_i, p_{m+1})$ {Equation 5}
5: **end for**
6: **return** Top-k indices with the smallest values in d

Algorithm 3 GPOP (Group-level POpularity Prediction)

Input: Partially observed content p_{m+1} during time $[1, t_1]$.
Historical contents $P = \{p_1, \ldots, p_m\}$. User groups $C = \{C_1, ..., C_l\}$. Number of latent dimensions R
Output: Prediction of p_{m+1}: $\{x_{m+1,t_1+1}, \ldots, x_{m+1,q}\}$
1: $P^g \leftarrow$ Top-$k(P, p_{m+1}, t_1, C)${group level}
2: $P^a \leftarrow$ Top-$k(P, p_{m+1}, t_1, \{V\})${population level}
3: Create $\mathcal{T}, \mathcal{Y}, \mathcal{Z}, \mathcal{Q}$ using Equations 6-9
4: $D, H, K, J, F \leftarrow$ Factorize-Tensors$(\mathcal{T}, \mathcal{Y}, \mathcal{Z}, \mathcal{Q}, R)$
5: $J^* \leftarrow$ Rows $(t_1 + 1)$ to q of J
6: $\mathcal{Q}^* \leftarrow [[K, J^*, F]]$
7: $x_{m+1,t_1+i} \leftarrow \{\mathcal{Q}^*_{1,i,j} | j = 1, \ldots, l\} \ \forall i = 1, \ldots, q - t_1$
8: **return** $\{x_{m+1,t_1+1}, \ldots, x_{m+1,q}\}$

Algorithm 4 Factorize-Tensors

Input: Tensors $\mathcal{T}, \mathcal{Y}, \mathcal{Z}, \mathcal{Q}$ as in Equations 6-9
R: Number of latent dimensions
Output: Factor matrices D, H, K, J, F
Compute $\mathcal{L}$ using Equation 15
while not converged **do**
  Compute step length α
  Compute the gradients $\nabla_D\mathcal{L}, \nabla_H\mathcal{L}, \nabla_K\mathcal{L}, \nabla_J\mathcal{L}, \nabla_F\mathcal{L}$ using Equations 16-20
  $D \leftarrow D - \alpha\nabla_D\mathcal{L}$; $H \leftarrow H - \alpha\nabla_H\mathcal{L}$; $K \leftarrow K - \alpha\nabla_K\mathcal{L}$
  $J \leftarrow J - \alpha\nabla_J\mathcal{L}$; $F \leftarrow F - \alpha\nabla_F\mathcal{L}$
  Compute $\mathcal{L}$ using Equation 15
end while
return D, H, K, J, F

dimension respectively. Thus, the factor along the 3^{rd} dimension of $\mathcal{Y}$ and $\mathcal{Q}$ is also the sum of the rows in the factor for $\mathcal{T}$ and $\mathcal{Z}$ along the 3^{rd} dimension.

Optimization solution: Algorithm 4 shows how to find the five factor matrices by using gradient descent to minimize the following objective function:

$$\begin{aligned}\mathcal{L} = &\tfrac{1}{2}\|\mathcal{T} - [[D, J, F]]\|_F^2 + \tfrac{1}{2}\|\mathcal{Y} - [[H, J, \vec{\mathbf{1}}^T F]]\|_F^2 \\ &+ \tfrac{1}{2}\|\mathcal{Z} - [[K, M^T J, F]]\|_F^2 + \tfrac{1}{2}\|\mathcal{Q} - [[K, M^T J, \vec{\mathbf{1}}^T F]]\|_F^2 \\ &+ \tfrac{\lambda}{2}(\|D\|_F^2 + \|J\|_F^2 + \|F\|_F^2 + \|H\|_F^2 + \|K\|_F^2) \end{aligned} \quad (15)$$

where $\lambda > 0$ is a regularization factor to avoid overfitting.

The gradients of $\mathcal{L}$ are:

$$\nabla_D\mathcal{L} = (\mathcal{T}^*_{(1)} - \mathcal{T}_{(1)})(F \odot J) + \lambda D \quad (16)$$

$$\nabla_H\mathcal{L} = (\mathcal{Y}^*_{(1)} - \mathcal{Y}_{(1)})(\vec{\mathbf{1}}^T F \odot J) + \lambda H \quad (17)$$

$$\begin{aligned}\nabla_K\mathcal{L} = &(\mathcal{Z}^*_{(1)} - \mathcal{Z}_{(1)})(F \odot (M^T J)) \\ &+ (\mathcal{Q}^*_{(1)} - \mathcal{Q}_{(1)})(\vec{\mathbf{1}}^T F \odot (M^T J)) + \lambda K \end{aligned} \quad (18)$$

$$\begin{aligned}\nabla_J\mathcal{L} = &(\mathcal{T}^*_{(2)} - \mathcal{T}_{(2)})(F \odot D) + (\mathcal{Y}^*_{(2)} - \mathcal{Y}_{(2)})(\vec{\mathbf{1}}^T F \odot H) \\ &+ M(\mathcal{Z}^*_{(2)} - \mathcal{Z}_{(2)})(F \odot K) \\ &+ M(\mathcal{Q}^*_{(2)} - \mathcal{Q}_{(2)})(\vec{\mathbf{1}}^T F \odot K) + \lambda J \end{aligned} \quad (19)$$

$$\begin{aligned}\nabla_F\mathcal{L} = &(\mathcal{T}^*_{(3)} - \mathcal{T}_{(3)})(J \odot D) + \vec{\mathbf{1}}(\mathcal{Y}^*_{(3)} - \mathcal{Y}_{(3)})(J \odot D) \\ &+ (\mathcal{Z}^*_{(3)} - \mathcal{Z}_{(3)})(M^T J \odot K) \\ &+ \vec{\mathbf{1}}(\mathcal{Q}^*_{(3)} - \mathcal{Q}_{(3)})(M^T J \odot K) + \lambda F \end{aligned} \quad (20)$$

where $\mathcal{T}_{(i)}$ denotes the mode-i matricization of $\mathcal{T}$, and "$\odot$" denotes the Khatri-Rao product as defined in [19].

5. EXPERIMENTS

5.1 Datasets

We use two real-world datasets for evaluation: **Behance** [1] and **Twitter** [21, 35] (see Table 1). Behance.net is a social network where users can share their creative works (projects) and appreciate each other's projects. Twitter is a micro-blogging platform where users post short messages (tweets) that may include hashtags. There is a directed following relationship among users in both these social networks. Since we only care if two neighboring users are active at the same time, we convert these networks into undirected networks. A content is a project in Behance or a hashtag in Twitter. The popularity of a project is the number of users who have appreciated it; whereas the popularity of a hashtag is the number of times it has been tweeted by users. We only use contents with at least 100 reacting users, and also remove users with less than 10 tweets on Twitter.

Table 1: Datasets

Datasets	Behance	Twitter
Data time range	June, 2014	Sep-Dec, 2009
#Users	85092	22255
#Edges (follows)	13428465	575819
#Contents	1326 projects	1015 hashtags
#Timestamps	60	24
Timestamp bin size	4 hours	4 hours
Prediction task		
History length t_1	18 (3 days)	12 (2 days)
Future length $(q - t_1)$	12 (2 days)	12 (2 days)

Comprehensive experimental analyses for both Twitter and Behance can be found in our technical report [2].

5.2 Quality of user clustering

We evaluate the quality of user groups in Problem 1 using entropy. A smaller entropy means more homogeneous groups. Given a content p_i, and a timestamp t, we define the active probability of users in a group C_j as the proportion of users with non-zero states, i.e., $p_{act} = |\{v_h \in C_j | \mathcal{S}_{ith} > 0)\}|/|C_j|$. Then, the entropy of $C = \{C_1, \ldots, C_l\}$ w.r.t. a set of historical contents P during a time period $[1, q]$ is:

$$h(C) = \sum_{j=1}^{l} \frac{|C_j|}{|V|} \frac{1}{mq} \sum_{p_i \in P; 1 \le t \le q} h(C_j, p_i, t) \quad (21)$$

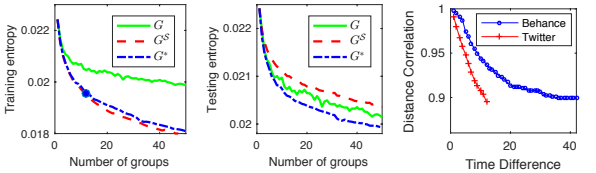
where $h(C_j, p_i, t) = -p_{act} \log p_{act} - (1 - p_{act}) \log (1 - p_{act})$

Effect of network structure: Fig. 7(a, b) show the average entropy over 5-fold cross validation as the number of groups varies when G, $G^{\mathcal{S}}$ and G^* are clustered for Behance. Obviously, the higher the number of clusters, the smaller the entropy. More importantly, the effect of overfitting can be seen clearly. Clustering the tensor graph $G^{\mathcal{S}}$ provides the best groups on the training sets; but the obtained groups fit the testing sets very poorly. On the contrary, the qualities of groups obtained from the network G are similar for both the training and testing sets, suggesting that the effect of the network structure is consistent across different contents. Finally, groups obtained from G^* have comparable quality to those from $G^{\mathcal{S}}$ on training sets, while superior to both the groups obtained from G and $G^{\mathcal{S}}$ on the testing sets.

Number of user groups: We use the elbow method [25] to choose the best number of clusters for each dataset: 12 for Behance (see Fig. 7a), and 11 for Twitter. Fig. 8 further shows the word clouds of users' topics of interest (the tags of appreciated projects) in 8 of the 12 user groups in Behance. Clearly, users' interests are consistent within each group.

5.3 Usability of top-k similarity queries

For the top-k prediction strategy in Section 4.2 to work, the distances between two contents should be consistent over time. Fig 7c shows the Pearson correlations between the



(a) Training entropy (b) Testing entropy (c) Distance correlation

Figure 7: (a,b) 5-fold clustering results for Behance. The best number of clusters chosen by the elbow method [25] is marked by a blue circle in figure (a). (b) Pearson correlation of distances at time t_1 and t_2 v.s. $(t_2 - t_1)$

distances computed at two different timestamps ($t_1 = 12$, $t_2 \in [19, 60]$ for Behance, $t_2 \in [19, 24]$ for Twitter). Though the correlations decrease as the time differences increase for both Behance and Twitter, the correlations remain high (> 0.89). Using top-k for prediction is thus a reasonable choice.

5.4 Quality of hierarchical prediction

5.4.1 Settings

Baselines: All compared methods are listed in Table 2 and divided into four groups: (i) GPOP and its variants, (ii) other tensor decomposition approaches for popularity prediction, (iii) hierarchical time series prediction, and (iv) population-level prediction only. We also note in Table 2 at which levels the predictions are performed for each method (user, group, and population levels).

Parameter setting: We set R as 50 for group-level and 100 for user-level predictions; imbalance factor $\beta = 0.03$; $k = 10$; $\lambda = 0.1$ (chosen using cross validation). t_1 and q are chosen as in Table 1. To cope with the instability of random initializations for gradient descent, we run each tensor decomposition three times, and choose the best results. Predictions are done for 5-fold cross validation. Experiments are run on a Debian machine with Intel i7, 3.50GHz CPU and 15GB RAM. Codes are written in Matlab, using the Tensor Toolbox [5], Poblano Toolbox [8] and METIS library [16].

Evaluation: We define the *Relative mean Error* for the group (REG) and population (REP) levels as below:

$$REG = \frac{1}{m}\sum_i \frac{\sqrt{\sum_{t>t_1,j}(\mathcal{X}_{itj} - \hat{\mathcal{X}}_{itj})^2}}{\sqrt{\sum_{t>t_1,j}\mathcal{X}_{itj}^2}} \times 100\%$$

$$REP = \frac{\sum_{i,t>t-1}|\sum_j \mathcal{X}_{itj} - \sum_j \hat{\mathcal{X}}_{itj}|}{\sum_{i,t>t_1,j}\mathcal{X}_{itj}} \times 100\%$$

where $\hat{\mathcal{X}}$ contains the predicted group-level popularity.

5.4.2 Quantitative Performance

The prediction results for all compared methods are shown in Table 3: our GPOP framework consistently outperforms all baselines. We next discuss the results in more details.

Choice of clustered graphs: The first three rows of Table 3 show the prediction errors for user groups obtained from G^*, G, and $G^{\mathcal{S}}$ respectively. Since the clusters from G^* are the most homogeneous, they produce the smallest errors for groups as well as smaller errors for population.

Variants of GPOP: We verify the effectiveness of GPOP's two components: top-k similarity query, and tensor-based hierarchical prediction. As shown in Table 3, GPOP outperforms GPOP-NoTop, which uses all historical contents instead of just the top-k similar contents, proving the benefit of top-k prediction. GPOP is also better than GPOP-NoNorm, i.e., computing top-k on the normalized popularity $\tilde{\mathcal{X}}$ (Equation 3) is better than on the raw data $\mathcal{X}$. Next, GPOP is far superior to naively taking the average of the top-k similar contents (Group-Avg). Predicting each group separately (GroupSep) and ignoring the relationship among groups is also significantly worse than GPOP, confirming the advantages of hierarchical prediction. Finally, predicting at the group level is much easier than at the noisy user level, as shown by the extremely high errors of GPOP-User.

Other tensor-based approaches: We test three different options. First, coupled matrix-tensor factorization (CMTF [4]) uses both the tensor $\mathcal{S}$ and the adjacency matrix of G to predict at user level. Here, we naively set the same weights for $\mathcal{S}$ and G in its objective function. The noise and high number of latent dimensions to be learned lead to extremely high errors. The network G thus actually makes it even more difficult for CMTF to converge to a good solution. Second, we test a probabilistic tensor decomposition method named TriMine [23], which is oblivious to the network G, and only uses the time period $[1, t_1]$ in tensor $\mathcal{S}$ for learning and predicting. It also predicts at the user level, and thus is prone to noise, leading to a much higher error compared to GPOP. Finally, we evaluate tensor completion for the group-based tensor $\mathcal{X}$ using PARAFAC (CP-wopt [3]), as discussed in Equation 2. We only test CP-wopt for $\mathcal{X}$ (group-level) because it does not scale for $\mathcal{S}$ (user-level)—the mask tensor $\mathcal{M}$ in Equation 2 is huge

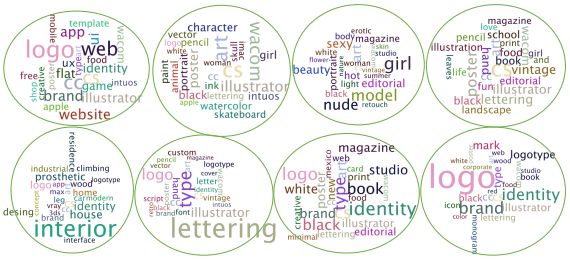


Figure 8: Word clouds of project tags for 8 user groups (Behance).

Table 2: Baselines, with notes on three levels of prediction: (⋆) Group, (†) Population, (‡) User.

Methods	Descriptions
GPOP ⋆†	top-k + hierarchical prediction
Variants of GPOP	
GPOP-NoTop ⋆†	GPOP that uses all historical contents instead of top-k similar content
GPOP-NoNorm ⋆†	GPOP where top-k is computed on $\mathcal{X}$ instead of $\tilde{\mathcal{X}}$ as in Equation 3
Group-Avg ⋆	Weighted average of top-k similar content
GroupSep ⋆	top-k + separate predictions for each group using GPOP with $\mathcal{Y}$ and $\mathcal{Q}$
GPOP-User †‡	GPOP with each user as a group
Other tensor-based approaches	
CMTF [4] ‡	Coupled matrix-tensor factorization ($\mathcal{S}$ & G)
TriMine [23] ‡	Co-evolving time-series prediction of $\mathcal{S}$
CP-wopt [3] ⋆	PARAFAC tensor completion (Eqn. 2)
Hierarchical time series prediction [13]	
ARIMA-COMB ⋆	ARIMA + optimal combination
ARIMA-BU ⋆	ARIMA + bottom-up
ETS-COMB ⋆	Exponential smoothing + opt. combination
ETS-BU ⋆	Exponential smoothing + bottom-up
Population-level popularity prediction only	
MRBF [27]†	Multivariate linear & Radial Basis Function

Table 3: Relative prediction errors (%). (*) marks experiments that did not finish after 1 day.

		Behance		Twitter	
		REP	REG	REP	REG
GPOP	GPOP in G^*	6.95	10.99	11.86	15.14
	GPOP in G	6.92	11.21	12.14	16.73
	GPOP in $G^{\mathcal{S}}$	6.95	10.99	12.04	15.47
Other baselines for G^*	GPOP-NoTop	7.81	12.57	12.57	33.38
	GPOP-NoNorm	7.37	12.15	11.78	16.7
	Group-Avg	7.48	11.44	12.05	15.73
	GroupSep	12.15	68.13	12.26	15.29
	GPOP-User	(*)	(*)	324.68	297.44
	CMTF	(*)	(*)	13343	20787
	TriMine	25.00	13.28	12.65	23.10
	CP-wopt	73.47	53.92	19.51	23.26
	ARIMA+COMB	9.52	16.70	34.72	37.54
	ARIMA+BU	9.36	16.05	33.15	35.96
	ETS+COMB	9.14	16.13	25.42	30.28
	ETS+BU	8.44	15.60	24.22	29.60
	MRBF	27.04	-	24.50	-

Table 4: Prediction errors (%) as $q - t_1$ and k vary for Behance

$q - t_1$	6	12	18	24	36	42
REP	3.91	6.95	9.49	11.62	15.09	16.7
REG	7.07	10.99	13.8	16.03	19.03	20.45
k	1	5	10	15	30	50
REP	7.75	7.07	6.95	6.91	6.94	6.97
REG	11.46	11.04	10.99	11.01	11.12	11.08

and dense for our data. Our results show that CP-wopt is much less stable than coupled tensor decomposition, and often gets stuck at sub-optimal solutions, causing high errors.

Hierarchical time series prediction: We test two classic time series prediction approaches: ARIMA, and Exponential Smoothing (ETS). We also use two different ways of combining their predictions hierarchically, i.e., bottom-up and optimal combination [13], leading to 4 baselines for hierarchical time series prediction (see Table 2). Here each time series corresponds to a user group. As shown in Table 3, GPOP outperforms all these 4 baselines significantly.

Population-level prediction only: MBRF [27] predicts the population-level popularity of Youtube videos by learning a multivariate linear model. Clearly, GPOP is superior to MBRF, both in terms of accuracy (Table 3) and the details of popularity at group level.

Future length and k: Table 4 shows the prediction errors of GPOP for Behance as the future periods $[t_1, q]$ and k vary ($t_1 = 18$). Clearly, the farther the future is, the harder it is to predict, leading to higher errors. As k increases, accuracy initially increases, but when k is too high, useless information is incorporated and increases the error.

5.4.3 *Qualitative Performance*

Fig. 9 shows the predictions of the next 7 days given the observations from the first 3 days for 4 example projects in Behance. As can be seen, GPOP makes good predictions for a variety of cases: typical projects p_1 and p_2 that are popular mostly in one user group; a project p_3 that are popular across all groups, possibly due to an unforeseen drift of users' interests w.r.t. the training data; and a project p_4 that ceased being popular after t_1.

5.5 Running time

Fig. 10 further shows that the average running time of GPOP (clustering and predicting) is linear in l, m, n, and k, making our solution scalable. On average, GPOP took 1.58 seconds for Behance, and 1.53 seconds for Twitter to predict one content. CMTF took more than 4 hours to finish one decomposition for Twitter, and did not finish after 1 day for Behance. TriMine finished predicting all content within 5 minutes but with much worse accuracy compared to GPOP.

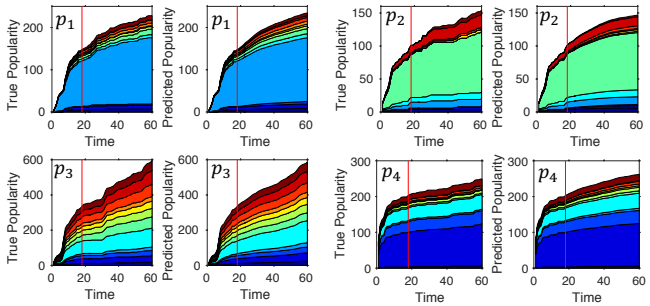


Figure 9: GPOP predictions for 4 example projects p_1-p_4 in Behance at time $t_1 = 18$ (3^{rd} day, marked by the vertical red lines).

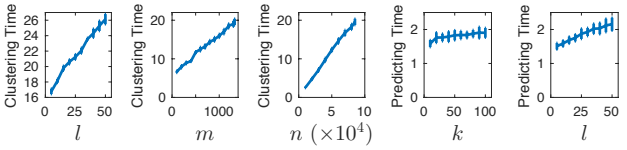


Figure 10: Running time for Behance (seconds) as l, m, n, k vary.

6. RELATED WORK

Popularity prediction: Since predicting the popularity of online content before publication is prone to large errors [6, 34], most earlier works focus on predictions after publication. Among these works, many papers simply use linear (or log-linear) regression to predict the aggregate (population-level) popularity of different types of contents [15,18,27,29–32], which often produces large errors [29, 33]. Thus, some papers adopt a classification approach to obtain higher accuracy at the loss of details: predict the range of popularity instead of the exact count [11, 14]. [20] uses the average of top-k similar tweets to predict a new tweet in Twitter, while [10] performs hierarchical prediction with ARMA model, obtaining good short-term but poor long-term predictions. [26, 28] combine data from different domains (websites) for prediction. Instead of treating user equally, several works also model behaviors of individual users (user-level popularity). For example, [22, 36] model users' behaviors to classify if a content would become popular; [38] proposes a probabilistic model based on Bayesian inference to predict the popularity of Twitter messages; [37] uses survival theory to predict the progression of an information cascade. These methods perform well for classification tasks but create large error for popularity count. We instead hierarchically predict popularity at group level, which is more fine-grained than the aggregate network level while less noisy than the individual user level.

Group-level information cascades: [9] and [39] solve the influence maximization and immunization problems for predefined groups respectively. [12] extracts community-level diffusion of retweets on the Weibo network but does not focus on predicting the future. We instead design the groups specifically for the task of predicting their future while also gaining insights into the group-level spread of information.

Time series modelling: Auto-regression and SIRS models have been added to tensor decomposition to model [24] and predict [23] time series (TriMine). Please see [13] for a survey of hierarchical time series prediction where predictions at different levels are combined in different ways.

7. CONCLUSION

In this paper, we developed a novel framework that addresses the important problem of online content prediction from a group-level popularity perspective. Our framework consists of two steps that first group users into clusters and then predict content popularity via a novel constrained tensor decomposition technique. Both network topology and interaction activities among users are exploited to learn a set of user clusters. Such a clustering solution is imposed as the hierarchical constraint in the PARAFAC tensor decomposition and we showed that optimizing its constrained function via gradient descent achieves faster convergence and leads to better prediction accuracy. Extensive empirical results demonstrate the effectiveness of our framework against eight baseline methods not only in terms of effectiveness but also of prediction accuracy, thus providing a better understanding about the spread of online content over social networks.

8. REFERENCES

[1] Behance.net social network. https://www.behance.net/.

[2] Hierarchical group-level popularity prediction for online contents. http://cs.ucsb.edu/~mhoang/gpop_longversion.pdf.

[3] E. Acar, D. M. Dunlavy, T. G. Kolda, and M. Mørup. Scalable tensor factorizations for incomplete data. *Chemometrics and Intelligent Laboratory Systems*, 106(1):41–56, 2011.

[4] E. Acar, T. G. Kolda, and D. M. Dunlavy. All-at-once optimization for coupled matrix and tensor factorizations. *arXiv preprint arXiv:1105.3422*, 2011.

[5] B. W. Bader, T. G. Kolda, et al. Matlab tensor toolbox version 2.6. http://www.sandia.gov/~tgkolda/TensorToolbox/, 2015.

[6] R. Bandari, S. Asur, and B. A. Huberman. The pulse of news in social media: Forecasting popularity. *arXiv preprint arXiv:1202.0332*, 2012.

[7] J. Cheng, L. Adamic, P. A. Dow, J. M. Kleinberg, and J. Leskovec. Can cascades be predicted? In *WWW*, 2014.

[8] D. M. Dunlavy, T. G. Kolda, and E. Acar. Poblano v1.0: A matlab toolbox for gradient-based optimization. Technical Report SAND2010-1422, Sandia National Laboratories, 2010.

[9] M. Eftekhar, Y. Ganjali, and N. Koudas. Information cascade at group scale. In *KDD*, 2013.

[10] G. Gürsun, M. Crovella, and I. Matta. Describing and forecasting video access patterns. In *INFOCOM*, 2011.

[11] L. Hong, O. Dan, and B. D. Davison. Predicting popular messages in twitter. In *WWW Companion*, 2011.

[12] Z. Hu, J. Yao, B. Cui, and E. Xing. Community level diffusion extraction. In *SIGMOD*, 2015.

[13] R. J. Hyndman, R. A. Ahmed, G. Athanasopoulos, and H. L. Shang. Optimal combination forecasts for hierarchical time series. *Computational Statistics & Data Analysis*, 55(9):2579–2589, 2011.

[14] S. Jamali and H. Rangwala. Digging digg: Comment mining, popularity prediction, and social network analysis. In *International Conference on Web Information Systems and Mining (WISM)*, 2009.

[15] A. Kaltenbrunner, V. Gomez, and V. Lopez. Description and prediction of slashdot activity. In *Web Conference, 2007. LA-WEB 2007. Latin American*, pages 57–66. IEEE, 2007.

[16] G. Karypis and V. Kumar. Metis: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices.

[17] G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed computing*, 48(1):96–129, 1998.

[18] S.-D. Kim, S.-H. Kim, and H.-G. Cho. Predicting the virtual temperature of web-blog articles as a measurement tool for online popularity. In *International Conference on Computer and Information Technology (CIT)*, 2011.

[19] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[20] S. Kong, F. Ye, and L. Feng. Predicting future retweet counts in a microblog. *Journal of Computational Information Systems*, 10(4):1393–1404, 2014.

[21] H. Kwak, C. Lee, H. Park, and S. Moon. What is twitter, a social network or a news media? In *WWW*, 2010.

[22] K. Lerman and T. Hogg. Using a model of social dynamics to predict popularity of news. In *WWW*, 2010.

[23] Y. Matsubara, Y. Sakurai, et al. Fast mining and forecasting of complex time-stamped events. In *KDD*, 2012.

[24] Y. Matsubara, Y. Sakurai, W. G. Van Panhuis, and C. Faloutsos. Funnel: automatic mining of spatially coevolving epidemics. In *KDD*, 2014.

[25] G. W. Milligan and M. C. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2):159–179, 1985.

[26] A. Oghina et al. Predicting IMDB movie ratings using social media. In *European Conference on Information Retrieval*, pages 503–507. Springer, 2012.

[27] H. Pinto, J. M. Almeida, and M. A. Gonçalves. Using early view patterns to predict the popularity of youtube videos. In *WSDM*, 2013.

[28] S. D. Roy, T. Mei, W. Zeng, and S. Li. Towards cross-domain learning for social video popularity prediction. *IEEE Transactions on multimedia*, 15(6):1255–1267, 2013.

[29] G. Szabo and B. A. Huberman. Predicting the popularity of online content. *Communications of the ACM*, 53(8):80–88, 2010.

[30] A. Tatar, P. Antoniadis, M. D. De Amorim, and S. Fdida. Ranking news articles based on popularity prediction. In *ASONAM*, 2012.

[31] A. Tatar et al. Predicting the popularity of online articles based on user comments. In *International Conference on Web Intelligence, Mining and Semantics*, 2011.

[32] A. Tatar et al. From popularity prediction to ranking online news. *Social Network Analysis and Mining*, 4(1):1–12, 2014.

[33] A. Tatar et al. A survey on predicting the popularity of web content. *Journal of Internet Services and Applications*, 5(1):1–20, 2014.

[34] M. Tsagkias et al. Predicting the volume of comments on online news stories. In *CIKM*, 2009.

[35] J. Yang and J. Leskovec. Patterns of temporal variation in online media. In *WSDM*, 2011.

[36] P. Yin, P. Luo, M. Wang, and W.-C. Lee. A straw shows which way the wind blows: ranking potentially popular items from early votes. In *WSDM*, 2012.

[37] L. Yu et al. From micro to macro: Uncovering and predicting information cascading process with behavioral dynamics. In *ICDM*, 2015.

[38] T. Zaman, E. B. Fox, E. T. Bradlow, et al. A bayesian approach for predicting the popularity of tweets. *The Annals of Applied Statistics*, 8(3):1583–1611, 2014.

[39] Y. Zhang et al. Controlling propagation at group scale on networks. In *ICDM*, 2015.