ReTrend: An Information Diffusion Model based on Retweet-tree encoding and Matrix Factorization

Zhenhao Cao & Ru Wang
School of Electronic Information and Electrical Engineering
Shanghai Jiao Tong University
Shanghai, China
Hazelnut@sjtu.edu.cn
williamwang15@gmail.com

Abstract

Social network has become a powerful communication tool which people around the world rely on. Patterns of information diffusion on social networks like Twitter and Facebook turn out to be more flexible but also more traceable, which shows great value to both researchers and practitioners. In this paper, we proposed a **Re**tweet-**Tr**ee **en**co**d**ing based matrix factorization method (ReTrend) to model the information diffusion process at a microscopic level. Specifically, this method borrows the idea of collaborative filtering for retweet behavior prediction. Our work differs from extant studies by avoiding onerous feature extraction and engineering, while overcoming the problem of insufficient information leverage. Substantial experiments on a real-world dataset (Twitter based) show improvements of our proposed ReTrend framework over the state-of-the-art methods.

1 Introduction

Social networks have provided an unprecedentedly flexible platform for information diffusion. Studies on social networks have drawn a profusion of academic attention, ranging from real-time event detection to text-based sentiment analysis. All the contents people read, watch, share or repost on social networks are information entities, and the systematic operation of social network essentially depends on the latent pattern of information diffusion. The mastery of this pattern will bring immense value to people. For example, advertisers hope to spread their commodity information better, while news media would also love to manipulate this pattern to maximize their visibilities. Thus, insights into the pattern of information diffusion will benefit both academia and industry.

Existing studies on information diffusion can be mainly classified into three types by methodology, namely, feature-based methods, time-series modeling and collaborative filtering methods. Feature-based methods either focus on the selection of different feature sets or the construction of hyper-features which thereafter get embedded into machine learning frames. Time-series modeling concentrate on the essence of information diffusion and aim to predict cascade size of anytime by stochastic point process modeling. In this work, we focus on collaborative filtering approaches for some of its improvable traits.

Collaborative filtering was initially proposed to address recommendation system problems. Stimulated by the Netflix Prize, this method has been proved to be greatly successful in recommendation system by modeling users rating or adoption behaviors. The key idea underlying entails the use of two sociological concepts, homophily and influence, among individual users or communities. Enlightened by this, researchers soon found the applicability of CF in the context of microscopic information diffusion modeling to predict the diffusion of an information entity (i.e., post) is to predict whether a user would adopt (retweet) it, and the reason for the adoption of a post can be comparable to that of a commodity.

However, extant work on collaborative filtering methods are facing some universal problems. One of them is that CF-based methods normally suffer from insufficient information leverage. The most basic idea of collaborative filtering is to carry our predictions based on users historical behaviors, which are statically recorded in a matrix. To leverage more auxiliary information, researchers tried to introduce social relationships and even item content features. However, all these works fail to find the latent bottleneck users historical behaviors are originally a permutation including temporal information (retweet sequence), spatial information (retweet topology), user-related information (user latent features) and content-related information (post latent features), however, current studies simply treat all the above information as a retweet matrix which is basically a flat snapshot, losing much of the necessary information.

In this work, we proposed a **Re**tweet-**Tr**ee **en**co**d**ing based matrix factorization method (ReTrend) to model the information diffusion process at a microscopic level. We introduce four matrices and cover almost all information of an observable retweet-tree. The four matrices are inherently an encoding of the behavior permutation, and the training process can be deemed as a dynamic inference process of the most likely final retweet-tree, based on currently observable data.

The main contributions of this work are as below:

1. We present a matrix factorization architecture ReTrend to almost-best leverage the permutation of users' historical behaviors. ReTrend covers almost all information of an observable retweet-tree, and play a role of an encoder of a behavior permutation. The training process can be deemed as a dynamic inference process of the most likely final retweet-tree, based on currently observable data.
2. To the best of our knowledge, this is the first work that casts insight into the relationship between Collaborative Filtering and a retweet-tree, namely, the essence of an information diffusion process. Specifically, we bridge Matrix Factorization with traditional tree-based analysis.
3. We introduce a novel concept of "user resistance" over a latent feature space. Apart from the Resistance Matrix, we also introduce other matrices to jointly model information diffusion and to reach a better performance.
4. We utilize a multi-layer perceptron for resistance inference to endow ReTrend with a high level of non-linearities and analytically show that MLP can be used as a non-linear transformation module in ReTrend for other usage.

2 Related Work

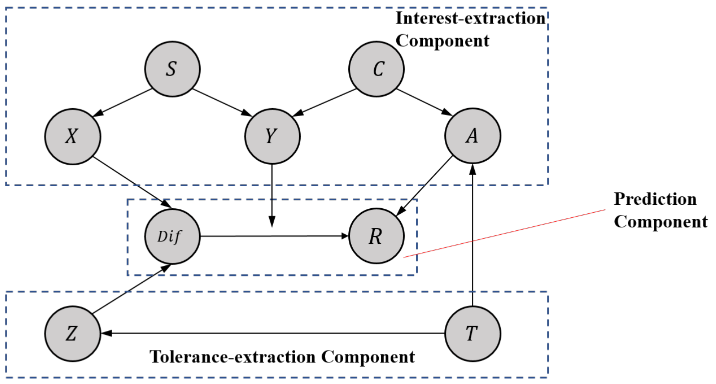
Many studies have casted insights into retweet prediction using CF. Li et al. (2015) takes the relevance of users interests, tweets content, and publishers influence into account simultaneously and uses SVD to track out the implicit factors. Jiang et al. (2015) carries out Matrix Factorization based on tweet content and introduces message clustering as a regularization factor into their Centroid-based Regularization Prediction Model (CRPM). Hoang et al. (2017) extends user-post matrix into a 3-order tensor by introducing temporal dimension and carries out tensor decomposition to predict adoption number at any given time. Numerous studies have focused on CF-based cascade prediction and we will present a few to illustrate our motivation.

SoRec Ma et al. (2008) assumes that the users should share the same user preference vector ui in the repost space and the social space, which is to say that the repost matrix and the social matrix should be coherent. SERec Wang et al. (2017) assume that people get information of items from their online friends and they do not have to share similar preferences, which is less restrictive and seems closer to reality. Instead, they utilize social information to capture user exposures rather than user preferences. HF-NMF Cui et al. (2011) jointly incorporates social factors and content factors for recommendation. It believes that the product of the latent user matrix and the latent item matrix should be the item-level social influence instead of the repost matrix directly. They construct an user-user influence matrix and use LDA to obtain an item-topic matrix.

3 Proposed Framework

We have noticed a common phenomenon in social networks. Nowadays, people are not so easy to be convinced into reposting. Some people may get interested in an article or tweet, still he/she

Figure 1: An overview of ReTrend



simply 'likes' the content without reposting. Thus, he/she makes no contribution to the diffusion of this information. We consider this intention 'to be convinced' to repost as an inherent attribute of user.

We also note that this attribute seems vary dramatically among users. Some people refuse to repost a content even though he is extremely intrigued, while some others repost a tweet immediately once this tweet entertains him/her a bit. In this work, we name this inherent attribute 'resistance', which varies over latent space but remains fixed for a fixed user. We deem that retweet behavior consists of two stages in a users view: First he/she decides whether to get interested; If he/she has no interests in some content then no retweet behavior will be triggered; If he/she is interested in the content, then he/she decides whether to retweet it. The second step is where the 'resistance' plays a role.

We defined following matrices:

- Subscription Matrix: user-user, $S_{ij} = 1$ when user i subscribes user j, otherwise $S_{ij} = 0$.
- Contagion Matrix: user-item, C_{ij} = number of reposts triggered by user i in his locality cardinality.
- Retweet Matrix: user-item, $R_{ui} = 1$ if user u retweet post i.
- Resistance Matrix: user-item, T_{ui} = proportion of user u's friends who retweet the post, if u DID NOT retweet i; otherwise $T_{ui} = 0$.

We also defined some factor matrices:

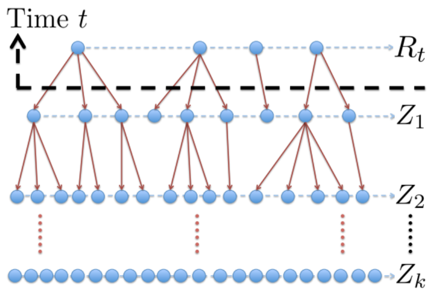
- Interest Matrix (X): User latent matrix w.r.t. interest.
- Item-Level Resistance Matrix (Z): User latent matrix w.r.t. resistance.
- Influence Matrix (Y): User latent matrix w.r.t. his social influence.
- Attraction Matrix (A): Item latent matrix w.r.t. its attraction.

3.1 ReTrend

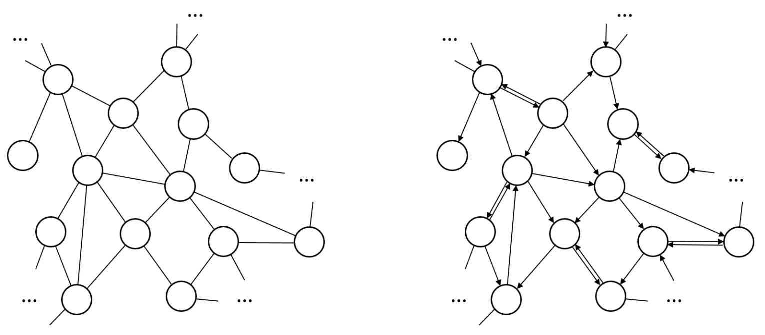
3.1.1 Encoding of Retweet-tree

The diffusion process of a post is often intuitively represented as a tree-based structure as is shown in Fig.2. The basic assumption behind a retweet-tree model is that a user's retweet behavior is triggered solely by one other user. This is reasonable because a user who finally decides to retweet a post necessarily 'choose' a previous retweeter and retweet from him/her. One can say that retweet tree is a most intuitive way to represent an information diffusion process with all the relevant information included: permutation, topologies, user and the specified post which diffuses over the tree. Note that retweet-tree is essentially a subgraph extracted from a network, namely, user relationship network N. In this sense, information diffusion process can be seen as a stochastic process on the whole network N where nodes on the retweet-tree randomly trigger their one-hop neighbors to be assimilated and added into the retweet tree.

Figure 2: Tree-based Retweet Model Zhao et al. (2015)



In this work, we define a user network as a directed graph rather than an undirected one, because prevalent social networks like Twitter and Facebook are all one-way networks, and an undirected graph can be basically regarded as a special case of directed graph. Part of a user network are shown in Fig.3.1.1.



(a) Part of a one-way user network (b) Part of a two-way user network

Figure 3: Two types of user network

To be clear, lets introduce colors to specify different meanings. A blue edge means a subscribing relationship. A red edge means the parent node succeeds in triggering the child node to retweet. A black edge means the parent node fails to make the child node retweet anyway. Note that what we really focus on is the retweet-tree. To avoid ambiguity, we name the edge in user network (i.e., blue ones) “fake edges”, name the edge in a retweet-tree (i.e., red ones) “branches”, and name the edge between a retweeter and a user who refuse to retweet (i.e., black ones) a “dead edge”.

An graphic illustration is shown in Fig.3.1.1. Initially, all edges are fake edges. In a), a user retweet a post. We say he/she is assimilated, or added into a retweet-tree. In b), we know that two child nodes are theoretically exposed to the post. One of them is assimilated, while the other has not seen it yet. Thus, one fake edge turns into a branch, while the other one remains a fake edge. In c), another three child nodes are exposed to the post. Two of them are assimilated. The other one does see it but refuse to retweet. Maybe he is not interested. Thus, two fake edges turn into branches, while the other one turns into a dead edge. d) shows a possible state after some time. At this moment, the subgraphs involved with red paths constitute a retweet-tree.

Now we show how our proposed ReTrend encodes the observable retweet-tree using subscription matrix, contagion matrix, resistance matrix and retweet matrix. We denote the retweet-tree for a post p at observing time τ is denoted as Rt^p_τ. If a user u shows up in Rt^p_τ, that means u retweeted the post p before the observing time. Suppose u is a subscriber or follower of user v and retweet p right from v. Then in Rt^p_τ, v is u's parent node.

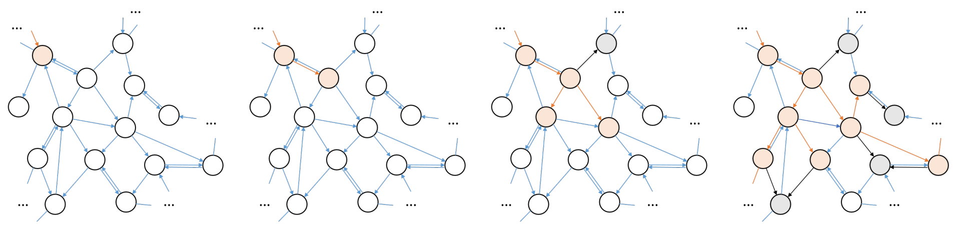
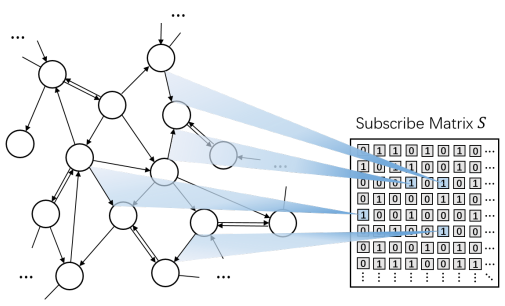

(a) a) (b) b) (c) c) (d) d)

Figure 4: A generative process to clarify the definition of edge types in this work.

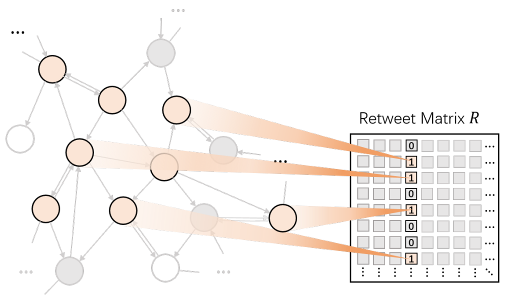
Subscribe Matrix S For a partially observable retweet-tree Rt^p_τ, Subscribe Matrix S provides the information of possible parent-child relationship, namely, the whole user network N. As the diffusion process goes on, some edges e in $\{N/Rt^p_\tau\}$ might be added into $Rt^p_{\tau'}$. In other words, Subscription Matrix S informs of all possible edges in graph, from which some of them might become branches in a retweet-tree.

Figure 5: Projection of a Subscribe Matrix onto a retweet-tree



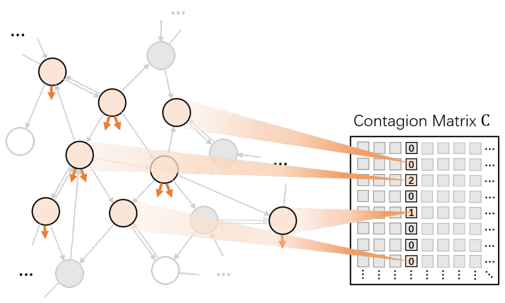
Retweet Matrix R Binary matrix R represents the snapshot of users' retweet behaviors up to the observing time τ. If the value of $R_{u,i}$ is 1, then u must be in Rt^p_τ and have an edge between herself and another user, who must be one of the users she subscribed and also retweeted i. Hence, R informs us which nodes are currently in Rt^p_τ and which are not.

Figure 6: Projection of a Retweet Matrix onto a retweet-tree



Contagion Matrix C The size of Contagion Matrix C is $U * I$ and the entry $C_{u,i}$ represents the retweet number of post i triggered directly by u among his friends. Clearly enough, for a partially observable retweet-tree Rt^p_τ, matrix C provides intuitively represent the our-degrees of a involved node. Note that Contagion Matrix C does not cover the information provided by Retweet Matrix R because leaf nodes in the retweet-tree have 0 outdegrees, making them undiscernible from unassimilated nodes without matrix R.

Figure 7: Projection of a Contagion Matrix onto a retweet-tree



With S, C and R, we can put it this way: given a whole network N, a set of nodes U_i, the out-degrees of all nodes in U_i, and the parent-child relationships of nodes in U_i, it can be proven that an one and only subtree can be determined with a high possibility. The proof is still under refinement and will be given as future work. This means that we actually achieve an approximate encoding of the observable retweet-tree, i.e. the user behavior permutation. To the best of our knowledge, our work is the first study to leverage users' historical behavior to such a degree and clearly bridge the relationship between Matrix Factorization based Collaborative Filtering and a Retweet-tree, which is the essence of information diffusion.

3.1.2 Matrix Factorization

This part first gives an intuitive illustration of the Matrix Factorization process in ReTrend, and then formulate the model with maths.

Factorization of Subscribe Matrix S We deem that a user choose to subscribe another user because he/she is interested in the authors content; and a user gets subscribed/followed because of his/her social influence. So by performing MF on user-user subscriber matrix (S), we obtain users interest latent matrix (X) and users' influence matrix (Y). In other words, the more a user u is interested, and the more a user v is influential, then the more likely that u chooses to subscribe v, that is, the entry $S_{u,v}$ approaches to 1. Note that S is not a symmetric matrix in our context. Assuming a Gaussian observation noise, we have

$$S_{u,v} \sim \mathcal{N}(X_u^T Y_v, \sigma_S^2)$$

where σ_S^2 represents the variation in case with S.

Factorization of Contagion Matrix C Every entry in Contagion Matrix (C) reflects two facts: to what degree a user can trigger his friends retweet an item, and how attractive an item is. The more influential is the user, the corresponding value will get larger; similarly, the more attractive the item is, the value of the entry gets larger. So we decompose (C) into users influence matrix (Y) and items attraction matrix (A). Assuming a Gaussian observation noise, we have

$$C_{u,i} \sim \mathcal{N}(Y_u^T A_i, \sigma_C^2)$$

where σ_C^2 represents the variation in case with C.

Factorization of Resistance Matrix T Every entry in Resistance Matrix (T) reflects two facts: to what degree a user resists the item, and again how attractive an item is. A larger value means more friends retweeting the post while this user REFUSE to retweet, so this reflects the resistance degree, or his tolerance towards something interesting. With respect to items, its conspicuous that the more attractive the item is, the value of the entry gets larger. So we decompose (T) into users 'resistance matrix (Z) and items' attraction matrix (A). Similarly, we have

$$T_{u,i} \sim \mathcal{N}(Z_u^T A_i, \sigma_T^2)$$

Factorization of Retweet Matrix R Every entry $R_{u,i}$ in Retweet Matrix R directly represents a retweet behavior of user u towards post i. We deem that u's interests, resistance, the attraction of i and the peer-to-peer influence of the previous retweeter (who exposes u to i, denoted by $par(u)$) all contribute to the triggering of u's retweet behavior, namely $R_{i,j}$. We use a combination of X_u, Z_u and $Y_{par(u)}$ to jointly represent the synthetic latent feature vector at a user-level. Denote the combination as function $g(\cdot)$. Assuming a Gaussian observation noise, we have

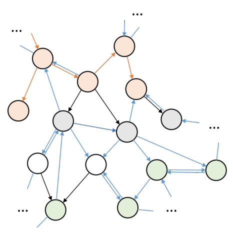
$$R_{u,i} \sim \mathcal{N}(g(X_u, Z_u, Y_{par(u)})^T A_i, \sigma_R^2)$$

Note that $par(u)$ represents a set of all retweeters whom u subuscribe. In this work, we define function g as below

$$g_{uk}(X_u, Z_u, Y_{par(u)}) = \begin{cases} \max\{X_{uk} - Z_{uk}, \max_{v \in paru}\{v_{uk}\}\}, & par(u) \neq \varnothing \\ 0, & par(u) = \varnothing \end{cases}$$

This means that when none of the users who u subscribe has retweeted i, u has no chance to see it, and thus is unlikely to retweet. It is mentionable that most extant work based on CF ingores the issue of "exposure", which means one could only behave on seeing the item or post first. It might happen that a community of users who could be highly interested into a post i never get a chance to see it, and therefrom, retweet it, as is shown in Fig.8. Essentially, the information diffusion is blocked.

Figure 8: Information diffusion might be blocked



Entire Matrix Factorization Model The factor graph of ReTrend based on matrix factorization is shown in Fig.1. We can now obtain the conditional distribution over all observed data as

$$p(S, C, T, R | X, Y, Z, A, \boldsymbol{\sigma}) = \prod_{u=1}^{N} \prod_{v=1}^{N} \left[\mathcal{N}(S_{u,v} | X_u^T Y_v, \sigma_S^2)\right]^{I_{u,v}^S}$$

$$* \prod_{u=1}^{N} \prod_{i=1}^{M} \left[\mathcal{N}(R_{u,i} | g(X_u, Z_u, Y_{par(u)})^T A_i, \sigma_R^2)\right]^{I_{u,i}^R} \left[\mathcal{N}(C_{u,i} | Y_u^T A_i, \sigma_C^2)\right]^{I_{u,i}^C} \left[\mathcal{N}(T_{u,i} | Z_u^T A_i, \sigma_T^2)\right]^{I_{u,i}^T}$$

We also place zero-mean spherical Gaussian priors on all latent feature vectors of users and posts, which are

$$p(X | \sigma_X^2) = \prod_{u=1}^{N} \mathcal{N}(X_u | 0, \sigma_X^2)$$

$$p(Y | \sigma_Y^2) = \prod_{u=1}^{N} \mathcal{N}(Y_u | 0, \sigma_Y^2)$$

$$p(Z | \sigma_Z^2) = \prod_{u=1}^{N} \mathcal{N}(Z_u | 0, \sigma_Z^2)$$

$$p(A | \sigma_A^2) = \prod_{i=1}^{M} \mathcal{N}(A_i | 0, \sigma_A^2)$$

Then we obtain the log of the posterior distribution over latent features, which is the raw object function of ReTrend. By modifying the log-likelihood, we obtain the loss function as

$$\min_{X,Y,Z,A} J(S,C,T,R,X,Y,Z,A) = \|R - g(X,Z,Y)^T A\|^2 + \lambda_S \|S - X^T Y\|^2 + \lambda_C \|C - Y^T A\|^2 + \lambda_T \|T - Z^T A\|^2 + \lambda_X \|X\|^2 + \lambda_Y \|Y\|^2 + \lambda_Z \|Z\|^2 + \lambda_A \|A\|^2$$

where $\| \cdot \|^2$ represents the Frobenius norm, and $\lambda_{Mtx} = \frac{\sigma_R^2}{\sigma_{Mtx}^2}$, $Mtx = S, C, T \cdots$.

3.1.3 Dynamic Inference of Retweet-tree Structure

At the initial state, the matrices only store and encode the observable data up till the observation time τ. Our purpose in this work is to carry out predictions on both the information diffusion process and the final cascade size, which can be both illustrated by a closured retweet-tree. Note that only when node has a assimilated parent node, can this node be exposed to the post and get assimilated. We call this type nodes ”susceptible nodes”. Generally, a study on information diffusion requires a inference on whether a susceptible node will be finally assimilated. But this description can be actually ambiguous because a susceptible node may have more than one assimilated parent node. In this work, based on our definition of different edges, we refine the problem to the inference on whether an edge from a retweeter node to a susceptible node will finally become a branch in retweet-tree. As a dynamic inference, the learning process can be accomplished by Matrix Factorization.

Fig.9 enunciates why Matrix Factorization would work in inferring the tree structure. we only show four factor matrices, which represents users' latent features and items' latent features respectively. These factor matrices are obtained by operating Matrix Factorization on the four observable matrices and thus carry all necessary information. Inference of retweet-tree structure comes down to the inference of edge: whether a fake edge is going to switch into a branch or a dead edge, and inference of an edge involves user interest, user resistance, parent influence and post attraction, which are learned by MF and then leveraged in this work.

Figure 9: Inference of potential branches in Matrix Factorization

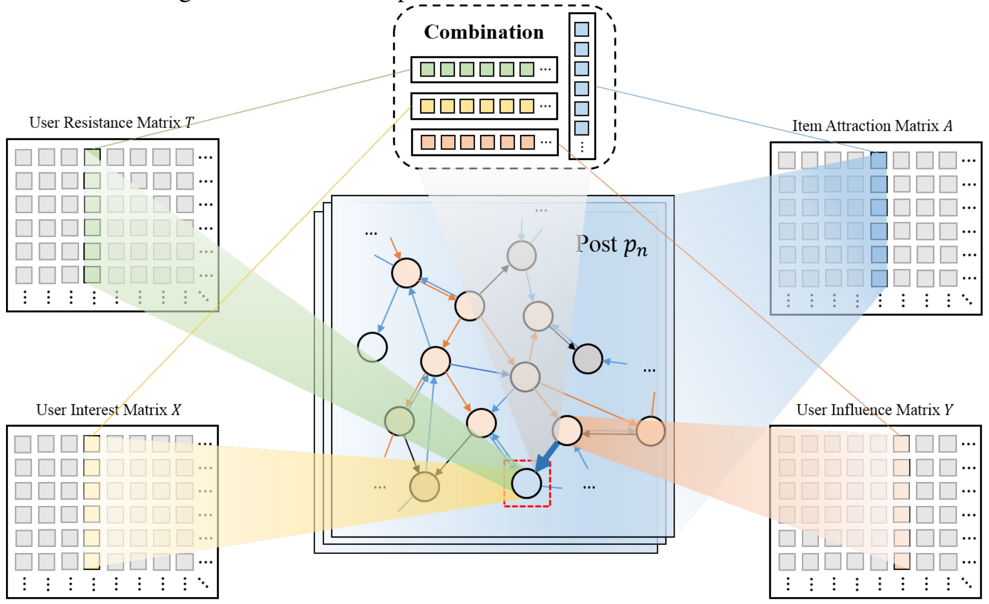
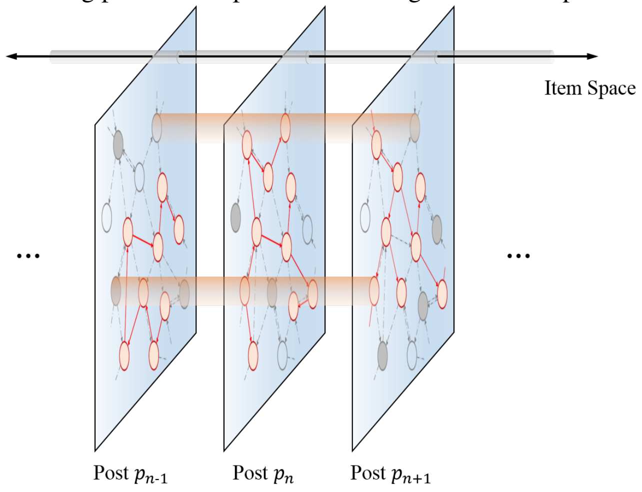

Combination
User Resistance Matrix T
Item Attraction Matrix A
Post p_n
User Interest Matrix X
User Influence Matrix Y

Fig.10 illustrates the training process with Matrix Factorization based on the matrices we construct. Note that a users all historical behaviors over the posts make contribution to the inference; information about users features transcends different retweet-trees through Matrix Factorization. More specifically, suppose user u has not retweeted post p_n by observation time τ, and now we hope to predict whether he/she is going to retweet p_n. Through MF, we can learn u's latent features (interests, resistance) based on his/her historical behaviors with other posts; learn his/her parents' influence in a similar way and the attraction features of the post p_n itself based on historical behaviors of user ensemble with post ensemble. Inherently, this is how homophily is leveraged using Collaborative Filtering based on MF.

Figure 10: Training process and post-transcending inference of potential branches



3.1.4 Convergence of ReTrend

For a real-world retweet-tree, there are basically two possible states. The first state is Unstable State, where there are still users who may or may not be assimilated. In other words, a retweet-tree in Unstable State is still extensive. The other state is Deterministic State, where all user states have been inferred already. In Deterministic State, there is no potential user left, and retweet-tree is not extensive anymore. We say a retweet-tree in Deterministic State is closured.

Fig.11(a) shows a retweet-tree in Unstable State, where yellow nodes are potential retweeters and the tree is still extensive. Fig.11(b) shows a retweet-tree in Deterministic State, while Fig.11(c) shows another situation. Note that there is a node without any color in Fig.11(c). This means the user never get a chance to get exposed to the post, that is, the diffusion is closured before passing it. Also note that the author of the post is in this subgraph.

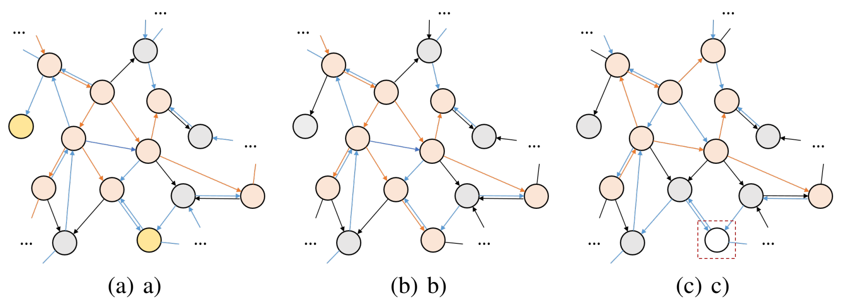


Figure 11: Unstable State (a) and Deterministic State (b)(c) of a retweet-tree

3.2 MLP for Resistance Inference

As is illustrated in He et al. (2017a), there are some possible limitations of MF caused by the use of a simple and fixed inner product to estimate complex user-item interactions in the low-dimensional latent space. Therefore, we address the limitation by learning the interaction function using DNN from data, and further implement it as a module in RBMF to enable higher level of non-linearilities in our model.

Moreover, The MF process of Resistance Matrix are constrained by many factors, which is practically difficult for our RBMF to optimize its parameters through learning the relationship between intermediate matrices. Thus for now, we only use MLP for Resistance Matrix, that is, replace the factorization of Resistance Matrix by a simple MLP module.

Figure 12: MLP module for Resistance Matrix

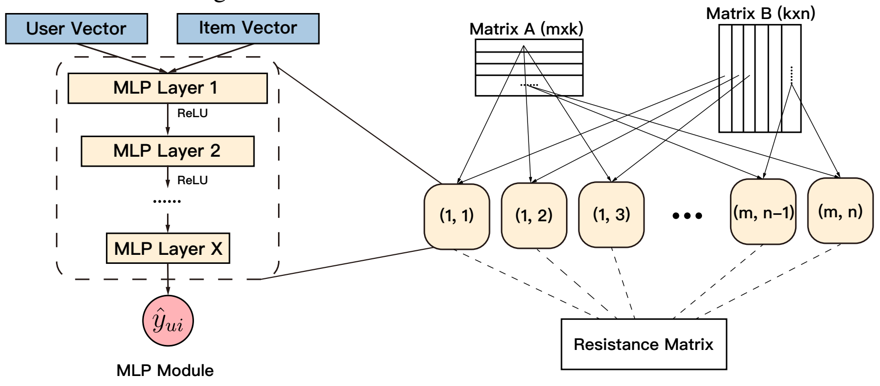


Figure 12 shows the above process. Suppose Resistance Matrix can be factorized as matrices $\mathbf{A}(\mathbf{m} \times \mathbf{k})$ and $\mathbf{B}(\mathbf{k} \times \mathbf{n})$, where each row of $\mathbf{A}$ is an intermediate vector of a particular user and each column of $\mathbf{B}$ is an intermediate vector of particular item. Therefore, each element of Resistance Matrix can be computed through the MLP module shown on the left.

We formulate the module as

$$\hat{y}_{ui} = \phi_{out}(\phi_X(...\phi_2(\phi_1(\mathbf{v_u}, \mathbf{v_i}))))$$

where $\mathbf{v_u}$, $\mathbf{v_i}$ represent intermediate user vector and item vector, respectively. ϕ_x are mapping functions for each layer. The output of MLP module $\hat{y}_{ui}$ is the predicted resistance value for user u and item i. And we use Mean Squared Error as loss function, which guarantees its parameters updated with the gradient of same scale as other parts of RBMF in the backward pass.

Since both $\mathbf{v_u}$ and $\mathbf{v_i}$ are encoded with collaborative information while training, it is possible that we can obtain a more accurate Resistance Matrix with MLP.

4 Experiments

4.1 Dataset

We use the dataset Twitter-Dynamic-Net[1] and Twitter-Dynamic-Action[2] in our experiments. The data including subscription and retweet behaviors have more than 90,000 users and 99,696,204 tweets related, which are so sparse and large that it is difficult to either train or test without preprocessing. Therefore, we select 10,000 most active users and 5,000 corresponding tweets that are most interacted with the users. And for Resistance Matrix inference, we select 1,246 users whose behavior are fully observed, together with 5,000 tweets stated above to validate the plausibility of our method.

[1]https://www.aminer.cn/data-sna#Twitter-Dynamic-Net
[2]https://www.aminer.cn/data-sna#Twitter-Dynamic-Action

4.2 Implementation Detail

4.2.1 Implementation of ReTrend

This part records and explains the necessary details of the implementation of ReTrend model to give a instructive guidance to both readers and our future work.

Matrix Storage The process of matrix factorization entails entry updating over all of the eight matrices (partially updating for the four observable matrices and fully updating for the four factor matrices). With the real-world data from Twitter, if we store the whole matrix with a size of $10000 * 5000$, then every epoch will involve 50 million updating with more than once matrix multiplication calculating the gradients. This would be an unconceivably time-consuming process. To support the SDG optimizing method with an randomized batch-data selection, we store the matrix in a tuple style $(user, post, value)$ where each entry will be represented by a line. Thus, the training process will be much time-saving applying SGD.

Indicator Matrices It mentionable that the negative values in observable matrix do face the risk of ambiguity, because a user might fail to adopt a post either for disinterest or just missing it. Default settings of ReTrend will deem all negative entries as disinterest and thus will introduce considerable errors. To handle this problem, we introduce indicator matrix to countervail the influence of entries.

Normalization Note that Retweet Matrix R and Subscribe Matrix S are binary while Contagion Matrix C and Resistance Matrix T are real-valued. Take C for example, the absolute value of $C_{u,i}$ represents the triggered retweet number. To compress the value into interval $[0, 1]$, the value should be normalized. A potential issue brought by this is that the distribution of this value over users comply to the heavy-tailed distribution, and thus most of the values will be close to 0 once normalized. To address this problem, we calculate log of matrix C then training with the log matrix.

4.2.2 Implementation of MLP

For MLP modules, we build our models with Keras. We train our model for 20 epochs with batchsize 256. To evaluate the performance of the inference process, we adopted the leave-one-out evaluation, which has been widely used in literature He et al. (2017a)Bayer et al. (2016) He et al. (2017b). For each user, we held-out his/her latest interaction as testing set and utilized the remaining data for training.

5 Results

5.1 Performance of ReTrend

In our experiments, we empirically set the feature number as 30. We first load four observable matrices data and randomize the four factor matrices. Thereafter, we training the whole model with 1000 epochs by SGD and output the final states of all the eight matrices. To predict the final cascade size, we calculate the column-sum of the corresponding post in Retweet Matrix R. Nonetheless, other matrices provide more insights into the information diffusion pattern.

We evaluate the following methods on the constructed dataset as our baseline:

- Random: The retweet prediction is a binary classification task. For each tweet, we randomly select a decision of retweet or not.
- Word Vector Based SVM: In this method, we sum all the word vectors as the feature vector of the tweet. Then we used these features to train a classifier (only evaluate in a subset of our dataset).
- Neural Collaborative Filter: Implement and generalize MF in neural network combined with MLP. The result is shown in Table 1

5.2 Performance of MLP

First of all, to validate the plausibility for Resistance inference using MLP module, we simply use 2 one-hot vector for User Vector and Item Vector followed by 2 embedding layers as input. The

Table 1: Performance of our method with baseline

Method	Precision	Recall
Random	0.264	0.485
SVM	0.382	0.649
NCF	0.782	0.821
ReTrend	0.763	0.837

Table 2: MLP for Resistance: One-hot Vectors As Input

Layer Settings	Accuracy-3	Accuracy-2
{128,64,32,16,8}	0.882	0.997
{64,32,16,8}	0.783	0.996

model converges quickly and the results are shown in Table 2.

Layer settings are the output size of each layer in MLP module. Acc-x denotes the ratio of predictions with absolute error less than 10^{-x}. Since resistance values are scalar ranging from 0 to 1, Accuracy-3 evaluates the performance of our model better than Accuracy-2, and higher Acc-3 indicates better performance.

We can see from the result that as we increase the length of embedding vectors and the depth of the network, the performance of our model is enhanced greatly. Since in this experiment, the only implicit information considered in the training phase is the influence of the tweets (represented by resistance of users), We can also conclude that MLP module has a better performance combined with RBMF due to collaborative subscription information in MF.

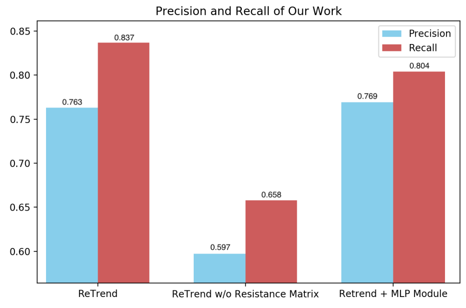
5.3 Performance of RBMF+MLP

We test 3 versions of our method,

- ReTrend: Original version of our method.
- ReTrend w/o Resistance Matrix: Remove the Resistance Matrix of our framework.
- ReTrend + MLP Module: Replace MF of Resistance Matrix with MLP module.

The results are shown in Figure 13

Figure 13: Performance of Our Method



6 Conclusion

Social networks have provided an unprecedentedly flexible platform for information diffusion, and the systematic operation of social network essentially depends on the latent pattern of information diffusion. The mastery of this pattern will bring immense value to people.

In this paper, we proposed a **Re**tweet-**Tr**ee **en**co**d**ing based matrix factorization method (ReTrend) to model the information diffusion process on a microscopic level. Our work differs from extent studies by avoiding onerous feature extraction and engineering, while overcoming the problem of insufficient information leverage. We contrive to encode all information of an observable retweet-tree and therefrom carry out matrix factorization to predict user retweet behavior. Then we combined Matrix Factorization and DNN (MLP) to achieve better performance. Substantial experiments on a real-world dataset (Twitter based) show improvements of our proposed ReTrend framework over the state-of-the-art methods.

However, The combined our work still needs further optimization in terms of the combination of matrix factorization and DNN(MLP). Moreover, a denser dataset of operable size is desired for our model since the implicit information are easily shadowed by noises in sparse dataset.

References

Immanuel Bayer, Xiangnan He, Bhargav Kanagal, and Steffen Rendle. A generic coordinate descent framework for learning from implicit feedback. *CoRR*, abs/1611.04666, 2016.

Peng Cui, Fei Wang, Shaowei Liu, Mingdong Ou, Shiqiang Yang, and Lifeng Sun. Who should share what?: item-level social influence prediction for users and posts ranking. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pp. 185–194. ACM, 2011.

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. *CoRR*, abs/1708.05031, 2017a.

Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. Fast matrix factorization for online recommendation with implicit feedback. *CoRR*, abs/1708.05024, 2017b.

Minh X Hoang, Xuan-Hong Dang, Xiang Wu, Zhenyu Yan, and Ambuj K Singh. Gpop: Scalable group-level popularity prediction for online content in social networks. In *Proceedings of the 26th International Conference on World Wide Web*, pp. 725–733. International World Wide Web Conferences Steering Committee, 2017.

Bo Jiang, Jiguang Liang, Ying Sha, and Lihong Wang. Message clustering based matrix factorization model for retweeting behavior prediction. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pp. 1843–1846. ACM, 2015.

Jun Li, Jiamin Qin, Tao Wang, Yi Cai, and Huaqing Min. A collaborative filtering model for personalized retweeting prediction. In *International Conference on Database Systems for Advanced Applications*, pp. 122–134. Springer, 2015.

Hao Ma, Haixuan Yang, Michael R Lyu, and Irwin King. Sorec: social recommendation using probabilistic matrix factorization. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pp. 931–940. ACM, 2008.

Menghan Wang, Xiaolin Zheng, Yang Yang, and Kun Zhang. Collaborative filtering with social exposure: A modular approach to social recommendation. *arXiv preprint arXiv:1711.11458*, 2017.

Qingyuan Zhao, Murat A Erdogdu, Hera Y He, Anand Rajaraman, and Jure Leskovec. Seismic: A self-exciting point process model for predicting tweet popularity. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1513–1522. ACM, 2015.