MoodCast: Emotion Prediction via Dynamic Continuous Factor Graph Model

Yuan Zhang[†§], Jie Tang[†§], Jimeng Sun[‡], Yiran Chen[†], and Jinghai Rao[♯]
[†]*Tsinghua National Laboratory for Information Science and Technology*
[§]*Department of Computer Science and Technology, Tsinghua University, China*
[‡]*IBM TJ Watson Research Center, USA* [♯]*Nokia Research Center Beijing*
{*fancyzy0526, mithich.chan*}*@gmail.com, jietang@tsinghua.edu.cn, jimeng@us.ibm.com, jinghai.rao@nokia.com*

***Abstract*—Human emotion is one important underlying force affecting and affected by the dynamics of social networks. An interesting question is "can we predict a person's mood based on his historic emotion log and his social network?". In this paper, we propose a MoodCast method based on a dynamic continuous factor graph model for modeling and predicting users' emotions in a social network. MoodCast incorporates users' dynamic status information (e.g., locations, activities, and attributes) and social influence from users' friends into a unified model. Based on the historical information (e.g., network structure and users' status from time 0 to $t-1$), MoodCast learns a discriminative model for predicting users' emotion status at time t. To the best of our knowledge, this work takes the first step in designing a principled model for emotion prediction in social networks. Our experimental results on both real social network and virtual web-based network show that we can accurately predict emotion status of more than 62% of users and 8+% improvement than the baseline methods.**

I. Introduction

We are living in an evolving social ecosystem and our emotion statuses are often influenced by a complex set of factors. For example, we may become happy because of watching a great movie, having delicious food, or having completed a difficult task; while we may also feel happy just because our friends feel happy. Christakis and Fowler [1] [2] qualitatively studied the problem of the spread of happiness in social networks. They found that within a social network, happiness spreads among people up to three degrees of separation, which means when you feel happy, everyone within three degrees to you has a higher likelihood to feel happy too.

In this work, we address an even harder problem: i.e., how to quantitatively predict one's emotion status. Given complex social dynamics, we focus on the following aspects of the social networks for doing the emotion prediction. First, *temporal correlation:* One's emotion status at current time is highly correlated to her emotion in the recent past. Figure 2(a) confirms this temporal correlation of the emotion status on a mobile social network (MSN) data set. Second, *social correlation:* One's emotion is correlated with the emotion status of her friends. Figure 2(b) confirms this network correlation of the emotion status on the LiveJournal data set. Third, *attribute correlation:* The environment and activities of a person can also affect her emotion status.

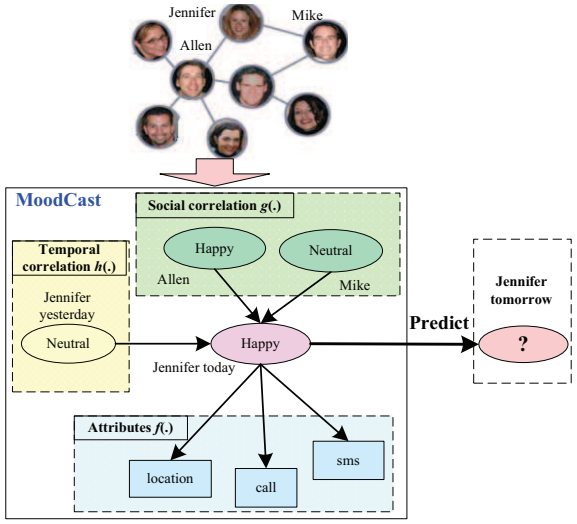


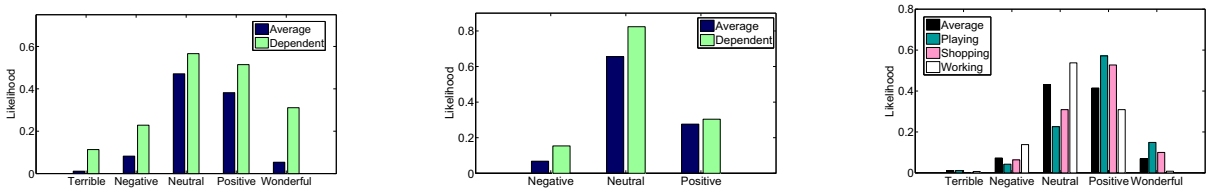
Figure 1. Model illustration of emotion prediction in a mobile social network.

Figure 2(c) shows that while playing and shopping, the person's emotion status more likely stays as "Positive" and "Wonderful". On the other hand, while working, the person's emotion status more likely stays as "Negative" or "Neutral".

By leveraging these aspects, we solve the problem of emotion prediction through a dynamic continuous factor graph model. Specifically, in this model, each user's attribute is modeled as a factor function and her friends' influences are modeled as an exponential decay function; while the user's emotional changes over time are modeled as a Markov chain. To learn the model, we design an approximate method MoodCast using Metropolis-Hastings sampling. We apply MoodCast to predict user's emotion for two real networks: one mobile social network and one online social network derived from LiveJournal.com. Experimental results on both data sets show that MoodCast can clearly improve the prediction accuracy against several baseline methods.

Figure 1 illustrates the process involved in the MoodCast model learning and prediction on the mobile network data set. The top figure shows Jennifer's social network and the bottom figure shows the proposed MoodCast model. In particular, for the mobile social network, we can use the mobile context data (e.g., GPS location, call logs, SMS

1550-4786/10 $26.00 © 2010 IEEE
DOI 10.1109/ICDM.2010.105



(a) Temporal correlation (b) Social correlation (c) Attribute correlation

Figure 2. (a) One's emotion status at previous time influences on her current mood. *Average* is the the likelihood of all users with a certain emotion and *Dependent* is the likelihood given that the user feels the same at previous time on the MSN data (statistics from the MSN data). (b) The friends' emotions influence on his current mood. *Average* is the the likelihood of all users with a certain emotion and *Dependent* is the likelihood given that the user's friends feel the same (statistics from the LiveJournal data). (c) The user's activities influence on his/her current mood (statistics from the MSN data).

text) as the attributes of each user. The model incorporates three different types of information including user's historic emotion status (temporal correlation), friends' emotion statuses (social correlation), and user attributes. The output of the learning phase is a predictive model, aka a dynamic continuous factor graph model in our case. The final step is to predict the emotion status of each user in the network.

II. Problem Definition

A static social network can be represented as $G = (V, E)$, where V is the set of $|V| = N$ users and $E \subset V \times V$ is the set of directed/undirected links between users. Given this, we can define the user's emotion status as follows.

Definition 1. ***Emotion****: A user v_i's emotion status at time t is represented as y_i^t. Let $\mathcal{Y}$ be the space of the emotion status. We can denote the historic log of every user's emotion status up to time t as $Y = \{y_i^t\}_{i,t}$, where $y_i^t \in \mathcal{Y}$.*

In general, the emotion status can be defined as a sequence of discrete events. For example, in the mobile social network, five different events are defined: "wonderful", "positive", "neutral", "negative", "terrible". Further, each user is associated with a number of attributes at a continuous time-scale and thus we have the following definition.

Definition 2. ***Continuous time-evolving attributes****: The continuous time-evolving attributes in the social network are defined as a set of historic attribute-value logs, i.e., X. Specifically, suppose each user has d attributes. When user v_i changes the value of her j-th attribute at time t, we add a three-dimensional element $(v_i, t, a_j) \rightarrow x_{ij}^t$ into the set X, where x_{ij}^t is the new value of the attribute a_j associated with user v_i at time t.*

Note that as the attribute of a user may change at any time, it is necessary to define the attribute changes at a continuous time-scale, rather than discrete timestamps. Figure 3 illustrates some examples of continuous time-evolving attributes of two users. The beginning of each color bar indicates the value change of a specific attribute.

Given this, we can define the input of our problem, a dynamic continuous network.

Definition 3. t***−Dynamic continuous network****: The dynamic continuous network (from time 0 to time t) is denoted as $G^t = (V, E^t, X^t, Y^t)$, where V is the set of users, $e_{ij}^t \in E^t$ is a link between users v_i and v_j created at time t, and X^t is the continuous time-evolving attributes of all users in the network, and Y^t represents the set of emotion status changes of all users in the social network.*

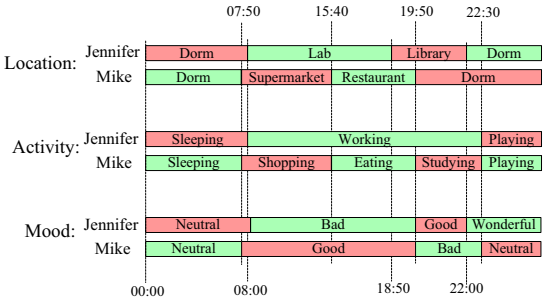


Figure 3. Example of dynamic attributes.

We use the superscript t to denote that the dynamic continuous information in the network G^t is up to time t, that is, all edges E, attribute changes X and emotion status changes Y are recorded until time t. Thus the learning task of our problem can be defined as:

Learning task: Given a dynamic continuous network G^t, the goal is to learn a predictive function f to predict the emotion status of users at a future time $(t + 1)$. Formally, we have $f(V, E^{(t+1)}, X^{(t+1)}|G^t) \rightarrow Y^{(t+1)}$.

To learn the emotion predictive model, there are several requirements. First, as the input is a dynamic continuous network, it is necessary to find a function to capture the continuous property. Second, the emotion status of each user is related to multiple factors, e.g., network structure, attribute changes, and users' historic emotion status, it is important to find a unified model which is able to consider all the information simultaneously. In existing works, Tan et al. [3] propose a graph model to predict user actions in the dynamic social network. Tang et al. [4] study how to quantify the social influence and Goyal et al. [5] investigate how to learn the influence from the history of users' actions. However, all the models do not consider user emotions. Christakis and Fowler [1], [2] study the problem of the spread of happiness in social networks. However, they only qualitatively test the spread of happiness on two small data sets. To the best of our knowledge, no previous work has been done for emotion prediction in the dynamic social network.

III. Our Approach

We formalize the problem of emotion prediction in a dynamic continuous factor graph model and propose an approach referred to as MoodCast to learn the model for predicting emotional status of individuals. Our basic idea is to define the correlations using different types of factor functions. An objective function is then defined by the joint probability of the factor functions, and the problem of emotion model learning is cast as learning the model parameters to maximize the objective function based on the input dynamic network. In summary, we define three kinds of factor functions:

- **Temporal correlation function** $h(y_i^{t'}, y_i^t)$,$t' < t$. It represents the dependency of one's emotion status at time t on his emotion at the recent past time t'.
- **Social correlation function** $g(y_i^t, y_j^{t'})$,$t' < t$. It represents the influence of user v_j's emotion at time t' on user v_i's emotion at time t.
- **Attribute correlation function** $\{f(x_{ik}^t, y_i^t)\}_k$. It denotes the influence of an attribute of v_i at time t.

The three factors can be instantiated in different ways, reflecting our prior knowledge for different applications. Here, we use the mobile social network as the example to explain how we define the factor functions. Based on the attributes associated with each user in the mobile social network, we define the following attribute factor functions:

Location: The feature represents the location of the user. We use GPS and GSM data to locate the user. The location is usually denoted as the longitude and the latitude value. To reduce noisy data, we only keep locations where the mobile user stays for more than 10 seconds.

SMS text: The feature represents whether or not a word is contained in the Short Message Service (SMS) text message sending to or received from one's friends.

Calling logs: The feature represents that the user makes (or receives) a call to his friend.

Activity: The features represents what the user is doing. There are eight predefined categories for the activities in the annotation system.

All the factor functions are time-dependent. For example, when there is a new call at time t, then a factor function is defined. All the attribute factor functions are converted into binary functions. For example, $f(x_{i1}^t = 1, y_i^t = \text{`}positive\text{'})$ represents if the user v_i's emotion status is "positive" at time t and the index of his GPS location is 1, then the feature value is 1, otherwise 0. Finally, for the historic attribute-value log X^t, we can accumulate all the factor functions and obtain a local entropy for all users:

$$\frac{1}{Z_1}\exp\{\sum_{v_i \in V}\sum_{x_{ik}^t \in X} \alpha_k f_k(x_{ik}^t, y_i^t)\} \tag{1}$$

where α_j is the weight of function f_j and Z_1 is a normalization factor.

For social correlation factor function, we define it based on pairwise network structure and the continuous-time information. That is, if user v_i and v_j has an relationship, then we define a social correlation factor function as follows:

$$g(y_i^t, y_j^{t'}) = e^{-\sigma_1(t-t')}\exp\{\beta_{ji}(y_i^t - y_j^{t'})^2\} \tag{2}$$

where t' is the latest past time when friend v_j changed her/his emotion (i.e., the latest record of emotion change in X^t); $e^{-\sigma_1(t-t')}$ is user-independent time-decay factor; σ_1 is a predefined parameter. In the model, we assume users' emotions at time t are conditionally independent of all the past states given the recent past statuses of friends' emotion. In addition, β_{ji} is the weight of the function, representing the influence degree of v_j on v_i. It can be easily seen that, without loss of generality, the parameter σ_1 can be absorbed into the learning process and combined with β_{ji}. So the above feature function can be rewritten as $g(y_i^t, y_j^{t'}) = \exp\{-\beta_{ji}(t-t')(y_i^t - y_j^{t'})^2\}$.

For temporal correlation factor function, we try to use it to model the decay of the user emotion based on his/her past status and define it as:

$$h(y_i^t, y_i^{t'}) = e^{-\sigma_2(t-t')}\exp\{\lambda_i(y_i^t - y_i^{t'})^2\} \tag{3}$$

where t' is the latest past time when the user v_i changed his emotion status; similarly, σ_2 is a predefined parameter; λ_i represents how likely user v_i changes her emotion. In reality, some users may easily change their emotion status while the emotion status of some other users may be more stable. Again, σ_2 can be absorbed and thus we have $h(y_i^{t'}, y_i^t) = \exp\{-\lambda_i(t-t')(y_i^t - y_i^{t'})^2\}$.

Finally, a factor graph model is constructed by combining Eqs. (1)-(3) together, i.e.,

$$\begin{aligned} p(Y|G^t) = \frac{1}{Z}\exp\{&\sum_{v_i \in V}\sum_{x_{ik}^t \in X} \alpha_k f_k(x_{ik}^t, y_i^t) \\ &+ \sum_{v_j \in NB(v_i)}\sum_{(y_i^t, y_j^{t'}) \in Y^t} -\beta_{ji}(t-t')(y_i^t - y_j^{t'})^2 \\ &+ \sum_{v_i \in V}\sum_{(y_i^t, y_i^{t'}) \in Y^t} -\lambda_i(t-t')(y_i^t - y_i^{t'})^2\} \end{aligned} \tag{4}$$

where Z is a normalization factor; $NB(v_i)$ denotes the set of neighbors of v_i in the network; $(y_i^t, y_j^{t'})$ indicates a pair of close emotion status between v_i and v_j recorded in Y^t.

Learning the factor graph model is to estimate a parameter configuration $\theta = (\{\alpha_k\}, \{\beta_{ji}\}, \{\lambda_i\})$ from a given historic attribute-value log X^t, which maximizes the log-likelihood objective function $\mathcal{L}(\theta) = \log p_\theta(Y|G^t)$, i.e.,

$$\theta^\star = \arg\max_\theta \ \log\, p(Y = y|x, \theta) \tag{5}$$

Input: number of iterations and learning rate η;
Output: learned parameters $\theta = (\{\alpha_k\}, \{\beta_{ji}\}, \{\lambda_i\})$;

Initialize $\theta = \{\alpha, \beta, \lambda\}$;
repeat
  % sample a new configuration Y' based on $q(Y'|Y)$;
  $Y' \leftarrow q(Y'|Y)$;
  $\tau \sim \min(\frac{p(Y'|G^t,\theta)}{p(Y|G^t,\theta)}, 1)$;
  toss a coin s according to a $Bernoulli(\tau, (1-\tau))$;
  if $(s = 1)$ **then**
    $Y \leftarrow Y'$; % accept the new configuration Y';
    if $(Err(Y') < Err(Y)$ & $\Delta\theta F < 0)$ **then**
      $\theta^{new} \leftarrow \theta^{old} + \eta(\Delta\theta F)$;
    end
    else if $(Err(Y') > Err(Y)$ & $\Delta\theta F \geq 0)$ **then**
      $\theta^{new} \leftarrow \theta^{old} - \eta(\Delta\theta F)$;
    end
  end
until *convergence*;

Algorithm 1: The MH-based learning algorithm.

IV. Model Learning

It is usually intractable to do exact inference in such a graphical probabilistic model. The intrinsic difficulty is to calculate the normalization factor Z, which sums up all possible configurations of Y, thus making the complex exponential to the number of nodes in the graph. Several methods have been proposed to address this problem. For example, [3] defines all factor as a integral (quadratic) function, so Z can be calculated by a transformation to a multivariate Gaussian distribution. Some other methods such as Junction Tree [6] and Belief Propagation [7] are used to obtain an approximate solution. In this paper, we use a sampling-based Metropolis-Hastings (MH) algorithm [8], a particular Markov-chain Monte Carlo algorithm. The advantage of the MH algorithm is that it can derive a global gradient update for each parameter, thus obtaining better performance. The MH-based learning algorithm mainly consists of two key ingredients: (1) a proposal distribution, which defines how likely a new conditional configuration should be accepted; (2) parameter update according to the training error. In the following, we explain the learning algorithm in details.

As summarized in Algorithm 1, in each iteration of the learning algorithm, by the Metropolis-Hastings algorithm, we first sample a new configuration Y' conditioned on Y according to a proposal distribution $q(Y'|Y)$, which is defined over all possible configuration space $\mathcal{Y}$. The algorithm accepts the new configuration with an acceptance ratio $\tau \sim \min(\frac{p(Y'|G^t,\theta)}{p(Y|G^t,\theta)}, 1)$. If the new configuration Y' is accepted, the algorithm continues to update the parameters θ. Basically, there are two cases to update the parameters. In both cases, we first calculate two scores: error Err and un-normalized likelihood difference $\Delta\theta F$. The error is simply counting the number of mistakenly predicted examples on the training data. That is, based on the currently learned parameters θ, if the model predicts the emotion status of a user is positive at time t and the user's status is also (annotated as) positive, we say that the the model makes a correct prediction; otherwise a mistake. The un-normalized likelihood difference is calculated by $\Delta\theta F = \theta F(Y') - \theta F(Y)$, where $F(Y')$ is the exponent component of Eq. (4) (i.e., the formula without the normalization factor and the exponential function). Then in the first case, if the error $Err(Y')$ of the new Y' is lower than Y but the likelihood difference is negative (ideally should be positive), the algorithm updates the parameters by $\theta^{new} \leftarrow \theta^{old} + \eta\Delta\theta F$. In the second case, if error $Err(Y')$ of the new Y' is larger than Y but likelihood difference is positive (should be negative), the algorithm updates the parameters by $\theta^{new} \leftarrow \theta^{old} - \eta\Delta\theta F$.

The proposal distribution $q(Y'|Y)$ in Algorithm 1 can be defined in different ways. For simplicity, we can use a random distribution, that is, from the current configuration Y, we randomly change the emotion variable for each node and then obtain Y'. Another strategy is to use a heuristic algorithm to sample Y'. For example, we can calculate the un-normalized likelihood difference of a new configuration and in each iteration we try to find a configuration with the largest difference to update the parameters. However, surprisingly, we find that the random solution always clearly outperforms the heuristic-based proposal distribution, either on efficiency or effectiveness.

Mood Forcasting Based on the learned parameter θ, we can predict the users' future emotions. Specifically, The prediction problem can be cast as finding the emotion status that maximizes the likelihood given the learned parameters and historical data. Formally, the problem is an instance of the Maximum a Posteriori (MAP) problem as follows.

$$\arg\max_{y \in \mathcal{Y}} p(Y = y|G^t, \theta)$$

The prediction algorithm is also based on the Metropolis-Hastings algorithm. Specifically, we first initialize a configuration of emotion status Y. In each iteration, we sample a new configuration Y'. Then we calculate $\mathcal{L}(Y')$, the un-normalized log-likelihood of Y', and decide whether we accept it in the same way as that in the learning algorithm. Then we compare it to the optimal solution we obtained so far. If Y' is better ($\mathcal{L}(Y') > max$), we record it as the optimal solution.

V. Experimental Results

We conduct experiments on two data sets to evaluate the effectiveness of our proposed MoodCast. All data sets and codes are publicly available.[1]

A. Experimental Setup

Data Sets We perform our experiments on two different genres of data sets: one real mobile social network (MSN)

[1] http://arnetminer.org/moodcast/

and one virtual web-based network (LiveJournal). Statistics of the two data sets are shown in Table I.

In the MSN data set, we have collected all-day communication (by SMS and call), calendar, alarm, Wifi signal, GPS location, activity, and mood information from 30 volunteers of a university from May to July, 2010. This data represents over 36,000 hours of continuous data on human activity and emotion status. In total, there are about 9,869 human labels of the emotion status. Users in the MSN data set form a small social network.

Table I
STATISTICS OF THE MSN AND THE LIVEJOURNAL DATA SETS.

Dataset	Users	Links	Avg. links	Label	Avg. Labels
MSN	30	96	3.2	9,869	329
LiveJournal	469,707	23,318,572	49.6	2,665,166	5.7

The LiveJournal data set was collected from LiveJournal,[2] a social media platform where users share common passions and interests. Basically, users on LiveJournal can post what they are doing, how they are feeling, and give comments to the posts of their friends. The system also allows the users to associate a "mood label" to each of their posts. We conduct an analysis on the most 155 commonly used mood labels and classify them into three categories: positive, negative, and neutral. We collected the LiveJournal data set in the following ways. First we choose the administrator of the computer science community as the anchor and use a crawler to extract her friends. Then for each user, we extract his/her mood label and friends. With the recursively crawling process, eventually we obtain a data set of over 5GB. The derived social network from the data set (as shown in Table I) consists of 469K users and 23 million friendships between the users. On average, each user has 49.64 friends and publishes 5.64 posts on the website. We use words in the posts as the user attributes.

Baseline Methods We define four baseline methods for the emotion prediction task.

- *SVM-Simple.* The method only uses user attributes as features to train a classification model and then employs the classification model to predict the user mood. For SVM, we use SVM-light [3].
- *SVM-Net.* Besides using user attributes, the method also include the network information (i.e. social correlation and temporal correlation) as features, i.e., the same features as in our MoodCast approach.
- *Naive Bayes (NB-Simple).* The method uses the same features as that in SVM-Simple. The only difference is that it uses the Naive Bayes as the classifier.
- *Naive Bayes (NB-Net).* It uses the same features as that in SVM-Net and uses the Naive Bayes classifier.

[2]http://www.livejournal.com
[3]http://svmlight.joachims.org/

Evaluation Measures In all experiments, we evaluate the mood prediction performance in terms of Precision, Recall, and F1-Measure. We also analyze the convergent property and the factor contribution of the proposed MoodCast approach.

B. Mood Forecasting Performance

On both the two data sets, we use the historic data (time 0 to $t-1$) of the users as the training data. Then we predict a user's emotion status at time t given the attributes' values of the user at time t. In particular, in the MSN data set, we choose the data in the last 4 days as the test data and the rest as the training data. In the LiveJournal data set, we choose the data in the last 3 months as the test set and the rest as the training data set. Table II lists the prediction performance of the different approaches on the two data sets with the following observations.

From the results, we see that our method clearly outperforms the baseline methods on both data sets. On average, MoodCast achieves a 8+% improvement compared with both SVM and Naive Bayes methods. Moreover, we see that MoodCast has a stable performance while SVM varies greatly on LiveJournal dataset. We analyze the result by SVM and find that the poor performance of SVM is caused by the sparse of the value of user attributes. The attributes of LiveJournal data set are the keywords in their posts. However, there are many posts that do not contain discriminative keyword words. For example, a user posted "Survey, everyone read, you will know me better" which contains no useful words for predicting the emotion status. Solely considering the attribute information (as in SVM-Simple and NB-Simple) is difficult to accurately predict the emotion status. From Table II, we can also see that simply combining all the features (social correlation, temporal correlations) together (as in SVM-Net and NB-Net) can improve the prediction performance, but the performance is still unsatisfactory. Our method can leverage and differentiate friends' influence information and users' historic emotion information, thus achieving a better performance.

C. Analysis and Discussion

To obtain deeper understanding of the results, we perform the following analysis.

Effect of the Number of Sampling Iterations We conduct an experiment to see the effect of the number of the sampling iterations. We use the average F1-Measure of three classifiers to measure the overall performance. Figure 4 illustrates the experiment result. We see our MH-based learning algorithm converges in less than 100 iterations on both data sets. This suggests that learning algorithm is very efficient and as well has a good convergence property.

Factor Contribution Analysis In MoodCast, we consider three different factor functions: social correlation (S), temporal correlation (T), and attribute correlation (A). Here we

Table II
PERFORMANCE OF MOOD PREDICTION ON THE TWO DATASETS WITH DIFFERENT APPROACHES(%).

Classifier	Method	MSN Dataset			LiveJournal Dataset		
		Precision	Recall	F1-score	Precision	Recall	F1-score
Positive	MoodCast	68.42	69.23	**68.82**	52.50	73.68	**61.32**
	SVM-Simple	60.88	71.08	65.58	49.56	48.57	49.06
	SVM-Net	59.12	72.70	65.21	50.72	60.29	55.09
	NB-Simple	67.30	56.21	61.25	57.08	43.34	49.27
	NB-Net	71.89	56.59	63.33	59.1	47.38	52.59
Neutral	MoodCast	67.78	76.57	**71.90**	59.61	84.92	**75.44**
	SVM-Simple	67.39	59.73	63.33	67.58	78.69	72.71
	SVM-Net	68.42	55.11	61.05	71.21	78.13	74.51
	NB-Simple	54.14	68.04	60.30	65.95	54.14	59.46
	NB-Net	51.06	71.62	59.62	61.70	61.53	61.61
Negative	MoodCast	30.77	13.95	19.20	45.45	54.98	**49.77**
	SVM-Simple	5.63	4.54	5.03	71.67	37.39	49.14
	SVM-Net	8.18	16.90	11.02	68.78	37.68	48.68
	NB	14.70	28.16	19.32	54.77	36.61	43.89
	NB-Net	17.88	32.08	**22.96**	51.70	41.18	45.84
Average	MoodCast	55.66	53.25	**53.31**	52.52	71.19	**62.17**
	SVM-Simple	44.63	45.12	44.65	62.94	54.83	56.97
	SVM-Net	45.24	48.23	45.76	63.57	58.70	59.42
	NB-Simple	45.38	50.80	46.95	59.26	44.69	50.87
	NB-Net	46.94	53.43	48.63	57.5	50.03	53.35

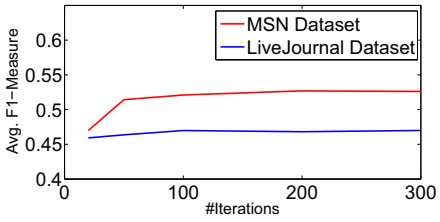


Figure 4. The influence of sampling iteration times.

perform the analysis to evaluate the contribution of different factor functions defined in our model. We first rank the individual factor by their prediction power, then remove those factors one by one in reversing order of their prediction power. In particular, we first remove social correlation factor function denoted as MoodCast-S, followed by removing the temporal correlation factor function denoted as MoodCast-ST, and then train and evaluate the prediction performance compared to MoodCast. In the MSN data set, we further remove the activity information (MoodCast-STA) and the location information (MoodCast-STAL).

Figure 5 shows the average F1-Measure score after ignoring the factor functions. We can observe clear drop on the performance when ignoring some of the factors. This indicates that our method works well by combining the different factor functions and each factor in our method contributes improvement in the performance. The only exception is that when ignoring the social correlation factor function on the MSN data set, there is no effect on the prediction performance. This is a bit surprising. Intuitively, in the real mobile network, users may be influenced by friends with a stronger degree than users in the virtual social network. By carefully investigating the data set, we found that in our MSN data set, the friendship network is sparse (averagely each user only has 3.2 friends) and the mobile users seldom

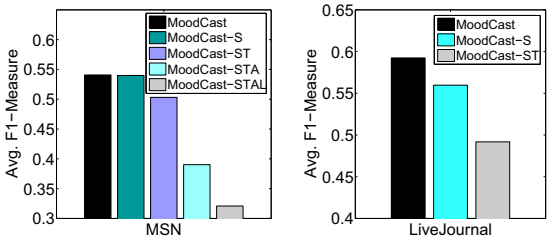


Figure 5. Contribution of different factor functions. The left is MSN result and the right is LiveJournal result. MoodCast-S stands for ignoring social correlation factor. MoodCast-ST stands for ignoring both the social correlation and the temporal correlation factors. MoodCast-STA stands for further ignoring activity attribute and MoodCast-STAL stands for further ignoring location attribute.

contact with each other through mobiles (this might be due to the limited number of participants).

VI. CONCLUSION

In this paper, we study a novel problem of *emotion prediction* in social networks. We propose a method referred to as MoodCast for modeling and predicting emotion dynamics in the social network. MoodCast formalizes the problem into a dynamic continuous factor graph model and defines three types of factor functions to capture the different types of information in the social network. For model learning, it uses a Metropolis-Hastings algorithm to obtain an approximate solution. Experimental results on two different real social networks demonstrate that the proposed approach can effectively model each user's emotion status and the prediction performance is better than several baseline methods for emotion prediction.

VII. *ACKNOWLEDGMENTS

The work is supported by the NSFC (No. 61073073, No. 60703059), NSFC Key Fund (No. 60933013), National High-tech R&D Program (No. 2009AA01Z138). It is also supported by a research award from Nokia China Research Center.

REFERENCES

[1] J. H. Fowler and N. A. Christakis, "Dynamic spread of happiness in a large social network: longitudinal analysis over 20 years in the framingham heart study," in *British Medical Journal*, 2008.

[2] J. Whitfield, "The secret of happiness: grinning on the internet," in *Nature*, 2008.

[3] C. Tan, J. Tang, J. Sun, Q. Lin, and F. Wang, "Social action tracking via noise tolerant time-varying factor graphs," in *KDD'10*, 2010, pp. 807–816.

[4] J. Tang, J. Sun, C. Wang, and Z. Yang, "Social influence analysis in large-scale networks," in *KDD'09*, 2009, pp. 807–816.

[5] A. Goyal, F. Bonchi, and L. V. Lakshmanan, "Learning influence probabilities in social networks," in *WSDM'10*, 2010.

[6] W. Wiegerinck, "Variational approximations between mean field theory and the junction tree algorithm," in *UAI'00*, 2000, pp. 626–633.

[7] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Generalized belief propagation," in *NIPS'01*, 2001, pp. 689–695.

[8] S. Chib and E. Greenberg, "Understanding the metropolischastings algorithm," *American Statistician*, vol. 49, no. 4, p. 327C335, 1995.