Collaborative Boosting for Activity Classification in Microblogs

Yangqiu Song[§], Zhengdong Lu[‡], Cane Wing-ki Leung[‡], Qiang Yang[‡]
yqsong@cse.ust.hk {lu.zhengdong,cane.leung,qiang.yang}@huawei.com
[§]Department of Computer Science and Engineering,
Hong Kong University of Science and Technology, Hong Kong
[‡]Noah's Ark Lab, Huawei, Hong Kong

ABSTRACT

Users' daily activities, such as dining and shopping, inherently reflect their habits, intents and preferences, thus provide invaluable information for services such as personalized information recommendation and targeted advertising. Users' activity information, although ubiquitous on social media, has largely been unexploited. This paper addresses the task of user activity classification in microblogs, where users can publish short messages and maintain social networks online. We identify the importance of modeling a user's individuality, and that of exploiting opinions of the user's friends for accurate activity classification. In this light, we propose a novel collaborative boosting framework comprising a text-to-activity classifier for each user, and a mechanism for collaboration between classifiers of users having social connections. The collaboration between two classifiers includes exchanging their own training instances and their dynamically changing labeling decisions. We propose an iterative learning procedure that is formulated as gradient descent in learning function space, while opinion exchange between classifiers is implemented with a weighted voting in each learning iteration. We show through experiments that on real-world data from Sina Weibo, our method outperforms existing off-the-shelf algorithms that do not take users' individuality or social connections into account.

Categories and Subject Descriptors
I.2.6[ArtificialIntelligence]: Learning; I.5.1[PatternRecognition]: Models
General Terms
Algorithms; Experimentation
Keywords
Boosting; Collaborative Classification; Activity Classification; Social Regularization.

1. INTRODUCTION

Social media platforms have become the most popular means for users to share online happenings about and around them, creating a rich repository of users' activity information. As users' activities inherently reflect their habits, intents and preferences, the ability to understand users' activity information is obviously invaluable for many services such as personalized information recommendation and targeted advertising. The task of *activity recognition* in social media, which is the task addressed in this paper, is about identifying what a user is doing at a particular time, given his or her social media postings such as microblog posts.

In the simplest sense, an "activity" can be any action that humans can perform. Examples include activities of daily living, such as doing housework, as well as social activities and entertainment, such as watching a movie. Traditionally, activity recognition is performed based on sensor readings obtained from, for instance, sensor-enabled environments or smartphones. It is a well-researched area in artificial intelligence, ubiquitous computing and computer vision, with wide applications in health care and smart home systems (e.g., [28, 13]). In social computing and data mining, however, research on activity recognition is just beginning, with much of users' activity information unexploited.

Note that activity recognition should not be confused with *event detection*, which has a similar goal of detecting real-life happenings in social streams. While an "event" can generally be defined as "something happening at a specific time and place" [25], most existing work focuses on *public events* that often generate large-scale or even population-level responses in social media. Examples include the use of Twitter[1] to detect and monitor natural disasters, political events and unusual activities such as protests [24, 17, 32].

The public nature of these events allows detection algorithms to utilize a large amount of aggregated data to, for instance, detect bursts or irregularities in topical contents within a short time span [25, 23]. Activity recognition, in contrast, is user-centric. Consider two users' microblog posts, saying "the referee is daydreaming" about a football match. While one user may be a spectator watching the match on TV at home, the other can be a commentator making live updates about the match. Yet, they could have posted similar contents at about the same time. The user-centric nature of activity recognition thus calls for personalized learning algorithms that can model the diversity and individuality of different users. Such a nature also results in data scarcity, because data of a single user are often too limited to train an accurate activity recognition model.

Currently, there exist two approaches to activity recognition in social media. The first is an *extraction* approach that extracts activities indicated by certain action verbs. For example, the work in [31] explored the use of Twitter to predict activities that users may perform in the near future, based on tweets mentioning a future time frame (published before 6 p.m. and contained the word "tonight") and a predefined activity word ("to watch"). This extraction approach, however, may fall short due to the brief, informal communication style of microblogs. For instance, a post men-

[1]http://www.twitter.com

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
KDD'13, August 11–14, 2013, Chicago, Illinois, USA.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
Copyright 2013 ACM 978-1-4503-2174-7/13/08 ...$15.00.

tioning "hamburger" may indicate that one is eating, even without action verbs like "eating" or "having". The second approach casts activity recognition as a *classification* task. The work in [18] used Conditional Random Fields (CRF) classifiers to predict users' future check-in locations, which are considered implied activities in the work. The classification approach, although avoiding the need to locate specific action verbs, is not without problems. For instance, the aforementioned example of two users posting about a football match shows that the mapping of textual contents to activities is not injective. We therefore need to resort to classifiers that can fully exploit the individuality of users, while addressing data scarcity resulting from the need for personalized models.

This paper proposes a collaborative learning framework to address the activity recognition task in social media, in particular in microblogs where users can share short messages and maintain social networks online. Instead of having one "global" classifier which cannot capture the individuality of users, our framework maintains many "local" classifiers, one for each user, while allowing collaborations between classifiers of socially connected users to alleviate data scarcity for improved classification performance. In the learning process, different learners compete (within each ensemble of classifiers for each user) and collaborate (across ensemble classifiers for different users) based on a designed mechanism for optimal performance. Knowledge is generated and transferred in a well controlled way, so that the classifier for one particular user can be well informed by others on the social network. Our learning framework is semi-supervised, as it can benefit from partially labeled data when unlabeled instances of one particular user dynamically receive labeling suggestions from neighboring users. Experiments on a data set we collected from Sina Weibo[2], a major microblog in China, validate the advantages of our model over several supervised baselines and state-of-the-art methods that perform activity classification with global classifiers.

We summarize the distinctive features of our proposed framework as follows.

- **Individuality:** the framework maintains many agents of the same type, each in charge of an independent classification task. Those classifiers have the same input/output and architecture, but with potentially different parameters and data.
- **Distributed:** the classifiers are fully distributed on the social network, each with its own training instances, including labeled and unlabeled ones. Those classifiers can be trained in a distributed manner, with a reasonable level of communication between them.
- **Connectivity:** the classification agents are inter-connected via some given relations, e.g. social relations in online social networks. Those relations will determine the way of communication between classifiers as well as the strength of it.
- **Collaborative:** connected classifiers can collaborate via sharing training instances, and exchanging classification opinions on instances owned by each other. This collaboration is a natural derivative of a carefully designed objective and our learning algorithm.

The rest of this paper is organized as follows. Section 2 introduces related studies and distinguishes our work from them. Section 3 formally defines our proposed framework, while Section 4 details the learning process of it. Section 5 describes experimental results before we conclude in Section 6.

2. RELATED WORK

The proposed framework is related to two threads of work, namely collective classification, and multi-task/transfer learning.

[2]http://www.weibo.com/

2.1 Collective Classification

Collective classification is concerned with classification problems on networked data [26], with the basic assumption that labels of connected nodes are more likely to be correlated. Different approaches have been proposed to incorporate this auxiliary correlation information. For example, some researchers proposed to align the labels of the classifiers [19, 20], while others allowed classifiers to borrow features from neighboring classifiers [7]. We also notice that there is one approach named as "network boosting" [29]. However, it assumes that the classifier is learned by a combination of classifiers on a random graph and each node on the graph should have the same set of training instances.

Collective classification has been applied to social media related tasks, such as sentiment analysis [27, 14]. Similar to our framework, existing work also uses a network-based regularization, and can be performed in a semi-supervised fashion. The major difference is that in collective classification, network regularization is performed on the group of instances to be classified, whereas ours is on a group of connected but distinctive classifiers. In other words, collective classification maintains a global classifier with inter-dependent instances, which is in clear contrast with our framework of collaborative learning with many local classifiers.

2.2 Multi-task learning

Multi-task learning (MTL) [6] is a learning setting where multiple related learning tasks are carried out jointly. It has been shown that when task relationships are properly exploited, the overall classification performance can be improved [1, 3, 4]. Our model can be naturally viewed as a special case of multi-task learning, since in our model the classifiers are on distinctive but related tasks, and the training of them is performed simultaneously.

Most MTL algorithms work by exploring unknown task relations [1, 3, 4]. Usually the relatedness between models of the tasks are captured by assuming that model parameters can be well described in some more "compact" representations, e.g, a few tight clusters or a subspace with low dimensionality. These assumptions can either be expressed through a regularization term on a joint loss function [10], or through some hardwired architecture [6]. Another thread of MTL explicitly learns the task relationships [16, 33, 34]. This thread of work usually assumes a reasonable amount of training data for the individual tasks; an amount that is sufficient for obtaining an acceptable model even when each task is trained alone. We do not make this assumption in our task, considering that a user often only has tens or hundreds of text messages on average. Such an amount of data is far from enough for obtaining a reasonable model. In other words, our task suffers from a more serious lack of training data.

There are two major characteristics differentiating our model from most other MTL models. Firstly, the measure of modeling relatedness/similarity in our model is rather distribution-dependent, which is fairly unique for our microblog-based classification tasks. In other words, the classification tasks are related not only in terms of model prediction behaviors, but also the distribution of the data they will likely see. Secondly, task relatedness in our model is exploited with a mechanism of dynamic opinion exchange (including training instances and labeling decisions) between related learning agents. This is in clear contrast with the static regularization over model parameters in most MTL models.

3. COLLABORATIVE CLASSIFICATION

In this section, we first formulate our task of text-based activity recognition in social media, and then present our collaborative learning framework.

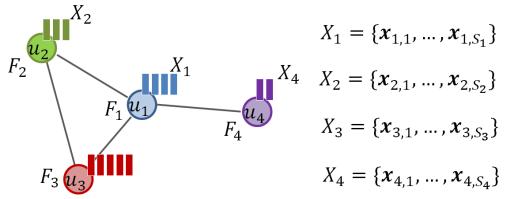


Figure 1: Illustration of a user's ego graph with notations and settings. The bars represent training instances.

3.1 Problem Definition

We represent a social network as a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, with node set $\mathcal{V} = \{u_1, \ldots, u_{|\mathcal{V}|}\}$ representing different users, and $\mathcal{E} = \{e_{ij}\}, i, j \in \{1, \ldots, |\mathcal{V}|\}$ specifying the social relations between them[3]. Each user u_i has S_i short microblog posts, denoted by the set $\mathcal{X}_i = \{\mathbf{x}_{i,1}, \ldots, \mathbf{x}_{i,s}, \ldots, \mathbf{x}_{i,S_i}\}$, with each $\mathbf{x}_{i,s} = [x_{i,s}^{(1)}, \ldots, x_{i,s}^{(n)}, \ldots, x_{i,s}^{(N)}]^\top$ representing the bag-of-word feature vector of the post indexed by (i, s). Some of the posts are labeled with predefined activities, e.g. Entertainment, Working, or None. For most of the users, only part of their instances (posts) are labeled. Without loss of generality, we index the labeled instances by $\{1, \cdots, S_i^\ell\}$, and the unlabeled ones by $\{S_i^\ell + 1, \cdots, S_i\}$.

We adopt the one-vs.-rest classification strategy to classify instances into activities. For each activity, the label of each $\mathbf{x}_{i,s}$, denoted as $y_{i,s}$, therefore takes binary values in $\{+1, -1\}$. Since we examine one activity label at a time under the one-vs.-rest strategy, we drop the activity index from the label $y_{i,s}$ for a concise representation. After applying our model to all activity labels, we can later combine the binary classification decisions to arrive at a final labeling decision. We omit this part and focus on our learning task in the subsequent discussions.

Our learning task is to find a mapping function from the feature vector $\mathbf{x}_{i,s}$ to the label $y_{i,s}$. In our collaborative learning framework, we assign a "local" classifier to each user to accommodate the user's individuality. Formally, our learning task is to find:

$$y_{i,s} \leftarrow F_i(\mathbf{x}_{i,s}), \tag{1}$$

where $F_i(\cdot)$ is the classifier of user u_i. This formulation grants us more modeling flexibility to describe the variety of users, but it naturally demands more training data for each user and apparently aggravates the problem of data scarcity. To alleviate this problem, we introduce a collaboration mechanism between the classifiers of different users, based on the following intuitions:

- We allow classifiers to share instances and exchange classification opinions with their neighbors in the social network. In what follows, we refer to this collaboration process as *opinion exchange* between classifiers, and the instances owned by a classifier as its *local data*.
- We maximize the margin on local data to measure the goodness of the classifier for each user. Since the data are in general only partially labeled, this large margin should be defined on both labeled and unlabeled data.

Fig. 1 shows an example of a user u_1's ego graph, which contains his friends u_2, u_3 and u_4. Each user has his own data and classifier. In what follows, we define the formal notations of classification and collaboration.

[3]Our model can take the the directionality (i.e., $e_{ij} \neq e_{ji}$) and the strength of edges (i.e., $e_{ij} \in \mathbb{R}^+$) into consideration, although in experiments we assume the edges are binary and undirected.

3.2 Classification and Collaboration

We define the following cost function over the classifiers $\{F_1, ..., F_{|\mathcal{V}|}\}$ in a social network:

$$J(F_1, ..., F_{|\mathcal{V}|}) = J_1(F_1, ..., F_{|\mathcal{V}|}) + \lambda J_2(F_1, ..., F_{|\mathcal{V}|}), \tag{2}$$

where $J_1(\cdot)$ is the margin-based cost defined on the local data of each classifier, and $J_2(\cdot)$ is the collaboration term promoting opinion exchange between classifiers.

Large Margin Cost

The cost function for the margin of local data (including labeled and unlabeled) is:

$$J_1(F_1, ..., F_{|\mathcal{V}|}) = \sum_i \frac{1}{S_i} \left[\sum_{s=1}^{S_i^\ell} e^{-V(y_{i,s}, F_i(\mathbf{x}_{i,s}))} + \mu \sum_{s=S_i^\ell+1}^{S_i} e^{-V_U(F_i(\mathbf{x}_{i,s}))} \right], \tag{3}$$

where

- $0 < \mu < 1$ controls the contribution of unlabeled data.
- $V(y_{i,s}, F_i(\mathbf{x}_{i,s}))$ is the cost function of labeled data $\mathbf{x}_{i,s}$,

$$V(y_{i,s}, F_i(\mathbf{x}_{i,s})) = y_{i,s} F_i(\mathbf{x}_{i,s}), \tag{4}$$

  where $y_{i,s} F_i(\mathbf{x}_{i,s})$ is considered the margin of $\mathbf{x}_{i,s}$. This part of the cost can be viewed as the empirical error on the labeled data [12].
- $V_U(F_i(\mathbf{x}_{i,s}))$ is the cost function of unlabeled data $\mathbf{x}_{i,s}$,

$$V_U(F_i(\mathbf{x}_{i,s})) \triangleq \hat{y}_{i,s} F_i(\mathbf{x}_{i,s}), \tag{5}$$

  with the pseudo labels $\hat{y}_{i,s} = \text{sign}(F_i(\mathbf{x}_{i,s}))$. This is the margin defined on unlabeled data $|F_i(\mathbf{x}_{i,s})|$, which pushes the decision boundary away from the region with dense distribution of unlabeled instances [9]. In addition, the notion of pseudo labels facilitates the design of the opinion exchange mechanism, and we defer descriptions of them to Section 4.3. Unlike the cost for labeled data, this part of the cost function is non-convex in F.

Social Network Regularization

The second term $J_2(\cdot)$ in Eq. (2) serves as a social regularization term to agitate the opinion exchange between classifiers, for better use of both labeled and unlabeled data distributed across the social network. Our design exploits two intuitions:

- **Classifier Similarity:** a user tends to mention the activities in a similar manner as his friends, which can be expressed as the similarity between the classification behaviors of the corresponding classifiers. Consider the football example we used in the introduction, with the message "the referee is daydreaming". If posted by someone in a circle of programmers, most likely it means that the user is watching TV at home (i.e., Entertainment). However, if posted by someone in a circle of sports commentators, it could mean the user is seriously at work (i.e., Working).
- **Circle-specific Distribution:** a user tends to mention similar entities and activities as his friends, but the vocabulary used to describe an activity could be significantly different between users from different social circles. For example, a college student may often use the words "football," "exams," "homework" and so on, whereas a senior researcher may tend to mention "collaboration," "workshop," and "conference trip," etc. This phenomenon is actually more important than one would expect since we are using word-based features with a huge vocabulary (say, 10^6), and the active terms used by any particular user or circle of users are only a small subset in the vocabulary. It is less meaningful for classifiers to collaborate on instances with features they barely use.

Thus, the social regularization term is defined as follows:

$$J_2(F_1, ..., F_{|\mathcal{V}|}) = \sum_{i,j:e_{ij}=1} \text{dis-sim}\left(F_i, F_j\right), \tag{6}$$

with the dis-similarity between two neighboring classifiers, F_i and F_j, measured as the difference between their classification decisions on the local data possessed by each of them:

$$\text{dis-sim}\left(F_i, F_j\right) = \sum_{s=1}^{S_i}\left(F_i(\mathbf{x}_{i,s}) - F_j(\mathbf{x}_{i,s})\right)^2 + \sum_{r=1}^{S_j}\left(F_i(\mathbf{x}_{j,r}) - F_j(\mathbf{x}_{j,r})\right)^2. \tag{7}$$

Assuming that a user can only see messages posted by his/her friends in a social network, the above equation captures our intuition that a user u_i's classifier should work similarly with his/her friend u_j's classifier on both users' local data[4]. It is easy to understand that although the similarity is explicitly defined on two socially connected classifiers, one classifier can still be affected by another that is several hops way through intermediate classifiers. However, one appealing property our model possesses is that, a classifier for one user is fairly robust to irrelevant social circles of his friends. We explain this with the following example. Suppose a college student Bob knows his friend Jerry in school, and Jerry has another social circle of colleagues in Microsoft, including a senior researcher Chris. Although the classifier for Bob may be affected by that of Chris through Jerry, this effect is minimal in our model if Bob's activity mentioning (e.g., "Went to a pub today, got totally drunk~") is significantly different from Chris's (e.g., "I am at Lake Tahoe for NIPS. Interesting workshops this year!").

4. COLLABORATIVE BOOSTING

This section is devoted to the learning process in our collaborative classification framework. Simply put, our learning model extends the conventional boosting model [21] to maintain multiple classifiers, one for each user, and introduces a new collaboration mechanism for socially connected classifiers.

The learning process can be viewed as a generalization of single classifier boosting, with much enriched mechanisms:

- Within the classifier of each user, there are a number of base learners competing to best fit the local data. This is similar to the single-classifier boosting system.
- Across socially connected classifiers, we propose a novel ensemble-level collaboration mechanism, encouraging opinion exchange between the classifiers. After each round of opinion exchange, a new base learner is generated for each classifier, again based on a competing mechanism.

The competing mechanism inherited from traditional boosting algorithm ensures a large margin on the local data owned by each user. As empirically verified by much work, this large margin often induces good generalization property of the classifiers. The collaboration mechanism, on the other hand, greatly alleviates the scarcity of training data for each individual classifier, while still maintains the locality of data transferring. It therefore avoids the spurious regularization problem faced by many collective classification models [14].

4.1 Functional Space Gradient

Our learning process is based on a collaborative boosting procedure, formulated as gradient descent in learning function space. Since the process involves an iterative algorithm, we introduce an index t in our notations hereafter to indicate the learning iteration, and T to denote the total number of learning iterations. For instance, $F_{i,t}$ is the classifier learned for user u_i at iteration t, and $F_{i,T}$ is that at the final learning process.

As a boosting algorithm, $F_{i,T}$ is learned through greedily growing the ensemble of base learners $f_{i,t}$ [11]:

$$y_{i,s} \leftarrow F_{i,T}(\mathbf{x}_{i,s}) = \sum_{t=1}^{T} \alpha_{i,t} f_{i,t}(\mathbf{x}_{i,s}), \tag{8}$$

where $f_{i,t}$ can be simple classifiers like decision trees, and T is the total learning steps when $F_{i,t}$ converges on training data. To make the representation clear, we call the ensemble $F_{i,t}(\cdot)$ the *classifier* and use the term *learner* for $f_{i,t}$'s.

Similar to the functional space gradient descent for single-task boosting [21], we define the optimization procedure of the cost function for each function $F_{i,t}(\cdot)$ as:

$$J(F_{i,t} + \epsilon f_{i,t+1}) \approx J(F_{i,t}) + \epsilon \langle \nabla J(F_{i,t}), f_{i,t+1} \rangle, \tag{9}$$

where the inner product $\langle F, G \rangle = \int_{\mathcal{X}} F(\mathbf{x})G(\mathbf{x})dp(\mathbf{x})$ is defined over the space of $\mathbf{x} \in \mathcal{X}$, and ϵ is a small value for first order Taylor expansion.

According to Eq. (9), the desired direction for each classifier $F_{i,t}$ is $\nabla J_{F_{i,t}}(F_{i,t})$. As in traditional boosting [21], we can select a function from a set of candidate functions from the function space, $f_{i,t+1} \in \mathcal{F}$, to approximate the direction of each step. $F_{i,t}$ is then a combination of functions, or more rigorously, $F_{i,t} \in \text{lin}(\mathcal{F})$. For the ease of training, we use a $f_{i,t+1}$ which itself is the sum of two base learners, each taking care of a term of $J(\cdot)$ in Eq. (2)

$$f_{i,t+1} = f^{(1)}_{i,t+1} + \lambda_{i,t+1} f^{(2)}_{i,t+1}, \tag{10}$$

where we have

$$f^{(1)}_{i,t+1} = \arg\max_{f \in \mathcal{F}} -\langle \nabla_{F_{i,t}} J_1(F_{i,t}), f \rangle \quad \text{and} \tag{11}$$

$$f^{(2)}_{i,t+1} = \arg\max_{f \in \mathcal{F}} -\langle \nabla_{F_{i,t}} J_2(F_{i,t}), f \rangle. \tag{12}$$

The first term $f^{(1)}_{i,t+1}$ attends to the fidelity to the local data (both labeled and unlabeled), while the second term $f^{(2)}_{i,t+1}$ agitates the opinion exchange between the classifiers of different users.

In the remainder of this section, we explain in detail how the two terms jointly determine the gradient descent, and how we control their interaction with a deterministic annealing process for achieving a better local optima.

4.2 Local Fitting

The negative derivative of $J_1(F_{i,t})$ for $F_{i,t}$ is itself a function, with values[5]:

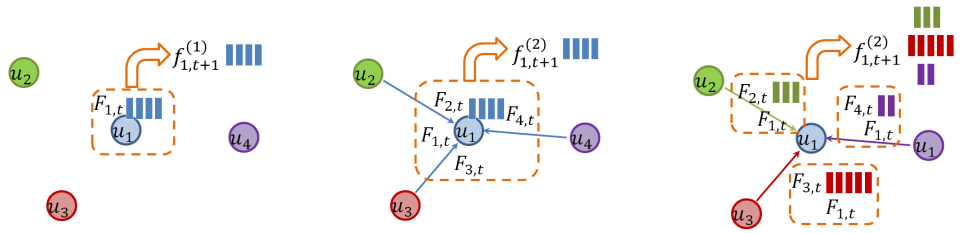
$$-\nabla_{F_{i,t}} J_1(F_{i,t}) = \begin{cases} y_{i,s} e^{-y_{i,s} F_{i,t}(\mathbf{x}_{i,s})} & s = 1, ..., S^\ell_i \\ \mu \hat{y}_{i,s,t} e^{-\hat{y}_{i,s,t} F_{i,t}(\mathbf{x}_{i,s})} & s = S^\ell_i + 1, ..., S_i \\ 0 & \text{otherwise} \end{cases}, \tag{13}$$

defined on data $\mathcal{X}_i$. A new classifier is then defined for user u_i by maximizing the inner product of $-\nabla_{F_i} J_1(F_{i,t})$ and $f^{(1)}_{i,t+1}$:

$$\begin{aligned} f^{(1)}_{i,t+1} &= \arg\max_{f \in \mathcal{F}} -\langle \nabla_{F_{i,t}} J_1(F_{i,t}), f \rangle \\ &= \arg\max \sum_{s=1}^{S^\ell_i} y_{i,s} f(\mathbf{x}_{i,s}) e^{-y_{i,s} F_{i,t}(\mathbf{x}_{i,s})} + \mu \sum_{s=S^\ell_i+1}^{S_i} \hat{y}_{i,s,t} f(\mathbf{x}_{i,s}) e^{-\hat{y}_{i,s,t} F_{i,t}(\mathbf{x}_{i,s})}. \end{aligned} \tag{14}$$

[4]Here we assume the graph is undirected, and we assume an edge between two users only if they follow each other. For directed graphs, we only check the data from one direction.

[5]For unlabeled data we take the sub-gradient [5] over $\mu \sum_{s=l_{i+1}}^{l_i+u_i} \hat{y}_{i,s} e^{-\hat{y}_{i,s} F_{i,t}(\mathbf{x}_{i,s})}$.



(a) Learn from my own data (b) Friends vote on my data (c) Borrow and learn from friends' data

Figure 2: Illustrations of the training process for user u_1's classifier (blue node and bars, best viewed in color). The arrows show the flow of classifiers' opinions, and bars represent training instances.

If we consider $f(\mathbf{x}_{i,s}) \in \{-1, +1\}$, the above optimization reduces to finding a function that minimizes a weighted error:

$$\text{err}_t^{(1)} = \sum_{s=1}^{S_i^\ell} D_t^{(1)}(\mathbf{x}_{i,s}) I_{[y_{i,s} \neq f(\mathbf{x}_{i,s})]} + \mu \sum_{s=S_i^\ell+1}^{S_i} D_t^{(1)}(\mathbf{x}_{i,s}) I_{[\hat{y}_{i,s,t} \neq f(\mathbf{x}_{i,s})]}, \quad (15)$$

given the fact that $yf(\mathbf{x}) = 1 - 2I_{[y \neq f(\mathbf{x})]}$, and $I_{[\cdot]}$ is the indicator function where $I_{\text{true}} = 1$ and $I_{\text{false}} = 0$. $D_t^{(1)}(\cdot)$ is a distribution given by:

$$D_t^{(1)}(\mathbf{x}_{i,s}) = \begin{cases} \frac{1}{Z_1} e^{-y_{i,s} F_{i,t}(\mathbf{x}_{i,s})} & \text{if } s = 1, ..., S_i^\ell \\ \frac{1}{Z_1} \mu e^{-\hat{y}_{i,s,t} F_{i,t}(\mathbf{x}_{i,s})} & \text{if } s = S_i^\ell + 1, ..., S_i \end{cases}, \quad (16)$$

where $Z_1 = \sum_{s=1}^{S_i^\ell} e^{-y_{i,s} F_{i,t}(\mathbf{x}_{i,s})} + \mu \sum_{s=S_i^\ell+1}^{S_i} e^{-\hat{y}_{i,s,t} F_{i,t}(\mathbf{x}_{i,s})}$ is a normalization constant.

As illustrated in Fig. 2(a), the learner $f_{1,t+1}^{(1)}$ only depends on the local data of u_1 and the classifier $F_{1,t}$ from previous step.

4.3 Opinion Exchange

Taking the negative derivative of $J_2(F_{i,t})$ for $F_{i,t}$ gives us

$$-\frac{1}{2}\nabla_{F_{i,t}} J_2(F_{i,t}) = \begin{cases} \sum_{j \in \mathcal{N}_i} F_{j,t}(\mathbf{x}_{i,s}) - |\mathcal{N}_i| F_{i,t}(\mathbf{x}_{i,s}) & s = 1, ..., S_i \\ F_{j,t}(\mathbf{x}_{j,r}) - F_{i,t}(\mathbf{x}_{j,r}) & r = 1, ..., S_j, \forall j \in \mathcal{N}_i \\ 0 & \text{otherwise} \end{cases}, \quad (17)$$

where $\mathcal{N}_i$ stands for the neighbors of user u_i with cardinality $|\mathcal{N}_i|$. For each u_i, we generate a base learner $f_{i,t+1}^{(2)}$ that learns from its own training samples as well as those from its friends j's. During the training process, the opinions of different users' classifiers are exchanged through the following two types of *pseudo labels*.

"Friends voting on my data"

Pseudo labels of this type are defined on user u_i's local data:

$$\hat{y}_{i,s,t}^{(1)} = \text{sign}\left(\sum_{j \in \mathcal{N}_i} F_{j,t}(\mathbf{x}_{i,s}) - |\mathcal{N}_i| F_{i,t}(\mathbf{x}_{i,s})\right), \quad (18)$$

with corresponding weight

$$D_t^{(2)}(\mathbf{x}_{i,s}) = \frac{1}{Z_2} \left|\sum_{j \in \mathcal{N}_i} F_{j,t}(\mathbf{x}_{i,s}) - |\mathcal{N}_i| F_{i,t}(\mathbf{x}_{i,s})\right|, \quad (19)$$

where Z_2 is a normalization constant:

$$Z_2 = \sum_{s}^{S_i} \left|\sum_{j \in \mathcal{N}_i} F_{j,t}(\mathbf{x}_{i,s}) - |\mathcal{N}_i| F_{i,t}(\mathbf{x}_{i,s})\right| + \sum_{j \in \mathcal{N}_i} \sum_{r}^{S_j} \left|F_{j,t}(\mathbf{x}_{j,r}) - F_{i,t}(\mathbf{x}_{j,r})\right|. \quad (20)$$

As suggested in Eq. (18), if u_i's classifier $F_{i,t}$ disagrees with the vote from its neighbors $\sum_{j \in \mathcal{N}_i} F_{j,t}(\mathbf{x}_{i,s})$ on its own unlabeled data, we flip the label of $\mathbf{x}_{i,s}$ to train the new classifier. Intuitively, when more friends are involved in the voting, the weight associated with the instance gets larger. As shown in Fig. 2(b), friends of u_1 pass their classifiers to u_1 and vote on the data of u_1 for learning the learner $f_{1,t+1}^{(2)}$.

"Borrowing friends' data"

The second type of pseudo labels is defined as:

$$\hat{y}_{j,r,t}^{(2)} = \text{sign}\left(F_{j,t}(\mathbf{x}_{j,r}) - F_{i,t}(\mathbf{x}_{j,r})\right), \quad (21)$$

and the corresponding weight is:

$$D_t^{(2)}(\mathbf{x}_{j,r}) = \frac{1}{Z_2} \left|F_{j,t}(\mathbf{x}_{j,r}) - F_{i,t}(\mathbf{x}_{j,r})\right|. \quad (22)$$

This means when checking with its friends' data $\mathbf{x}_{j,r}$, if the classifier $F_{i,t}(\cdot)$ is different from its friend's classifier $F_{j,t}(\cdot)$, or has a smaller margin on the data, we flip the label of $\mathbf{x}_{j,r}$ to train the new classifier. As shown in Fig. 2(c), friends of u_1 pass both their data and classifiers to u_1 and the difference between their classifiers and u_1's classifier is measured on their data for learning the learner $f_{1,t+1}^{(2)}$.

Putting the learning with the two types of pseudo-labels together, the learning of $f_{i,t+1}^{(2)}$ can be summarized as:

$$\begin{aligned} f_{i,t+1}^{(2)} &= \arg\max_{f \in \mathcal{F}} -\langle \nabla_{F_{i,t}} J_2(F_{i,t}), f \rangle \quad (23) \\ &= \arg\max_{f \in \mathcal{F}} \sum_{s}^{S_i} D_t^{(2)}(\mathbf{x}_{i,s}) \hat{y}_{i,s,t}^{(1)} f(\mathbf{x}_{i,s}) + \sum_{j \in \mathcal{N}_i} \sum_{r}^{S_j} D_t^{(2)}(\mathbf{x}_{j,r}) \hat{y}_{j,r,t}^{(2)} f(\mathbf{x}_{j,r}), \end{aligned}$$

where the last equation corresponds to a classification problem with different weights on its own data and its friends' data. The classification error is computed as:

$$\text{err}_t^{(2)} = \sum_{s}^{S_i} D_t^{(2)}(\mathbf{x}_{i,s}) I_{[\hat{y}_{i,s,t}^{(1)} \neq f(\mathbf{x}_{i,s})]} + \sum_{j \in \mathcal{N}_i} \sum_{r}^{S_j} D_t^{(2)}(\mathbf{x}_{j,r}) I_{[\hat{y}_{j,r,t}^{(2)} \neq f(\mathbf{x}_{j,r})]}. \quad (24)$$

In this case, both the friend's data and opinion on classification (margin on their data) are passed to u_i.

Note that the unlabeled data, although making the local fidelity term J_1 non-convex, serve as the medium in the opinion exchange term J_2. By using social regularization, i.e., by friends' voting and by checking friends' data, we use $f_{i,t+1}^{(2)}$ to correct the margins of $f_{i,t+1}^{(1)}$ in each step. Thus, the regularization is regarded as a help from friends to jump out from the local optima of the unlabeled data's margin. In the next section, we describe in more detail our approach to getting out of poor local optima with a deterministic annealing technique.

4.4 Deterministic Annealing

In this section, we present a learning process that can control the gradient descent method. In general, we need to find a greedy solution to the optimal $J(F_1, ..., F_{|\mathcal{V}|})$. However, if we emphasize too much on the regularization term, all the classifiers tend to be strongly affected by its neighboring classifiers. In our problem, we want to maintain the flexibility of individual classifiers, which means we care more about $J_1(F_1, ..., F_{|\mathcal{V}|})$. Therefore, we design a

mechanism that can decrease the objective function of both $J(F_1, ..., F_{|\mathcal{V}|})$ and $J_1(F_1, ..., F_{|\mathcal{V}|})$. This is done by first relaxing the controlling parameter λ in $J(F_1, ..., F_{|\mathcal{V}|})$ to be dynamically changing by iterations, and then we learn a global parameter for all the users to make sure the cost function for the network is decreasing in each step.

Controlling Each User

In each iteration, we choose a $\lambda_{i,t+1}$ to make sure all $J_1(F_{i,t})$ and $J(F_{i,t}, \lambda_{i,t+1}) = J_1(F_{i,t}) + \lambda_{i,t+1}J_2(F_{i,t})$ decrease. Suppose the new function for $F_{i,t}$ in the descent direction in function space is:

$$f_{i,t+1} = f^{(1)}_{i,t+1} + \lambda_{i,t+1} f^{(2)}_{i,t+1}, \tag{25}$$

where $\lambda_{i,t+1}$ is the parameter that can guarantee $J(F_{i,t}) = J_1(F_{i,t}) + \lambda_{t+1}J_2(F_{i,t})$ decreases, $f^{(1)}_{i,t+1}$ and $f^{(2)}_{i,t+1}$ are learned from Eqs. (14) and (23) respectively. The following lemma describes how to choose a suitable $\lambda_{i,t+1}$.

LEMMA 1. *Denote as:* $a_i = \langle \nabla J_2(F_{i,t}), f^{(2)}_{i,t+1} \rangle$, $b_i = \langle \nabla J_1(F_{i,t}), f^{(2)}_{i,t+1} \rangle + \langle \nabla J_2(F_{i,t}), f^{(1)}_{i,t+1} \rangle$ *and* $c_i = \langle \nabla J_1(F_{i,t}), f^{(1)}_{i,t+1} \rangle$. *Define*

$$L(\lambda_{i,t+1}) = \langle \nabla J_1(F_{i,t}) + \lambda_{i,t+1} \nabla J_2(F_{i,t}), f^{(1)}_{i,t+1} + \lambda_{i,t+1} f^{(2)}_{i,t+1} \rangle. \tag{26}$$

We have [30] :

- *In the case* $b_i \leq 0$, *for any* $\lambda_{i,t+1} > 0$, $L(\lambda_{i,t+1}) \leq 0$.
- *In the case* $b_i^2 - 4a_ic_i < 0$, *for any* $\lambda_{i,t+1} > 0$, $L(\lambda_{i,t+1}) < 0$.
- *In the case* $b_i > 0$ *and* $b_i^2 - 4a_ic_i \geq 0$, *for* $0 \leq \lambda_{i,t+1} \leq \frac{-b_i + \sqrt{b_i^2 - 4a_ic_i}}{2a_i}$, $L(\lambda_{i,t+1}) \leq 0$.

Given the learning objections of $f^{(1)}_{i,t+1}$ and $f^{(2)}_{i,t+1}$ from Eqs. (14) and (23) respectively, we have $a_i < 0$ and $c_i < 0$. The objective in Eq. (26) can be rewritten as: $L(\lambda_{i,t+1}) = a_i\lambda^2_{i,t+1} + b_i\lambda_{i,t+1} + c_i, \quad a_i < 0, c_i < 0$. The Lemma can be proved using some quadratic function analysis, which we omit here due to space limit.

Network Synchronization

Since we have $|\mathcal{V}|$ users, we need to find a λ_{t+1} to form an objective function that satisfies all users' functions. The following remark describes how we select the λ_{t+1} at iteration $t + 1$.

REMARK 1. **Deterministic Annealing:** *We select* $\lambda_{t+1} = \min_{\forall i} \lambda_{i,t+1}$ *to guarantee all the learners lead to the decreasing of the functionals* $J(F_1, ..., F_{|\mathcal{V}|})$ *and* $J_1(F_1, ..., F_{|\mathcal{V}|})$.

After determining the objective function for each iteration, and finding the possible direction of gradient in the function space, we want to go as far as we can using the Newton's method. We optimize the step size η_{t+1} for the overall objective function of at iteration $t + 1$ as follows:

$$\eta^*_{t+1} = \arg\min_{\eta_{t+1}} J(F_{1,t} + \eta_{t+1} \cdot f_{1,t+1}, \ldots, F_{|\mathcal{V}|,t} + \eta_{t+1} \cdot f_{|\mathcal{V}|,t+1}). \tag{27}$$

By taking the derivatives of η_{t+1}, we define (the detailed derivation is omitted due to the limited space): $g(\eta_{t+1}) = \frac{\partial J}{\partial \eta_{t+1}}$ and $h(\eta_{t+1}) = \frac{\partial^2 J}{\partial \eta^2_{t+1}}$. Then at each iteration over the graph, we compute η_{t+1} follows:

$$\eta_{t+1} \leftarrow \eta_{t+1} - g(\eta_{t+1})/h(\eta_{t+1}) \tag{28}$$

until convergence.

THEOREM 1. **Convergence:** *The deterministic annealing procedure will finally converge to a local optimum of maximum margin of each classifier.*

We skip the full proof here due to space constraint. The intuition is that since we choose a direction to ensure that the objective function $J_1(F_1, ..., F_{|\mathcal{V}|})$ decreases in each step, c_i's in Lemma 1 should be less than zero. Moreover, when $J_1(F_1, ..., F_{|\mathcal{V}|})$ converges to local optimum (follow the way of proof in [21]), c_i's tends to be zero. Then the selected $\lambda_{i,t+1}$ tends to be zero given the constraint in Lemma 1. In this case, the social regularization does not affect the further update of classifiers. The final base learner of this procedure is the same as semi-supervised boosting on each classifier, but the margin on unlabeled data has been refined through the procedure.

4.5 The Overall Algorithm

Putting the contents in Sections 4.1 to 4.4 together, we get our collaborative boosting algorithm, known as SocialBoost. Algorithm 1 provides the flowchart of our algorithm.

Algorithm 1 SocialBoost Algorithm.

Input Data: A social network as a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$. Each user has a set of data $\mathbf{x}_{i,s}$. If $\mathbf{x}_{i,s}$ is labeled, its label is $y_{i,s} \in \{+1, -1\}$.
Input Parameters: Maximum iteration number T; shrinkage parameter $0 < \lambda_0 < 1$.
Initialization: Learn f_0 from all the labeled data, let $F_{i,0} = f_0$ for $i = 1, \ldots, |\mathcal{V}|$.
while $1 < t < T$ **do**
  for Each user u_i **do**
    1. Learn $f^{(1)*}_{i,t+1}$ by minimizing $\mathsf{err}^{(1)}_t$ in Eq. (15).
    2. Learn $f^{(2)*}_{i,t+1}$ by minimizing $\mathsf{err}^{(2)}_t$ in Eq. (24).
    3. Determine $\lambda_{i,t+1}$:
    if $b_i > 0$ *and* $b_i^2 - 4a_ic_i \geq 0$ (from Lemma 1) **then**
      $\lambda_{i,t+1} = \lambda_0 \cdot \frac{-b_i + \sqrt{b_i^2 - 4a_ic_i}}{2a_i}$
    else
      $\lambda_{i,t+1} = \lambda_0$
    end if
  end for
  Set $\lambda_{t+1} \leftarrow \min \lambda_{i,t+1}$ for $J(F_{1,t}, ..., F_{|\mathcal{V}|,t})$.
  Compute $\eta_{t+1} \leftarrow \eta_{t+1} - g(\eta_{t+1})/h(\eta_{t+1})$ until convergence.
  Set $F_{i,t+1} \leftarrow F_{i,t} + \eta^*_{t+1}(f^{(1)*}_{i,t+1} + \lambda_{i,t+1} f^{(2)*}_{i,t+1})$, $i = 1, \ldots, |\mathcal{V}|$.
end while
Output: $F_{i,T}$, $i = 1, \ldots, |\mathcal{V}|$.

5. EXPERIMENTS

In this section, we present our experiments on real world data collected from Sina Weibo. All algorithms are evaluated based on classification accuracy.

5.1 Baseline Methods

We compared our algorithms with a total of six baseline methods, summarized as follows.

Linear Classification Without Regularization: The first baseline is a generic linear classifier:

- LS: This is a least squares linear classifier[6] with no regularization.

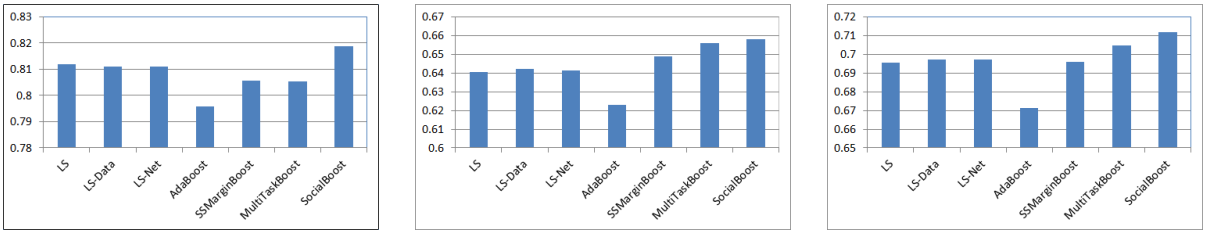
Graph-regularized Classification: Inspired by the collective classification algorithm in [14], we considered two types of graph-based regularization on top of LS:

- LS-Data: This baseline follows conventional graph-based semi-supervised learning [2], and derives a similarity graph between short messages based purely on their contents. We used a 5-NN graph with binary edges for better performance.
- LS-Net: In this baseline, the message-message similarity graph is derived from the relationships between users. Specifically, two messages are considered similar if their authors are socially connected.

In both LS-Data and LS-Net, classification is performed with a linear classifier with square loss and graph Laplacian regularization. Note that both methods perform instance-level regularization.

Boosting Algorithms: We considered the following three boosting-based methods with varying types of learning settings:

[6]We experimented with both supervised and semi-supervised settings, and found the square loss yields superior or comparable performance compared to hinge loss and logistic regression loss.



(a) Results on activity: ENTERTAINMENT (b) Results on activity: LIVING (c) Results on activity: OUTDOOR

Figure 3: Average activity classification result over the 316 sub-graphs.

- AdaBoost: The traditional AdaBoost algorithm [11] is used as a baseline for supervised learning without graph regularization. We use the default implementation in Mallet [22].
- SSMarginBoost: This is a semi-supervised margin-based boosting baseline we implemented following [5]. It leverages unlabeled data without any graph regularization.
- MultiTaskBoost: This is a multi-task boosting algorithm we implemented following [8], as the state-of-the-art supervised model that exploits task relationships. This algorithm uses a common classifier trained on the entirety of labeled data, and allows individual classifiers to deviate from the global classifier with designed penalty.

5.2 Data Set

Our experiments are based on a data set we crawled from Sina Weibo. The full data set contains short messages and social graphs of about 10 million users. The original data set is unlabeled, and we exploited the "check-in" information provided by users to derive the gold standard for evaluation. A check-in indicates the point of interest where the user was located. About 0.85 million users have at least one check-in. We describe below how we constructed the final data set for experiments from this set of users.

Constructing user sub-graphs: For each user, we traversed his social graph to find his followees who (i) follow him mutually, and (ii) also have at least one check-in. Then we build a node-induced sub-graph from this set of users. We call such sub-graph a *user sub-graph*, and we include only mutual followers to ensure that each sub-graph represents a reasonable *community* with a closely connected group of users. There are 60k user sub-graphs containing more than 10 nodes, and 316 containing at least 50 nodes. We focus on these 316 sub-graphs hereafter.

We further filtered out users in the 316 sub-graphs with fewer than 10 messages to obtain our final data set. After filtering, each sub-graph contains an average of 24.49 ± 11.58 users and 57.59 ± 39.38 edges. Each user has on average 23.88 ± 16.93 messages with check-in information.

Assigning activity labels: We used a rule-based program to infer the activity associated with each message based on its check-in location, posting time and named entities mentioned in the text. We identified a total of 33 activities, which we group into 3 main categories (surrogate labels):

1. ENTERTAINMENT: include watching movies, going to Karaoke, shopping, etc.
2. LIVING: include having meals, staying at home, going to spa or hair cutting, etc.
3. OUTDOOR: include sports, travel, waiting for bus, etc.

We engaged an external annotator to manually examine about 3,200 messages. The inferred labels are found to be correct for over 90% of them.

Text preprocessing: We used the Stanford Chinese segmenter[7] to tokenize the messages, and filtered out stop-words based on a list of 1,208 common Chinese words. We used Mallet for the rest of preprocessing and for converting the messages into feature vectors.

5.3 Comparison of Different Algorithms

We demonstrate the superiority of our algorithm in two settings. First, we test all the algorithms on each of the 316 sub-graphs (single sub-graph setting). Second, we combine some of the sub-graphs into a larger graph to form a multiple sub-graph setting, and the idea of which is to create a network with multiple circles of users. In each experimental trial, we split the data set into a labeled training set (30%), an unlabeled set (50%) and a validation set (20%). Experiments are performed in a transductive setting [15] with 10 random trials. Reported results are averages of the 10 trials on the unlabeled sets. The validation sets are used for early stopping of the boosting-based algorithms.

Among the tested algorithms, LS, AdaBoost and MultiTaskBoost are supervised methods. LS-Data, LS-Net, SSMarginBoost and SocialBoost are transductive semi-supervised approaches, where unlabeled data are involved in the training process.

Results on Single User Sub-graphs

Table 1 presents the results of the various algorithms on three randomly selected sub-graphs. We first observe that SocialBoost outperforms the other algorithms in most cases. LS-Data and LS-Net only achieve slight improvements over the LS algorithm. Both SSMarginBoost and MultiTaskBoost beat their generic counterpart AdaBoost on the three graphs, showing the advantages of the semi-supervised setting and exploiting task relationships.

In Fig. 3, we summarize the average performance of all algorithms over the 316 sub-graphs. We can see that SocialBoost performs the best on average for all three activity categories.

Results on Multiple User Sub-graphs

We now compare the performance of the different algorithms in the multiple sub-graph setting, formed by combining the three sub-graphs described in the last experiment. Table 2 reports the results on the combined graph, showing that SocialBoost still outperforms all other algorithms.

One interesting observation we make is that the instance-level regularization in LS-Data and LS-Net reduces classification accuracy. Our conjecture is the following. The social relations between distantly connected users (e.g., friend's friend) are in general fairly weak, especially when the users are from two different circles of their common friends. These distant connections, however, still play a rather important role in the regularization given the large number of them. To be more specific, for LS-Data, the text similarity graph could get rather noisy with more messages in a large so-

[7] http://nlp.stanford.edu/projects/chinese-nlp.shtml

Table 1: Comparison of different algorithms on single sub-graphs.

	Sub-graph One			Sub-graph Two			Sub-graph Three		
	# Node: 13, # Edge: 30, # Message: 265			# Node: 20, # Edge: 33, # Message: 403			# Node: 26, # Edge: 104, # Message: 518		
	ENTERTAINMENT	LIVING	OUTDOOR	ENTERTAINMENT	LIVING	OUTDOOR	ENTERTAINMENT	LIVING	OUTDOOR
LS	75.30(1.48)%	68.26(1.29)%	70.76(1.74)%	81.79(1.30)%	68.26(1.29)%	65.47(1.67)%	77.84(1.11)%	60.46(3.42)%	75.75(2.01)%
LS-Data	75.30(2.55)%	68.36(1.25)%	71.36(1.46)%	81.89(1.30)%	68.36(1.25)%	65.47(2.07)%	77.92(0.63)%	60.69(2.77)%	76.21(2.23)%
LS-Net	75.90(1.96)%	68.16(1.69)%	71.06(1.96)%	81.99(0.82)%	68.16(1.69)%	65.97(1.30)%	77.61(0.82)%	60.23(3.36)%	75.98(2.61)%
AdaBoost	72.58(0.83)%	65.77(2.09)%	70.45(5.33)%	78.51(4.39)%	65.77(2.09)%	63.08(3.13)%	76.22(2.66)%	56.14(3.10)%	72.98(4.07)%
SSMarginBoost	75.91(1.96)%	69.75(0.82)%	70.61(3.80)%	80.40(1.03)%	69.75(1.81)%	**68.06(2.62)%**	78.22(1.11)%	**62.47(3.65)%**	75.14(2.07)%
MultiTaskBoost	73.95(3.16)%	67.56(1.77)%	70.68(1.89)%	82.33(0.78)%	67.56(1.77)%	65.59(2.33)%	77.37(3.16)%	60.77(2.91)%	74.98(3.14)%
SocialBoost	**77.26(1.94)%**	**70.51(2.75)%**	**72.39(2.19)%**	**83.13(0.94)%**	**70.51(1.74)%**	66.36(2.50)%	**79.15(1.54)%**	60.77(3.58)%	**76.29(1.86)%**

Table 2: Comparison of different algorithms on a combined graph. (# node: 59, # edge: 167, # message: 1186)

	ENTERTAINMENT	LIVING	OUTDOOR
LS	80.99(0.91)%	66.61(1.22)%	71.63(1.31)%
LS-Data	74.24(1.60)%	59.60(1.34)%	64.20(1.64)%
LS-Net	74.18(1.73)%	60.73(0.86)%	63.84(1.56)%
AdaBoost	76.37(1.06)%	62.78(0.81)%	69.13(1.66)%
SSMarginBoost	78.17(1.15)%	66.53(1.58)%	71.92(1.23)%
MultiTaskBoost	79.05(1.47)%	67.03(1.55)%	71.91(1.55)%
SocialBoost	**81.10(1.04)%**	**67.17(1.08)%**	**73.32(1.28)%**

cial graph, which may hurt the performance of the semi-supervised learning algorithm. For LS-Net, the message-message similarity is derived from the social relations between users. The assumption behind this is that text messages of a user and those of his friends tend to mention the same activity. The regularization could easily become harmful when too much regularization is added to one global classifier, resulting in a form of under-fitting. We want to note here that activity classification might be more sensitive to users than tasks like sentiment analysis [14]. One can assume that one user's sentiment can be consistent on social media whereas activities can be more diverse.

On the other hand, SocialBoost is more robust to distant connections of a user. As we stated in Section 3.2, in SocialBoost, a classifier is only influenced by its neighbors on instances they own (which are a subset of the messages they can both see with the Weibo policy). As a result, a classifier of user u_i is less affected by that of a two-hop friend u_k, if u_k is from another social circle and has little in common with u_i. In an extreme case, if u_i and u_j always talk about different things and have no common word features in their messages, then their classifiers can only be affected by instances owned by their common friend, say u_j. Moreover, if u_j's messages deviate greatly from u_i's, then the decision from u_i's classifier usually has a small margin and hence less weight in voting.

Knowing that the linear classifiers do not perform well on a network with multiple user sub-graphs, we further create a larger network with 38 sub-graphs to study the performance of the boosting-based algorithms. Table 3 shows the results. When there are sufficient labeled data, the large amount of unlabeled data may not help as much as in the case when labeled data are limited. As a result, SSMarginBoost does not show improvements over AdaBoost. Moreover, the global classifier learned by MultiTaskBoost from all data eventually dominates the classification results. For SocialBoost, as it always considers local data of users, it does not depend heavily on the size of training set. Furthermore, it is always able to benefit from social network information.

5.4 The Effect of Social Relationships

In this section, we investigate the effect of social relations on the performance of SocialBoost. The experiments are designed as follows. We alter the connections between users in the user-sub-graphs to introduce noise/randomness to them, apply SocialBoost to the altered sub-graphs, and compare its performance with that achieved on the original sub-graphs. If social connections are indeed useful for activity classification as we expected, then there shall be a decay in classification performance as randomness is introduced to the original sub-graphs.

Table 3: Comparison of different algorithms on a combined graph of 38 sub-graphs. (# node: 812, # edge: 1,431, # message: 18,169)

	ENTERTAINMENT	LIVING	OUTDOOR
AdaBoost	80.62(0.76)%	64.38(0.24)%	74.55(0.35)%
SSMarginBoost	80.52(0.53)%	64.30(1.22)%	73.93(1.93)%
MultiTaskBoost	80.88(1.85)%	62.61(3.14)%	72.31(2.46)%
SocialBoost	**82.07(0.82)%**	**65.90(0.64)%**	**75.78(0.49)%**

We consider the following three ways of altering the original sub-graphs, with increasing noise levels/spurious connections:

- **Disconnected:** we remove all social relations in a sub-graph to make users disconnected from each other, and hence all classifiers become independent.
- **50% Random Edges:** we remove all social relations, and add random connections between users. Each pair of users has 50% probability to be connected[8].
- **Reversely Connected:** we turn off all the original social relations, and connect those previously unconnected user pairs.

Table 4 reports the performance of SocialBoost under these alteration settings, on the combined graph of three sub-graphs used in last experiment. Not surprisingly, SocialBoost works best on the original connections. Its performance deteriorates as the number of spurious connections increases, and is the worst when connectivity is completely reversed. These confirm the usefulness of social connections in boosting classification performance.

Table 4: Performance of SocialBoost on the original social connections and under the three alteration settings. (# node: 59, # edge: 167, # message: 1186)

	ENTERTAINMENT	LIVING	OUTDOOR
Original Graph	**79.38(1.29)%**	**64.23(1.21)%**	**70.65(2.64)%**
Disconnected	78.35(1.18)%	63.10(1.73)%	69.83(2.16)%
50% Random Edges	77.56(2.00)%	60.82(1.59)%	68.43(2.70)%
Reversely Connected	76.37(1.73)%	60.72(1.58)%	68.39(2.45)%

5.5 Results on Runtime

We also tested the runtime of SocialBoost with varying numbers of users in the network, and compared it with MultiTaskBoost. The base learner is a decision tree with a maximum depth of 3. Starting with a network with 10% of randomly selected user sub-graphs, we

[8]We experimented with different percentages of random edges and observed similar qualitative results.

progressively add another 10% of randomly selected sub-graphs to the network until all sub-graphs are added. When the number of user sub-graphs in the network increases, the number of users as well as the number of text messages increase monotonically. The final network combining all 316 sub-graphs contains 5,787 users, 14,124 edges and 136,594 text messages. In each experimental trial, we run both algorithms for three learning iterations and record their runtime results. Fig. 4 shows that SocialBoost requires more runtime than MultiTaskBoost, and its runtime grows superlinearly with respect to the number of users. This is because when a user is added to the network, our algorithm will actually perform learning not only on the user alone, but also on his/her friends for opinion exchange.

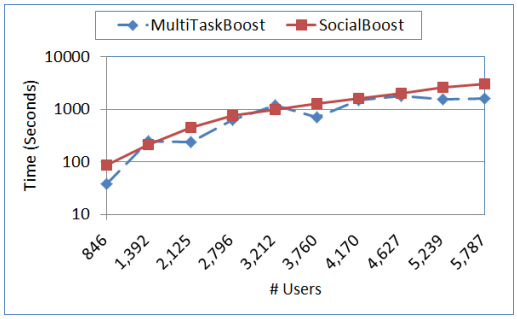


Figure 4: Runtime results of MultiTaskBoost and SocialBoost.

6. CONCLUSIONS AND FUTURE WORK

This paper presents a collaborative boosting algorithm for the activity classification problem in microblogs. Our work is motivated by two properties of the problem, namely users' individuality and data scarcity resulted from the need for personalized classifiers. We propose a collaborative learning framework that maintains a classifier for each user, while allowing opinion exchange between classifiers that are connected through social relationships based on our proposed collaboration mechanism. We show through experiments on real-world data from Sina Weibo that our algorithm outperforms several state-of-the-art algorithms and baselines.

We conclude by outlining our future work. Firstly, the proposed SocialBoost algorithm in essence maintains classifiers distributed over users in a social network. We would like to try implementing our algorithm in a distributed environment and experimenting with larger data sets. Secondly, we plan to explore more applications of our algorithm. While the work in this paper is motivated by the nature of the activity classification task, we expect our algorithm to be generally applicable to other tasks with a similar nature. One such task may be sentiment analysis, for which it may be desirable to maintain personalized sentiment classifiers for individual users while exploiting their friends' opinions for better classification results. Another example would be pattern-based information extraction, where we are interested in the effectiveness of using social regularization to bootstrap extraction patterns induced by base learners from social voting.

7. ACKNOWLEDGEMENTS

We appreciate the kind suggestions and helpful discussion from Dr. Hang Li when writing this paper. We would like to thank the reviewers for their helpful comments to improve this paper. We also thank the support of Hong Kong CERG grants 621211 and 620812.

8. REFERENCES

[1] J. Baxter. A bayesian/information theoretic model of learning to learn viamultiple task sampling. *Machine Learning*, 28(1):7–39, 1997.

[2] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *JMLR*, 1(1):1–48, 2006.

[3] S. Ben-David, J. Gehrke, and R. Schuller. A theoretical framework for learning from a pool of disparate data sources. In *KDD*, pages 443–449, 2002.

[4] S. Ben-David and R. Schuller. Exploiting task relatedness for mulitple task learning. In *COLT*, pages 567–580, 2003.

[5] K. P. Bennett, A. Demiriz, and R. Maclin. Exploiting unlabeled data in ensemble methods. In *KDD*, pages 289–296, 2002.

[6] R. Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

[7] S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In *SIGMOD*, pages 307–318, 1998.

[8] O. Chapelle, P. Shivaswamy, S. Vadrevu, K. Weinberger, Y. Zhang, and B. Tseng. Multi-task learning for boosting with application to web search ranking. In *KDD*, pages 1189–1198, 2010.

[9] F. d'Alché Buc, Y. Grandvalet, and C. Ambroise. Semi-supervised marginboost. In *NIPS*, pages 553–560, 2001.

[10] T. Evgeniou and M. Pontil. Regularized multi–task learning. In *KDD*, pages 109–117, 2004.

[11] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55:119–139, 1997.

[12] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer-Verlag, 2001.

[13] A. Helal, D. J. Cook, and M. Schmalz. Smart home-based health platform for behavioral monitoring and alteration of diabetes patients. *Journal of diabetes science and technology (Online)*, 3(1):141, 2009.

[14] X. Hu, L. Tang, J. Tang, and H. Liu. Exploiting social relations for sentiment analysis in microblogging. In *WSDM*, 2013.

[15] T. Joachims. Transductive inference for text classification using support vector machines. In *ICML*, pages 200–209, 1999.

[16] Z. Kang, K. Grauman, and F. Sha. Learning with whom to share in multi-task feature learning. In *ICML*, pages 521–528, 2011.

[17] R. Lee, S. Wakamiya, and K. Sumiya. Discovery of unusual regional social activities using geo-tagged microblogs. *WWW Journal*, 14:321–349, 2011.

[18] D. Lian and X. Xie. Collaborative activity recognition via check-in history. In *SIGSPATIAL Workshop on Location-Based Social Networks*, pages 45–48, 2011.

[19] Q. Lu and L. Getoor. Link-based classification. In *ICML*, pages 496–503, 2003.

[20] S. A. Macskassy and F. Provost. Classification in networked data: A toolkit and a univariate case study. *JMLR*, 8:935–983, 2007.

[21] L. Mason, J. Baxter, P. L. Bartlett, and M. Frean. Functional gradient techniques for combining hypotheses. *Advances in Large Margin Classifiers*, pages 221–246, 2000.

[22] A. K. McCallum. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.

[23] A. Ritter, Mausam, O. Etzioni, and S. Clark. Open domain event extraction from twitter. In *KDD*, pages 1104–1112, 2012.

[24] T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *WWW*, pages 851–860, 2010.

[25] H. Sayyadi, M. Hurst, and A. Maykov. Event detection and tracking in social streams. In *ICWSM*, 2009.

[26] P. Sen, G. M. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.

[27] C. Tan, L. Lee, J. Tang, L. Jiang, M. Zhou, and P. Li. User-level sentiment analysis incorporating social networks. In *KDD*, pages 1397–1405, 2011.

[28] E. Tapia, S. Intille, and K. Larson. Activity recognition in the home using simple and ubiquitous sensors. *IEEE Pervasive Computing*, pages 158–175, 2004.

[29] S. Wang and C. Zhang. Network game and boosting. In *ECML*, pages 461–472, 2005.

[30] Z. Wang, Y. Song, and C. Zhang. Homotopy regularization for boosting. In *ICDM*, pages 1115–1120, 2010.

[31] W. Weerkamp and M. de Rijke. Activity prediction: A twitter-based exploration. In *SIGIR Workshop on Time-aware Information Access*, 2012.

[32] J. Weng and B.-S. Lee. Event detection in twitter. In *ICWSM*, 2011.

[33] Y. Zhang and D.-Y. Yeung. A convex formulation for learning task relationships in multi-task learning. In *UAI*, 2010.

[34] Y. Zhang and D.-Y. Yeung. Multi-task boosting by exploiting task relationships. In *ECML/PKDD*, pages 697–710, 2012.