Biased Random Walk based Social Regularization for Word Embeddings

Ziqian Zeng1*, Xin Liu1,2*, and Yangqiu Song1
1Department of CSE, The Hong Kong University of Science and Technology
2School of Data and Computer Science, Sun Yat-sen University
zzengae@cse.ust.hk, xliucr@connect.ust.hk, yqsong@cse.ust.hk

Abstract

Nowadays, people publish a lot of natural language texts on social media. Socialized word embeddings (SWE) has been proposed to deal with two phenomena of language use: everyone has his/her own personal characteristics of language use and socially connected users are likely to use language in similar ways. We observe that the spread of language use is transitive. Namely, one user can affect his/her friends and the friends can also affect their friends. However, SWE modeled the transitivity implicitly. The social regularization in SWE only applies to one-hop neighbors and thus users outside the one-hop social circle will not be affected directly. In this work, we adopt random walk methods to generate paths on the social graph to model the transitivity explicitly. Each user on a path will be affected by his/her adjacent user(s) on the path. Moreover, according to the update mechanism of SWE, fewer friends a user has, fewer update opportunities he/she can get. Hence, we propose a biased random walk method to provide these users with more update opportunities. Experiments show that our random walk based social regularizations perform better on sentiment classification.

1 Introduction

Word embeddings [Mikolov et al., 2013a; Mikolov et al., 2013b; Pennington et al., 2014] have been widely used in natural language processing tasks. When analyzing the natural language use on social media platforms or customer review websites such as Facebook, Twitter, and Yelp, we observe that everyone has his/her own personal characteristics of language use. For example, everyone has his/her own preference for diction and expression method. The hypothesis of distributional representation states that words with similar meanings tend to appear in similar contexts [Harris, 1954]. For different people, contexts around a word would be different due to their personal characteristics of language use. Hence, there is a need to consider personal characteristics of language use in word embeddings. We also observe that socially connected people tend to use language in similar manners. As indicated by [Hovy, 2015], language use can be affected by demographic factors such as age, gender, race, geography and so on [Rosenthal and McKeown, 2011; Eckert and McConnell-Ginet, 2003; Green, 2002; Trudgill, 1974; Fischer, 1958; Labov, 1963]. For example, scientists on social media may mention more scientific and hi-tech related terms while movie stars may mention more about entertainment news. Moreover, some groups of people say “Y gotta do it the right way.” while others say “You have to do it the right way.” It is not straightforward to access demographic information, but it is reasonable to believe that friends on social media tend to have some demographic factors in common. Hence, it is reasonable to incorporate this phenomenon when we consider personalized language use.

Socialized word embeddings (SWE) [Zeng et al., 2017] has been proposed to deal with aforementioned two phenomena with two modifications of word embeddings: personalization and socialization. For personalization, SWE added a social regularization term to impose user vectors between friends to be similar. It was shown that SWE can improve word embeddings on word representation learning for social media sentiment analysis [Zeng et al., 2017]. However, there are still two problems.

First, we observe that the spread of language use is transitive. Namely, one user can affect his/her friends and the friends can further affect their friends. However, the social regularization in SWE only applies to pairwise friends, i.e., one-hop neighbors. The users outside the one-hop social circle will not be updated until one of their one-hop neighbors is considered, so the transitivity is modeled implicitly. It would be more effective if we can model the transitivity more explicitly in the social regularization. Second, since there are in general more texts input by users with more friends than texts input by users with fewer friends, when optimizing the social regularization term, users with fewer friends will be updated much less than users with more friends. It would be helpful if we can find a better way to model users with fewer friends to train their user embeddings more frequently.

*Equal contribution. This work was done when Xin Liu was a research assistant at HKUST.
To solve the above two problems, in this paper, we propose to use random walk based methods for the social regularization, which can generate paths to explicitly model the transitivity and can control the process of user sampling. The nodes in the path are analogous to affected users during the propagation of language use. Each user on a path will be affected by his/her adjacent user(s) on the path. The change triggered by start user will pass along the path and all users will be updated accordingly. We propose to use both first-order and second-order random walk based regularizations. In first-order random walk, a random walker moves to next node based on the last node while in second-order random walk [Grover and Leskovec, 2016], it relies on both the last and the second last nodes. Experiments show that social regularizations using aforementioned random walk methods perform better than SWE. Moreover, we propose a biased random walk based regularization which introduces a bias coefficient to adjust transition probabilities. The bias coefficient is associated with the number of friends of a user so that users with fewer friends can be sampled more frequently. Experiments show that our random walk based social regularizations perform better on sentiment classification on Yelp review datasets. The code is available at https://github.com/HKUST-KnowComp/SRBRW.

2 Related Work

In this section, we review our related work in two categories.

2.1 Personalization and Socialization in Language Modeling

Language models are fundamental to natural language processing. Users of search engines often have different search purposes even when they submit the same query. To consider personalization, personalized language models have been developed by [Croft et al., 2001; Song et al., 2010; Sontag et al., 2012] and applied to personalized web search. However, some users may not have sufficient corpora to train personal language models. Hence, socialized language models [Yosecky et al., 2014; Huang et al., 2014; Yan et al., 2016] were proposed to deal with the sparsity problem. Word embeddings are also important in NLP and can be easily integrated into downstream tasks. Socialized word embeddings [Zeng et al., 2017] was proposed to deal with phenomena of language use. Everyone has his/her own personal characteristics of language use and socially connected users are likely to use language in similar ways.

2.2 Random Walk Methods

A random walk is a stochastic process which consists of movements from a node to another adjacent node. Each movement relies on previous node(s) and associated transition probabilities. Random walks have been applied in different research fields [Weiss, 1983]. Network representation learning is one of successful applications. Inspired by word2vec [Mikolov et al., 2013a; Mikolov et al., 2013b], DeepWalk [Perozzi et al., 2014] adopted a first-order random walk method to generate walks by treating walks as the equivalent of sentences. The transition probabilities used in DeepWalk are uniform. To capture a diversity of network structures, node2Vec [Grover and Leskovec, 2016] used second-order random walks to learn better representation. Node2vec used two tunable parameters to flexibly adjust exploration strategies.

3 Methodology

We first briefly introduce the SWE model [Zeng et al., 2017] and then introduce our random walk methods.

3.1 Socialized Word Embedding (SWE)

Suppose there are \( N \) users \( u_1, \ldots, u_N \) in a social network. A user \( u_i \)’s one-hop neighbors set is denoted as \( \mathcal{N}_i = \{ u_{i,1}, \ldots, u_{i,N_i} \} \), where \( N_i \) is the number of one-hop neighbors of user \( u_i \). We aggregate all documents published by user \( u_i \) as a corpus \( \mathcal{W}_i \). In CBOW [Mikolov et al., 2013a] based SWE model, given a sequence of training words, the first objective is to minimize the negative log-likelihood:

\[
\mathcal{J}_1 = - \sum_{i}^{N} \sum_{w_j \in \mathcal{W}_i} \log P(w_j|\mathcal{C}(w_j, u_i)), \tag{1}
\]

where \( w_j \) is the predicted word and \( \mathcal{C}(w_j, u_i) \) is a collection of context words around \( w_j \). To apply personalization to a word \( w_j \), SWE represented a word as \( w_j^{(i)} = w_j + u_i \), where \( w_j \in \mathbb{R}^d \) is the global word embedding and \( u_i \in \mathbb{R}^d \) is the local user embedding for user \( u_i \). The representation of context words \( \mathcal{C}(w_j, u_i) \) is \( \{ w_j^{(i-c)}, \ldots, w_j^{(i+c)} \} \), where \( c \) is the half window size.

A socialized regularization term is added to consider the second phenomenon:

\[
\mathcal{J}_2 = \sum_{i}^{N} \sum_{u_j \in \mathcal{N}_i} \frac{1}{2} \| u_i - u_j \|_2^2, \tag{2}
\]

where \( u_j \) is a friend of user \( u_i \). This regularization aims to force user vectors between friends to be similar.

By combining two parts, the final objective of SWE is

\[
\mathcal{J} = \mathcal{J}_1 + \lambda \mathcal{J}_2 \quad \text{s.t.} \quad \forall u_i, \ r_1 \leq \| u_i \|_2 \leq r_2, \tag{3}
\]

where \( \lambda \) is a trade-off parameter, \( r_1 \) and \( r_2 \) are a lower bound and upper bound for \( u_i \)’s \( L_2 \)-norm respectively. \( r_1 \) can avoid the situation where user embeddings might collapse to zero vectors and thus SWE will degenerate to word2vec [Mikolov et al., 2013a] while \( r_2 \) can prevent user embeddings dominating global word embeddings.

In SWE, when a document published by \( u_i \) is observed, \( u_i \) will be updated due to \( \mathcal{J}_1 \), and user embeddings of \( u_i \)’s friends \( \mathcal{N}_i \) will also be updated due to \( \mathcal{J}_2 \). But users outside this one-hop social circle will not be updated until one of texts published by themselves or their one-hop neighbors is observed.


Algorithm 1 SWE with regularization using Random Walks.

Input: User set \( U \) = \{\( u_1, u_2, \ldots, u_N \)\}, where each user has a corpus \( V_{ui} = \{d_{i1}, \ldots, d_{iM_i}\} \) and \( M_i \) is the number of documents written by \( u_i \), maximum iteration \( T \), learning rate on Eq. (1) \( \eta_1 \), learning rate on Eq. (2) \( \eta_2 \), trade-off parameter \( \lambda \), \( n_i \) paths that random walks generate for user \( u_i \), path length \( l \), return parameter \( p \), in-out parameter \( q \), restart rate \( \alpha \), bias weight \( \beta \).

for \( \text{iter} = 1 \) to \( T \) do
  for all \( u_i \in U \) do
    for all \( d_{i,j} \in V_{ui} \) do
      \( w_k := w_k - \eta_1 \cdot \frac{\partial c_i}{\partial w_k} \)
      \( u_k := u_k - \eta_1 \cdot \frac{\partial c_i}{\partial u_k} \)
    end for
    \( \text{walks} = \text{RandomWalk}(u_i, n_i, l, p, q, \alpha, \beta) \) following Eqs. (5)-(7)
    for \( \text{walk} \in \text{walks} \) do
      SocialRegularization(walk, \( \eta_2 \), \( \lambda \))
    end for
    for \( \text{walk} \in \text{walks} \) do
      Algorithm 2
    end for
  end for
end for

3.2 Random Walk based Social Regularization

As we explained in the introduction, there are two major problems with the above social regularization framework: implicit modeling on the transitivity and lack of concern of users with fewer friends. To remedy the problems, we propose to augment the social regularization with a random walk based approach. Intuitively, instead of imposing a regularizer within the one-hop social circle, we sample a set of random walks starting from the user. Then we impose the regularizer over all the sampled users in a path to explicitly model the transitivity. To emphasize the users with fewer friends, we also propose a biased random walk to sample more these users in the path. So we still follow the SWE framework but use random walks based regularization.

Here, we only consider random walk methods on the unweighted and undirected graph \( G = (V, E) \). Given a source node \( s \), random walk methods aim to generate a walk with fixed length \( l \). Let \( c_i \) denote the \( i \) th node in the walk, starting with \( c_0 = s \). Suppose a random walker has just traversed node \( c_{i-1} \) and will probabilistically move to \( c_i \). The transition probability \( P(c_i|c_{i-1}, c_{i-2}) \) can be computed as follows,

\[
P(c_i|c_{i-1}, c_{i-2}) = \begin{cases} \pi_{c_i|c_{i-1}, c_{i-2}} w_{c_{i-1}c_i}, & \text{if } (c_{i-1}, c_i) \in E, \\ 0, & \text{otherwise} \end{cases}
\]

where \( Z \) is a normalizing factor, and \( \pi_{c_i|c_{i-1}, c_{i-2}} \) is the unnormalized transition probability to \( c_i \) given \( c_{i-1} \) and \( c_{i-2} \), and \( w_{c_{i-1}c_i} \) is the weight of edge \((c_{i-1}, c_i)\). In an unweighted graph, \( w_{c_{i-1}c_i} = 1 \). We assume our random walk is a Markov chain with stationary transition probabilities. Based on the transition probability \( P(c_i|c_{i-1}, c_{i-2}) \), we can sample a sequence of nodes, starting with \( c_0 = s \).

After generating node sequences using random walks, we apply the social regularization as Eq. (2) to users. The detailed algorithm is shown in Algorithms 1 and 2. Different from SWE, for each node in generated walks, a node in the path will be updated by its adjacent nodes. In this way, we can model the transitivity more explicitly than SWE.

Now the remaining problem is how to formulate the unnormalized transition probability \( \pi_{c_i|c_{i-1}, c_{i-2}} \). We will describe it in the following to show differences in three random walk methods.

First-order Random Walk

In the first-order random walk (FRW), a random walker moves to next node based on the last node. The unnormalized transition probability \( \pi_{c_i|c_{i-1}, c_{i-2}} = \pi_{c_i|c_{i-1}} \) is computed as follows,

\[
\pi_{c_i|c_{i-1}} = \begin{cases} 1, & \text{if } (c_{i-1}, c_i) \in E, \\ 0, & \text{otherwise} \end{cases}
\]

This method will encourage a very deep walk, which means the random walker tends to go far away from the source node. From the perspective of propagation of language use, the start user might have little influence on the users who are far away from him/her. On the contrary, the start user will have more impacts on his/her close neighbors. This random walk method is not efficient as it tends to sample a lot of remote users.

Second-order Random Walk

In the second-order random walk (SRW), a random walker moves to next node based on the last node and the second last node. The unnormalized transition probability introduced in [Grover and Leskovec, 2016] is computed as follows,

\[
\pi_{c_i|c_{i-1}, c_{i-2}} = \begin{cases} \frac{1}{p}, & \text{if } d_{c_{i-2}c_i} = 0 \\ \frac{1}{q}, & \text{if } d_{c_{i-2}c_i} = 2 \\ 1, & \text{if } d_{c_{i-2}c_i} = 1 \end{cases}
\]

where \( d_{c_{i-2}c_i} \) is the shortest path distance between nodes \( c_{i-2} \) and \( c_i \). \( p \) is the return parameter controlling the likelihood of immediately revisiting last node \( c_{i-2} \) in the walk, and \( q \) is the in-out parameter controlling the walker to explore remote friends or close friends. When \( p < \min(1, q) \), the walker tends to revisit the last node. When \( q < \min(1, p) \), it tends to explore remote neighbors. When \( 1 < \min(p, q) \), it tends to explore mutual friends.
Biased Second-order Random Walk

According to update mechanism of SWE, the fewer friends a user has, the fewer update opportunities he/she can get. To tackle this problem, we propose a biased second-order random walk (BRW) to sample more users who have fewer friends by introducing a bias coefficient to adjust transition probabilities. The bias coefficient is associated with the number of friends of a user. We define the unnormalized transition probability \( \pi'_i|e_{i-1}, e_{i-2} \) as follows,

\[
\pi'_i|e_{i-1}, e_{i-2} = \varphi(c_i) \cdot \pi_{c_i|e_{i-1}, e_{i-2}},
\]

(7)

where \( \varphi(\cdot) \) is the bias coefficient and \( \pi_{c_i|e_{i-1}, e_{i-2}} \) is the same as Eq. (6). The transition probability is biased by a factor of \( \varphi(\cdot) \) which differs from one node to another. We define \( \varphi(x) \) as follows,

\[
\varphi(x) = \frac{1 + \beta}{d_x + 1} \cdot \frac{(1 + \beta) + 1}{1 + \beta},
\]

(8)

where \( d = |E|/|V| \) is averaged number of friends of a social network, \( d_x \) is the number of friends of node \( x \), and \( \beta \) is a parameter to adjust the shape of \( \varphi(x) \) (see Figure 1(a)).

When \( d_x < d \), we have \( \varphi(x) > 1 \), which means when the number of friends of node \( x \) is below the average, the walk tends to move to \( x \) with a larger probability than before. When \( d_x > d \), we have \( \varphi(x) < 1 \), which means when the number of friends of node \( x \) is above the average, the walk tends to move to \( x \) with a smaller probability than before. When \( d_x = d \), we have \( \varphi(x) = 1 \), which is exactly the same with the aforementioned probability in the second-order random walk. In Figure 1(a), we can see that smaller \( \beta \) leads to larger adjustment and encourages the random walker to move to users who have fewer friends with larger probability. In particular, when \( \beta = 0 \), \( \varphi(x) = \frac{2}{d_x + 1} \) is the ratio of the harmonic mean to the number of friends of node \( x \).

In practice, methods \[Perozzi et al., 2014; Grover and Leskovec, 2016\] using random walks will generate \( n \) paths with fixed length \( l \) at a time. To sample more users who have fewer friends, we allow the number of paths to vary with the number of friends. We define the number of paths which associated with a source node \( s \) as follows,

\[
n_s = \varphi(s) \cdot n,
\]

(9)

where \( \varphi(\cdot) \) refers to the same definition in Eq. (8). In this way, a user with fewer friends will generate more paths.

As indicated by \[Pan et al., 2004\], the restart mechanism is a useful technique in random walks. It is reasonable to introduce the restart mechanism to our biased random walk method since it will encourage more close neighbors. The effect of restart is similar to a small value of \( p \) in SRW. The difference is that the restart is more flexible since a walker can move back to source node \( s \) at any movement no matter how far away from it. Before a walker makes each movement, restart mechanism allows the walker to determine whether to move back to source node \( s \) with a probability as follows,

\[
\alpha_s = \varphi(s) \cdot \alpha,
\]

(10)

where \( \varphi(\cdot) \) refers to the same definition in Eq. (8), and \( \alpha \) is the initial restart rate. Compared with opinion leaders in social media, users who have fewer friends might have fewer impacts on the users far away from him/her, so it is reasonable to allow them to restart with larger probability.

4 Experiments

In this section, we show experiments to demonstrate the effectiveness of random walk based social regularizations for word embeddings.

4.1 Datasets

We conducted all experiments on Yelp Challenge datasets which provide a lot of review texts along with large social networks. At Yelp, people can write reviews for restaurants, bars, etc., and can follow other users to share information. From the simple statistics shown in Table 1, we can see Yelp Round 10 has more reviews and users than Yelp Round 9.

4.2 Experimental Settings

We randomly split data to be 8:1:1 for training, developing, and testing identically for both training word embeddings and downstream tasks, in which we ensure that reviews published by the same user can be distributed to training, development, and test sets according to the proportion. All the following results are based on this fixed segmentation. For SWE, we use the code released by \[Zeng et al., 2017\].

We use CBOW \[Mikolov et al., 2013\] to train word embeddings for all methods. To make a fair comparison, we set the hyper-parameters to be the same as the SWE. For constraint \( r_1 \), we empirically set it to \( r_1^2 = 0.2r_2^2 \).

The social regularization using first-order random walks (SR-FRW), using second-order random walks (SR-SRW), and using biased (second-order) random walks (SR-BRW) involve extra hyper-parameters and they have some hyper-parameters in common. We train embeddings on the training set and search hyper-parameters on the development set using sentiment classification. We use the following strategy to reduce time on searching. We perform grid search for SR-FRW, SR-SRW, and SR-BRW in sequence to determine optimal hyper-parameters. Once one method performs grid search on some hyper-parameters and get optimal values, other methods will not search them again. Hence, we will not

---

1 https://www.yelp.com/dataset_challenge
Table 2: Statistics of one-fifth of Yelp Round 9 and 10 data.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Yelp Round 9</th>
<th>Yelp Round 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Users</td>
<td>16,768</td>
<td>18,976</td>
</tr>
<tr>
<td>#Avg. Reviews</td>
<td>11.68</td>
<td>11.63</td>
</tr>
<tr>
<td>#Avg. Friends</td>
<td>27.83</td>
<td>27.48</td>
</tr>
</tbody>
</table>

Table 3: Sentiment classification accuracies (in %) on one-fifth development and test sets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Yelp Round 9 Dev</th>
<th>Yelp Round 9 Test</th>
<th>Yelp Round 10 Dev</th>
<th>Yelp Round 10 Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>W2V</td>
<td>58.98</td>
<td>58.90</td>
<td>59.79</td>
<td>60.09</td>
</tr>
<tr>
<td>SWE</td>
<td>59.28</td>
<td>59.12</td>
<td>60.11</td>
<td>60.31</td>
</tr>
<tr>
<td>SR-SRW</td>
<td>59.28</td>
<td>59.32</td>
<td>60.24</td>
<td>60.45</td>
</tr>
<tr>
<td>SR-BRW</td>
<td>59.28</td>
<td><strong>59.44</strong></td>
<td>60.32</td>
<td><strong>60.53</strong></td>
</tr>
</tbody>
</table>

Table 4: Statistics of head users and tail users in the one-fifth of the training set. “Perc.” means the percentage of head/tail users sampled in the random walk paths.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Yelp Round 9 Head</th>
<th>Yelp Round 9 Tail</th>
<th>Yelp Round 10 Head</th>
<th>Yelp Round 10 Tail</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Users</td>
<td>3,631</td>
<td>13,137</td>
<td>4,123</td>
<td>14,853</td>
</tr>
<tr>
<td>#Avg. Reviews</td>
<td>26.97</td>
<td>7.45</td>
<td>27.83</td>
<td>7.43</td>
</tr>
<tr>
<td>#Avg. Friends</td>
<td>72.59</td>
<td>19.99</td>
<td>72.01</td>
<td>19.99</td>
</tr>
<tr>
<td>Perc. in SR-SRW</td>
<td>56.20%</td>
<td>43.80%</td>
<td>56.68%</td>
<td>43.32%</td>
</tr>
<tr>
<td>Perc. in SR-BRW</td>
<td>44.03%</td>
<td>55.97%</td>
<td>45.94%</td>
<td>54.06%</td>
</tr>
</tbody>
</table>

4.3 Sentiment Classification

In this section, we evaluate different regularizations on sentiment classification for Yelp reviews. As shown in [Yang and Eisenstein, 2017], taking language variance and linguistic homophily into consideration can help sentiment analysis. For example, words such as “good” can indicate different sentiment ratings depending on the author. Hence, it is a valid task to demonstrate the effectiveness of socialized word embeddings. In Yelp, users can write text reviews to describe his/her feelings and opinions towards businesses and then give a star rating. We take the averaged word embeddings of all words (except stop words) in a review as input, and then use the one-vs-rest logistic regression implemented by LibLinear\(^2\) to predict the ratings scaled from 1 to 5.

We compare our social regularizations with two baseline embedding methods, namely, W2V and SWE. For efficiency, we randomly select one-fifth of the training data to train a logistic regression classifier. The statistics of one-fifth training data are shown in Table 2. In Table 3, results suggest that SWE has better performance than W2V. Social regularizations using random walks outperform SWE, and SR-BRW performs the best among all social regularizations.

As pointed out by [Zeng et al., 2017], previous studies usually preprocessed the data on their methods on partial data containing sufficient user information [Tang et al., 2015; Chen et al., 2016]. Hence, we also report the performance on partial data. For efficiency, we still use the same one-fifth of the training data as our training set. But we perform the same preprocessing steps as [Zeng et al., 2017] to obtain head and tail users’ data. Here for head users, we mean users published a lot of reviews, while tail users publish less. The statistics of head and tail users are shown in Table 4.

\(^2\) https://www.csie.ntu.edu.tw/~cjlin/liblinear

From the table, we can see that head users tend to publish more reviews and have more friends than tail users.

We conduct experiments using the head and tail subsets as training data respectively. To evaluate the significance of the improvements, we run experiments ten times on randomly sampled 60% of the one-fifth training data to report mean, standard deviation, and t-test results. The results are shown in Table 5. We can see that both random walk based methods outperform SWE on both head and tail data. It reflects that our explicit modeling on the transitivity is better. From statistics in Table 4, compared with SR-SRW, the proportion of tail users sampled in SR-BRW increases, which shows that the biased coefficient and restart can help to sample more tail users. Moreover, in Table 5, SR-BRW outperforms SR-SRW, so we can conclude that sampling more tail users can improve performance. It is interesting that improvements in head users are more significant than in tail users. For example, in Yelp 10, SR-BRW improves SWE by 1.0% in head users but 0.4% in tail users. The reason might be that SWE will enforce all one-hop neighbors of a head user to be similar to the head. But in reality, head users might have many friends, e.g., 1,000. Forcing all one-hop neighbors to be similar to the head is unreasonable. In our method, only one-hop neighbors sampled by paths are forced to be similar to the head user, others can still maintain their personal characteristics of language use. But tail users do not have this problem.

4.4 Parameters Sensitivity

We first perform a grid search over \(n\) and \(l\) for SR-FRW. In Figure 1(b), areas highlighted by blue circles represent satisfying accuracies. When \(n = 80\), accuracies are consistently high, which means \(l\) is insensitive when \(n = 80\). But we cannot find such a value of \(l\) where accuracies always are satisfactory with varying \(n\), which indicates \(n\) is more sensitive than \(l\).

Hyper-parameter \(p\) and \(q\) work together to control exploration strategies, so we perform grid search over \(p\) and \(q\) for SR-SRW. In Figure 1(c), many light areas are connected, e.g., the area within the blue square, which means we can find a good combination easily in a wide range.

Figure 1(d) shows that light areas are very large, which means it is easy to search satisfying hyper-parameters. When \(\alpha = 0.1\) or \(\beta = -0.9\), accuracies are consistently high, which means when we fix one hyper-parameter to a certain value, the other one can be insensitive. Compare to \(\alpha = 0\), the accuracies of \(\alpha = 0.1\) are consistently better, which indicates the restart is effective as long as \(\alpha\) falls in the right range.
Table 5: Mean and standard deviation of accuracies (in %) on full one-fifth test data. Overall / Head / Tail means training on randomly sampled 60% full one-fifth / only head users' / only tail users' data. The marker * refers to p-value < 0.0001 in t-test compared with SWE.

Table 6: Comparison of our model and other baseline methods in accuracy (%) on user attention based deep learning for sentiment analysis.

4.5 User Vectors for Attention

For document-level sentiment classification on Yelp data, the most interesting work [Chen et al., 2016] shows that by using a user attention vector, accuracies can be improved comparing with original models. This way is consistent with SWE framework because it embeds not only words but also users. It is natural to adopt user vectors from methods under SWE framework (SWEs), i.e., SWE, SR-SRW, and SR-BRW, as user attention vectors.

We design three settings in experiments to demonstrate the effectiveness, namely, without user attention, using the user vector from SWEs as fixed attention, and trainable attention. From the perspective of learning a user attention, fixed attention is an unsupervised method since it is trained by one of the methods under the SWE framework while trainable attention is a supervised method since training an attention vector requires supervision of rating scores.

We compare three settings using hierarchical convolutional neural networks (HCNN) and hierarchical long short term memory recurrent neural networks (HLSTM) [Tang et al., 2015; Chen et al., 2016]. For without attention and trainable attention, we use word embeddings from W2V as input. For fixed attention, we use global word embeddings from SWEs as input. We train on the same one-fifth training data of Yelp Round 9 and Round 10 and evaluate on the same one-fifth development and test sets.

In Table 6, these unsupervised methods even outperform the supervised one, which demonstrates the effectiveness of user vectors under SWE framework. Although fixed attention is unsupervised, it uses rich information from social network while trained attention does not, which might explain the good performance of unsupervised methods. We believe that user and word embeddings trained by unsupervised methods can enhance performance in social network related text understanding tasks if mining rich social information. Moreover, random walk based methods outperform SWE, which demonstrates the effectiveness of explicit modeling of the transitivity. It is interesting that in this experiment SR-BRW could be a little worse than as SR-SRW. This may be because in deep learning framework, the other parameters can be trained based on the fixed user embeddings. SR-SRW without bias may keep more fidelity to the user graph, which results in better training for the other parameters. On the contrary, SR-BRW modified the sampling process in the random walk, which may be better for linear logistic regression since it uses averaged word embeddings as input which is less flexible to learn the parameters.
5 Conclusion
In this paper, we adopt random walk based social regularizations to explicitly model the transitivity of language use. Moreover, we propose a biased random walk based social regularization to sample more users who have fewer friends. We demonstrate the effectiveness of our random walk based social regularizations. One important future work would be how to use more social information, not just the number of friends, to improve socialized word embeddings.

Acknowledgements
This paper was supported by China 973 Fundamental R&D Program (No. 2014CB340304) and the Early Career Scheme (ECS, No. 26206717) from Research Grants Council in Hong Kong. Ziqian Zeng has been supported by the Hong Kong Ph.D. Fellowship. We also gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan GPU used for this research.

References


