Tackling the Achilles Heel of Social Networks: Influence Propagation based Language Model Smoothing

Rui Yan
Baidu Inc.
No. 10, Shangdi 10th Street,
Beijing 100085, China
yanrui02@baidu.com

Ian E.H. Yen
Dept. of Computer Science
University of Texas at Austin
Austin, TX 78712, USA
ianyen@cs.utexas.edu

Cheng-Te Li
Academia Sinica
No 128, Academia Road,
Taipei 11529, Taiwan
ctli@citi.sinica.edu.tw

Shiqi Zhao
Baidu Inc.
No. 10, Shangdi 10th Street,
Beijing 100085, China
zhaoshiqi@baidu.com

Xiaohua Hu
College of Info. Sci. and Tech.
Drexel University
Philadelphia, PA 19104, USA
xh29@drexel.edu

ABSTRACT

Online social networks nowadays enjoy their worldwide prosperity, as they have revolutionized the way for people to discover, to share, and to distribute information. With millions of registered users and the proliferation of user-generated contents, the social networks become "giants", likely eligible to carry on any research tasks. However, the giants do have their Achilles Heel: extreme data sparsity. Compared with the massive data over the whole collection, individual posting documents, (e.g., a microblog less than 140 characters), seem to be too sparse to make a difference under various research scenarios, while actually they are different. In this paper we propose to tackle the Achilles Heel of social networks by smoothing the language model via influence propagation. We formulate a socialized factor graph model, which utilizes both the textual correlations between document pairs and the socialized augmentation networks behind the documents, such as user relationships and social interactions. These factors are modeled as attributes and dependencies among documents and their corresponding users. An efficient algorithm is designed to learn the proposed factor graph model. Finally we propagate term counts to smooth documents based on the estimated influence. Experimental results on *Twitter* and *Weibo* datasets validate the effectiveness of the proposed model. By leveraging the smoothed language model with social factors, our approach obtains significant improvement over several alternative methods on both intrinsic and extrinsic evaluations measured in terms of perplexity, nDCG and MAP results.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing—*Text mining*; J.4 [**Social and Behavioral Sciences**]: Miscellaneous

WWW 2015, May 18–22, 2015, Florence, Italy.
ACM 978-1-4503-3469-3/15/05.
http://dx.doi.org/10.1145/2736277.2741673.

General Terms

Algorithms, Experimentation, Performance

Keywords

Language model smoothing; influence propagation; social network

1. INTRODUCTION

Along with the prosperity of Web 2.0, online social networks have emerged as a brand new medium in spotlight and revolutionized the way for people to discover, to share and to propagate information via peer-to-peer interactions [13]. Among the popular websites which provide social services, big names cover nearly every corner of online social life and events: *Facebook*[1] encourages photo/video/blog sharing and leaving comments among friends, *Twitter*[2] provides succinct, fast spreading microblogs for news feeds, *Foursquare*[3] is a location based mobile social network which provides comments generally about places of interest.

In this sense, we are living in an era with "giants": with incredibly large number (in billions [1]) of affiliated users and the proliferation of user-generated contents, the online social networks are the mighty giants of our times, plausibly invincible and eligible to carry on any kind of research tasks. In fact, people have conducted a series of studies on social networks, such as search and retrieval [31], information recommendation [36, 5], link prediction [12, 7] and summarization [16, 38].

Although powerful as giants, social networks still suffer from their weakness: extreme sparsity. For many textual related researches such as retrieval, mining or recommendation etc., accurate estimation of document language models are likely to be an essential part to the success of new methods and models. More observed data generally allow people to establish a more accurate statistical model. However, due to the real-time propagation on social networks, the posted documents are sometimes officially limited within a restricted length to facilitate fast spreading (e.g., 140 characters per tweet on Twitter). In this case, we have to estimate the language model based on a small piece of texts (e.g., a tweet), while given

[1]https://www.facebook.com
[2]https://www.twitter.com
[3]https://www.foursquare.com

limited data sampling, the language model estimation usually encounters with severe zero count problem: the maximum likelihood estimator would give unseen terms a zero probability rather than a reliable approximation. Therefore, sparsity actually becomes the Achilles Heel of social networks for texts related studies, and we aim at tackling the bottleneck.

Statistical language models have attracted much attention in the information retrieval community due to their solid theoretical background as well as their success in a variety of tasks [25, 14]. Language model smoothing is proposed to address the sparsity problem, and has been demonstrated to affect retrieval performance significantly [40]. The quality of textual related tasks heavily relies on proper smoothing of the document language model. Although much work on language model smoothing has been investigated, the intuitions behind are generally related to two major concerns: 1) semantic association [34, 18, 30] and 2) positional proximity [41, 19, 34]. However, for social networks, there is much more information hidden in social factors, which could be utilized to measure language model smoothing. Therefore, social interaction could be a third direction to investigate better language modeling.

The key idea for language model smoothing is to propagate term counts via certain ways of projection to other places where they originally do not really exist. Hence we propose a socialized factor graph model to investigate various factors which could have impacts on language models, and measure the influence propagated along the factor graph. According to the influence estimated, we propagate the term occurrence in discounted counts and hence smooth the original language models. To the best of our knowledge, we are the first one to mapping social influence onto the textual dimension to facilitate socialized language model smoothing. Our 1st contribution is to fuse a series of social attributes with textual information and form an integrated objective function to balance both. Intuitively, a smoothed language model should enhance the coherence between terms with semantic association, and analogously for those through social interactions. In other words, the terms that are associated via close friends based on the social linkage could have similar (or smoothed) probabilistic distributions: stronger social ties could also lead to more similar smoothed language models.

Another main technical challenge lies in how to define the attributes, factors and formulate the propagation functions to model the joint probabilistic factor graph. We explore several different factors captured from posting document pairs and user pairs, and evaluate their dependencies on each other. To be more specific, we have examined features such as text similarity, text quality, social status and social interactions and so on, and then grouped them as factor functions. Factor functions are finally formulated into one objective function to calculate the estimated influence and hence to smooth the language model accordingly.

In this paper, we define the problem of Socialized Language Model Smoothing (SLMS) based on the factor graphs. We evaluate the effectiveness of our proposed language model smoothing method using two different social network datasets from *Twitter* and *Weibo*[4]. Both of them are mainstream microblogs, one in English and the other one in Chinese. We apply intrinsic evaluation measured in perplexity and extrinsic evaluation measured in nDCG and MAP in our experimental setups. Experimental results show that our proposed influence propagation based language model smoothing on factor graphs consistently outperforms the baseline smoothing methods, and the improvement is significant, which indicates the effectiveness of our approach. In other words, the social factors along with the textual information in combination could to some extent tackle the Achilles Heel of social networks.

The rest of the paper is organized as follows. We start by introducing the problem definition and then the influence propagation based language model smoothing on factor graphs, using textual and social information in combination. We describe the experiments and evaluation in Section 4, review previous works in Section 5 and finally draw the conclusions.

[4]https://www.weibo.com

2. PROBLEM FORMULATION

In this section, we introduce some notations and representations necessary for language modeling and smoothing on social networks, and then elaborate the problem of Socialized Language Model Smoothing (SLMS) via factor graph model.

Definition 1. (Document and Collection.) Given a posting document d_0 to smooth, we have a whole document collection $D = \{d_1, d_2, \ldots, d_n\}$ as the background set to smooth d_0.

In the context of web documents on social networks, e.g. Twitter or Weibo, etc., each posting document is written by a particular user. Compared with traditional documents that are defined merely based on plain texts, these posting documents are associated with more interesting elements. For instance, social networks support particular *relationships* established among users, i.e., *follower-followee* on Twitter. Also, there are social interactions spread between the posting documents, e.g., *replying*, *sharing* and *reposting* documents. We hence have a hidden network structure behind the document collection. User activities implicitly reflect more information behind the documents to model the textual properties. We believe that the social relationships would help to better describe the text information. Thus, integrating the document contents and social information can disclose a more accurate estimation of the document language model to smooth. In this paper, we mainly employ the microblogging service as the basis for our study and hence make a good utilization of its characteristics for illustration.

Specifically, given a posting document d_i, and its associated user u_i, together with the associated user networks, we give the following definition of socialized augmentation network.

Definition 2. (Socialized Augmentation Network.) We have the heterogeneous graph of posting documents and their corresponding users. We denote the whole graph G as a collection of nodes V and edges E, and have G=(V, E)=$(V_u, V_d, E_u, E_d, E_{u,d})$. It is obvious to see that there are three kinds of relationships associated: 1) (V_d, E_d) is a weighted directed graph between posting document pairs, where $V_d = \{d_i | d_i \in D\}$ is the posting collection with a size of $|D|$, and E_d is the set of relationships, indicating the influence from source postings to the target postings, which is our goal to estimate; 2) (V_u, E_u) is also a weighted directed graph indicating the social ties between users. $V_u = \{u_i | u_i \in V_u\}$ is the set of users with a size of $|V_u|$. E_u is established by the social behavior among users, which will be described in Section 3.2; 3) $(V_{u,d}, E_{u,d})$ is the unweighted bipartite graph representing authorship of posting documents and users. $V_{u,d}$ = $V_u \cup V_d$. Edges in $E_{u,d}$ connect each posting document with all of its authors and help mapping from social dimension to textual dimension. Usually a posting document d is written by only one user u.

Now we give a formal definition of our proposed SLMS problem:

Input: Given the entire document set D, and the socialization augmentation networks, we aim to smooth the language model of the target document, denoted as $P(w|d_0)$, based on the influence from all other documents d_i where $\{d_i|d_i \in D\}$.

Output: The smoothed language model of $P(w|d_0^+)$ for every original document d_0.

With these preliminaries, we show that relationships from document pairs, user pairs and user-document pairs can be all formulated and hence mapped as the instances on the factor graph using potential functions to form a combined objective function.

3. METHODOLOGY

In this section, we propose a socialized factor graph to compute influence propagation, and formulate the socialized language model smoothing problem into a unified learning framework. The model simultaneously incorporates all resources (i.e., texts and social information) into the augmentation contexts to generate high-quality estimation for document language models after smoothing.

The socialized language model smoothing problem contains several sub-problems: 1) document pair influence measurement, 2) user pair influence measurement, and 3) variable pair influence measurement. We aim to quantify the correlation between document pairs based on semantic association derived from contents, while also we intend to augment the pairwise relationship between documents from the interactions of users on social level. Social contexts contain rich information about the document pairs. We also analyze the dependency of variables on each other based on the same authorship. Finally, we use the estimated influence propagation on the documents to smooth the original language model, and then apply to a series of related research tasks. The framework is illustrated in Figure 1, and the details are explained later.

3.1 Proposed Model

Factor graph assumes observation data are cohesive on both features and relationships between each other [11]. It has been successfully applied in many applications, such as social influence analysis [28, 27], social relationship mininig [7, 33], and linked data disambiguation [12, 32]. In this work, we formulate the social features and associated networks into the factor graph model, which is shown in Figure 1. Given the document pairs, let E_d be the pairwise links between two posting documents, and E_u be the user social ties. The input of the factor model is the whole document collection and the socialized augmentation contexts, and a target document d_0 to smooth. Both pairs are digested into the attribute factors, which are observable. There is also a set of hidden variables $Y = \{y_i\}_{i=1}^n$, representing the influence inferred from the observed pairs and coordination among the hidden variables.

We define two feature functions in the proposed factor model: *attribute factor* and *dependency factor*.

• **Attribute Factor**. The influence from a posting document to another could be estimated by some attributes (represented as $\mathbf{x}$), which refer to features that are inherent to the documents and their authors. In general, we define a series of features for the document pairs and user pairs. These features include the textual based contents such as text quality, similarity and popularity, as well as the social ties such as user relationships, interactions, authoritativeness and so on. Details of the defined features are given in the next section. We use the feature function $f(y_i, \mathbf{x}_i)$ to represent the posterior probability of label y_i given $\mathbf{x}_i$ contained in the pairwise information among the heterogenous nodes.

We define this type of potential function as a linear exponential function and to estimate the significance of each feature, we intro-

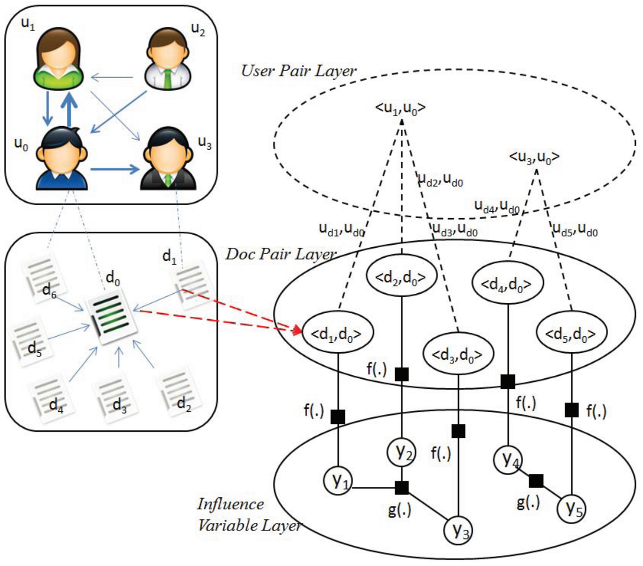


Figure 1: Graphical representation of the socialized augmentation factor graph. The left part shows the heterogeneous graph, consisting of the document collection and users with social ties. The right part indicates the decomposable factor graph. The top layer shows the user pairs, which could be instantiated into several user pairs identified by different document pairs on the middle layer. The lower layer indicates the influence to estimate between document pairs. There are factor functions within the same layer ($g(.)$) and factor functions across layers ($f(.)$). In the final step, The language models are smoothed based on the influence estimated on the lower layer. Some links between documents and users are omitted to keep concise.

duce a vector of weight variable α for each feature c, and formally we could define the attribute factors as the local entropy as follows:

$$f_i(y_i, \mathbf{x}_i) = \frac{1}{Z_\alpha} \exp\{\sum_c \alpha_c f_{i,c}(y_i, x_{i,c})\} \quad (1)$$

where $x_{i,c}$ is the c-th attribute to calculate the influence. $f_c(.)$ is the function to calculate the result from the c-th feature and α_c is the corresponding weight. Z_α is a normalization factor.

• **Dependency Factor**. As proposed, we introduce factors that are capable of handling multiple hidden variables on the variable layer, to characterize the dependencies among the posting documents generated by the same user. The dependency factor is to propagate the social influence among all posting documents from the same user. The heuristics is that if document d_0 is influenced by document d_i. It is highly possible to be influenced by the document d_k from the same user, which is actually a belief propagation.

To capture this intuition, we define the potential function to model the correlation of a candidate variable y_i with another candidate variable y_k in the factor graph. The function is defined as:

$$g(y_i, y_k) = \frac{1}{Z_{ik,\beta}} \exp\{\beta_k g_k(y_i, y_k)\} \quad (2)$$

where $g(.)$ is a function indicating whether two variables are correlated or not. Note that if $g(y_i, y_k)$=0, there will be no dependency between the two variables. In other words, the two variables are not correlated. Actually we can group the document set into clusters and each cluster is associated with one user, and we use Y_i to denote the cluster where y_i is in. Hence, for $\forall y_k \in Y_i$, y_i has dependency on y_k. We further revise Equation (2) as:

$$g_i(y_i, Y_i) = \prod_{y_k \in Y_i} g(y_i, y_k) = \frac{1}{Z_\beta} \exp\{\sum_{y_k \in Y_i} \beta_k g_k(y_i, y_k)\} \quad (3)$$

Again, Z_β is the normalization factor.

In this way, the influence estimation could be viewed as a sequence labeling process [38], i.e., the judgment on a certain pair is affected by the "similar" pairs (i.e., the documents written by the same user in this work), which is exactly the intuition for language model smoothing [21].

• **Objective Function**. In general, the attribute factors capture the potential influence from document/user pairs and the dependency factor captures correlations between variables. In Equation (1) we define the features $f_c(.)$ for all attributes, where α_c is the corresponding weight. In Equation (3), we define the correlation where β_k indicates the weights. Let Y and X be the sets of candidate variables and attribute variables respectively, we define a conditional probability encoded within the factor graph model by multiplying all potential functions and can be written as

$$P_\theta(Y|X) = \prod_i f_i(y_i, \mathbf{x}_i) g_i(y_i, Y_i) \quad (4)$$

Hence, by integrating the defined factor functions, and also following the labeling assumption [10], we can define the following log-likelihood over all the undetermined labels of all instances, i.e., Y=$\{y_i\}_{i=1}^n$, where the objective function summing up likelihood:

$$\begin{aligned} \mathcal{O}(\theta) &= \log P_\theta(Y|X) \\ &= \sum_i \sum_c \alpha_c f_{i,c}(y_i, x_{i,c}) + \sum_{y_k \in Y_i} \beta_k g_k(y_i, y_k) - \log Z \end{aligned} \quad (5)$$

$Z = Z_\alpha Z_\beta$ is the normalization factor, which sums up the conditional likelihood of P_θ over all instances. θ is the collection of parameters indicating weights, i.e., $\theta = \{\alpha\} \cup \{\beta\}$.

Calculating the marginal distribution for each factor (in deriving the log-gradient of the objective function) requires a loopy sum-product inference algorithm. With the learned parameters, we may estimate an undetermined influence between document pairs in the test set by inferring the influence propagated and then smooth language models accordingly. The inference algorithm is introduced in section 3.3.

3.2 Function Definitions

Many features have been designed for document correlation and social associations among users in previous literature. In this paper, we investigate 8 features or factors. Besides, some features that are widely used in traditional text analysis, we also utilize several features extracted from users' social interactions, for instance, following behaviors, social status, and other statistics. We start from the feature definition first.

3.2.1 Attributes

We use the potential functions in the factor graph model to learn the potential influence for a document d_i to shade on d_0. Referring to Equation (1), we define the attribute functions as follows:

Text Similarity. It is intuitive that the textual similarity between two documents play an important role in language model smoothing [34, 18, 30]. Similar documents should be smoothed with higher weights since it is more consistent with their existing models. We use the cosine similarity between two unigram models:

$$f_{sim} = \frac{d_0 \cdot d_i}{||d_0|| ||d_i||} \quad (6)$$

Text Quality. We also measure the text quality of document d_i. It is not a good idea to smooth the target language model using a piece of text with low quality. Hereby we use the Out-Of-Vocabulary (OOV) ratio to measure the textual quality. The lower OOV ratio, the higher quality would be. Against the vocabulary from the official news corpora [42], OOV in microblogs often refers to a set of misspellings, informal terminologies, and irregular symbols.

$$f_{oov} = 1 - \frac{|\{w | w \in (d_i \cap \text{OOV})\}|}{|d_i|} \quad (7)$$

Technically, the measurement of *text quality* is not a pairwise function strictly between d_0 and d_i, but the criteria is indeed a practical indicator to decide whether or not to propagate the influence from d_i to d_0. We also include similar criteria for user pairs.

Posting Popularity. It is intuitive that a popular posting document is more likely to influence on many other posting documents. We use the aggregated numbers of social interaction (i.e., replies and retweets) as the approximation of popularity for d_i.

Social Status. Since micro-blogging service requires no reciprocal linkage among users, it is natural to assume that the social status is asymmetric between two users. A followee is more likely to influence the followers. This feature is represented by nominal values, e.g., '1' - the user of d_0 follows the user who writes d_i; '-1' - the user of d_0 is followed by the user who writes d_i; '0' - the two users have no direct follow behaviors.

User Similarity. We presume that people within the same social circle will have a larger probability to influence each other. Still, due to the asymmetry, we measure the Jaccard distance of the common followees of two users as their similarity. We use function $\mathcal{F}(u)$ to denote the social circle set for the user u. The $\mathcal{F}(.)$ of "followee" can be replaced by "followers" or extended to "friends".

$$f_{usim} = \frac{|\mathcal{F}(author(d_0)) \cap \mathcal{F}(author(d_i))|}{|\mathcal{F}(author(d_0)) \cup \mathcal{F}(author(d_i))|} \quad (8)$$

Interaction Strength. We also include the strength of interactions between the user pairs. It is possible that if two users have frequent social interactions, they are likely to share the writing preference on the term usage. Due to the asymmetrical social relationship, we only count the times for $author(d_0)$ to repost from $author(d_i)$ to measure how likely for the user to be influenced.

User Impacts. On social networks, some users are intrinsically have much larger influence on the others, e.g., sports stars, political celebrities, etc. Their words are usually copied, quoted and spreaded. There are many different ways to evaluate the user impacts, while we use the classic PageRank [23] scores to denote user impacts. The linkage based PageRank algorithm is quite suited to the scenario of user impact measurement. With a large number of in-links, the user is almost guaranteed to have high social impacts.

3.2.2 Dependency

As for the dependency function between candidate variables, referring to Equation (3), we define the function $g(.)$ for two candidate variables associated by the same user authorship in Y_i, and let the corresponding variables be y_i and y_k, respectively. The dependency function aims at encoding the influence propagation between posting documents from the same user, defined as follows.

$$\begin{aligned} g_k(y_i, y_k) &= \mathcal{I}\{author(d_i) = author(d_k)\} \times f_{sim}(d_i, d_k) \\ &= \mathcal{I}\{author(d_i) = author(d_k)\} \times \frac{d_i \cdot d_k}{||d_i||||d_k||} \end{aligned} \tag{9}$$

The boolean indicator $\mathcal{I}(.)$ is 1 when $author(d_i)$=$author(d_k)$ and 0 otherwise. We assume that two candidate variables will have higher correlation if they represent documents written by the same user. The correlation should also be affected by text similarity.

3.3 Model Inference

There are two scenarios for parameter estimation. If we do not have any labels of influence values among the candidate random variables, we can only propose to optimize the factor graph model via a two-step iterative procedure. At first, all of the model parameters (i.e., θ) are randomly initialized. Given the parameters, we infer the marginal probabilities of candidate variables. Then, given the marginal probabilities, we are able to evaluate experimental performance (Section 4.3) with respect to a set of held-out validation data, and search for better model parameters until convergence.

Unfortunately, this inference algorithm is way too indirect, depending on external evaluation metrics. Besides, it takes a longer time to converge. It is impossible to manually label influence values, but we do have partial labels available due to the special phenomenon of *repostings* on Twitter and Weibo! For each retweeted pairs, they have exactly the same contents and hence we label the influence as 1. With the partially labeled variables, we can now estimate the parameter configuration $\theta = \{\alpha, \beta\}$ to maximize the log-likelihood objective function $\mathcal{O}(\theta) = \log P_\theta(Y|X)$ approximately with gradient based methods, e.g., stochastic gradient descent [2].

$$\theta^* = \text{argmax}_\theta \ \mathcal{O}(\theta) \tag{10}$$

Algorithm 1: Learning algorithm on the factor graph

```
Input: a factor graph $G$, and the learning rate $\eta$.
Output: Estimated parameters $\theta = \{\alpha, \beta\}$.
Initialize all elements in $\theta \leftarrow 1$
begin
    repeat
        Compute potential function values S according to
        Equations (1)-(9)
        Run inference method using current $\theta$ to get $P_\theta(Y|X)$
        for $\{\alpha, \beta\} \in \theta$ do
            Compute gradient of $\frac{\partial \mathcal{O}(\theta)}{\partial \theta}$ using S according to
            Equation (13).
                $\frac{\partial \mathcal{O}(\theta)}{\partial \theta} = \mathbb{E}_{P_\theta(Y|Y^L)}S - \mathbb{E}_{P_\theta(Y)}S$
            Update parameter $\theta$ with the learning rate $\eta$:
                $\theta = \theta + \eta \cdot \frac{\partial \mathcal{O}(\theta)}{\partial \theta}$
    until Convergence;
```

Based on Equations (5), the probability distribution $P(Y|X)$ can be written in a more brief format as follows:

$$P(Y|X) = \frac{1}{Z}\exp\{\theta \sum_i s(y_i)\} = \frac{1}{Z}\exp\{\theta \cdot S\} \tag{11}$$

where $s(y_i) = (f_i(y_i, \mathbf{x}_i), g_i(y_i, Y_i))$ and $S = \sum_i s(y_i)$.

In this way, the objective function could be written as

$$\begin{aligned} \mathcal{O}(\theta) = \log P(Y^L|X) &= \log \sum_{Y|Y^L} \frac{1}{Z}\exp\{\theta S\} \\ &= \log \sum_{Y|Y^L} \exp\{\theta S\} - \log \sum_{Y} \exp\{\theta S\} \end{aligned} \tag{12}$$

where Y^L denotes the known labels and $Y|Y^L$ is a labeling configuration of Y inferred from Y^L. In order to maximize the objective function, we adopt the gradient decent method. We calculate the gradient for each parameter θ.

$$\begin{aligned} \frac{\partial \mathcal{O}(\theta)}{\partial \theta} &= \frac{\partial(\log \sum_{Y|Y^L} \exp\{\theta S\} - \log \sum_Y \exp\{\theta S\})}{\partial \theta} \\ &= \frac{\sum_{Y|Y^L} \exp \theta S \cdot S}{\sum_{Y|Y^L} \exp \theta S} - \frac{\sum_Y \exp \theta S \cdot S}{\sum_Y \exp \theta S} \\ &= \mathbb{E}_{P_\theta(Y|Y^L)}S - \mathbb{E}_{P_\theta(Y)}S \end{aligned} \tag{13}$$

$\mathbb{E}_{P_\theta(Y|Y^L)}S$ and $\mathbb{E}_{P_\theta(Y)}S$ are two expectations of S. The value of S can be obtained naturally using approximated inference algorithms, such as Loopy Belief Propagation (LBP) algorithm [39], Gibbs Sampling [4] or Contrastive Divergence [3]. One challenge here is that the graphical structure in the model can be arbitrary and may contain cycles, which makes it intractable to directly calculate the marginal distribution. We chose Loopy Belief Propagation due to its ease of implementation and effectiveness. It should be noted that the proposed ranked-margin algorithm can be exploited not just for graphical models, but also for other learning models as long as the gradient of the expected difference can be calculated. The general idea is to use two steps, one step for calculating $\mathbb{E}_{P_\theta(Y|Y^L)}S$ and the other step for calculating $\mathbb{E}_{P_\theta(Y)}S$, to estimate the gradient of a parameter θ w.r.t. the objective function.

With the marginal probabilities, the gradient can be obtained by summing over all undetermined labels. It is worth noting that we need to perform the LBP process twice in each iteration, one time for estimating the marginal distribution of unknown variables $y_i =$?, and the other time for marginal distribution over all variables. Finally with the gradient, we update each parameter with a learning rate η. The learning algorithm is summarized in Algorithm 1. After learning the optimal parameters θ, we need to apply the inference algorithm again to compute the marginal probability to infer the unknown labels by finding a label configuration which maximizes the probability $P(Y|X)$. Finally, each node in the factor graph is assigned with label that maximizes the marginal probability.

$$Y^* = \text{argmax}_{Y|Y^L} \ P(Y|X) \tag{14}$$

3.4 Distributed Learning

As a social network may contain millions of users and hundreds of millions of posting documents generated by users, it is beneficial to learn the factor graph from the full social network data using multiple machines rather than a single one. To address this challenge, we deploy the learning task on a distributed system under the map-reduce programming model [6]. Within every individual framework, our target is to smooth the document d_0 based on the subset of influential variables (e.g., group of variables as Y_i), which makes the graph easy to divide: it is natural to extend the individual-centric method to a distributed environment.

Map-Reduce is a programming model for distributed processing of large data sets. In the map stage, each machine (called a process

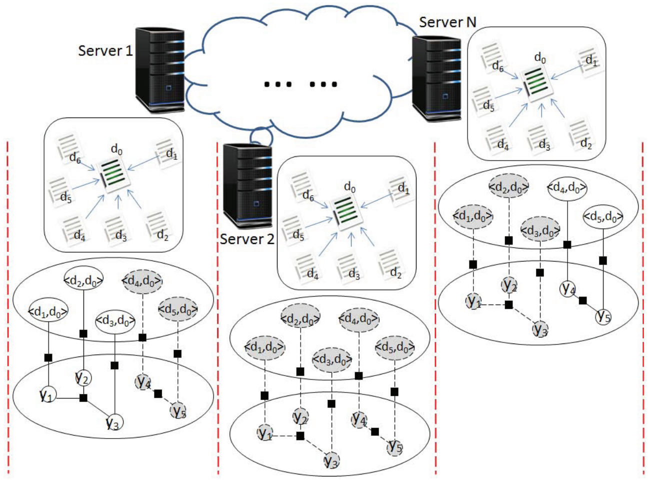


Figure 2: Distributed experimental environment setups. The whole graph is distributed to several computing nodes and the shaded nodes indicate inactive document pairs on that node.

node) receives a subset of data as input and produces a set of intermediate values. In the reduce stage, each process node merges all intermediate values associated with the same target documents and outputs the final computation results. Users specify a map function that generates a set of intermediate values, and a reduce function that merges all intermediate values.

In the process of distributed learning, the whole graph is first divided into several subgraphs that are assigned to slave nodes. Then LBP is performed on each slave to compute the marginal probabilities and the parameter gradient. There is a master node collects and sums up all gradients from subgraphs, and updates parameters by gradient descent method. For details, please refer to [29, 28].

3.5 Language Model Smoothing

Given the estimated influence from all other posting documents, we now propose a term-level language model smoothing approach based on influence propagation. Each word propagates the evidence of its occurrence to other documents based on the estimated influence. To capture the *proximity* heuristics used in language model smoothing [21, 19, 30, 34], we assign "close-by" words with higher propagated counts than those ones which are "far away" from each other. In other words, most propagated counts come from "nearby" terms, while higher influence indicates closer proximity [37, 35, 43]. Theoretically, all positions can have a full vocabulary with different term distributions: each word has a certain non-zero probability to occur in each of the posting document, as if all words had appeared with a variety of discounted counts, while in practical scenarios, we still choose to filter noisy terms with extremely low counts.

In general, we smooth a specific posting document using the background information from the document collection. Consider the traditional way of concatenating the document language model $P(w)$ and the background $P_B(w)$ in a weighted linear combination, i.e., $P'(w)$=$(1-\lambda)P(w)$+$\lambda P_B(w)$. The tradeoff is balanced by the damping factor λ. Compared with this arbitrary and equal weighting for all documents, our socialized language model smoothing based on propagated influence is in a finer-granularity.

The idea for term projection is that if a word w occurs at an influential posting document, we would like to assume that the highly influenced document will also have the words occurred, with a discounted count. The larger the influence estimated, the larger the propagated term counts there will be. Note that each propagated count has a value less than 1.

Let $d_0 = \{w_1, w_2, \dots, w_{|d_0|}\}$ where $|d_0|$ is the length of the document. We use $c(w, d_0)$ to denote the original term count within document d before smoothing. If w does not occur in d_0, $c(w, d_0)$ will be 0, which is a zero count problem. We have calculated the values of Y=$\{y_i\}_{i=1}^n$ from the last section, indicating the influence from the d_i to the document d_0 to smooth. The function actually serves as a discounting factor for terms measured in pairwise. We use $c'(w, d_0)$ to denote the total propagated count of term w from its occurrences in all other documents, i.e.,

$$c'(w, d_0) = (1-\lambda)c(w, d_0) + \lambda \sum_{y_i \in Y} y_i \cdot c(w, d_i) \tag{15}$$

Even if $c(w, d_0)$ is 0, $c'(w, d_0)$ may be greater than 0.

Based on term propagation, we have a term frequency vector $\{c'(w_1, d_0), \dots, c'(w_v, d_0)\}$ for the virtual document d_0^+ extended from document d_0. We can see that term information with calculated influence has been stored in this vector. Thus the language model of this new **smoothed** virtual document can be estimated as

$$P(w|d_0^+) = \frac{c'(w, d_0^+)}{\sum_{w' \in V} c'(w', d_0^+)} \tag{16}$$

where V is the vocabulary set and $\sum_{w' \in V} c'(w', d_0^+)$ is the length of the virtual document after smoothing.

4. EXPERIMENTS AND EVALUATION

4.1 Datasets and Experimental Setups

Utilizing the data crawled from the online social networks through the following linkage, we construct 2 datasets of microblogs and the corresponding users, which form the heterogeneous network. The crawler monitored Twitter data from 3/25/2011 to 5/30/2011, and Weibo data from 9/29/2012 to 11/30/2012. We use roughly one month as the training set and the rest as testing set. The details of the data are listed in Table 1 and #Repost denotes the number of reposting (namely "retweeting" in Twitter or "转发" in Weibo) which we used as labels in our training.

Table 1: Statistics of the social network datasets.

	#User	#Document.	#Link	#Repost	Lang
Twitter	9,449,542	364,287,744	596,777,491	55,526,494	EN
Weibo	3,923,021	216,302,309	258,543,931	101,024,128	CN

Pre-processing. Basically, the social network factor graph can be established from all posting documents and all users, however, the data is noisy. We first pre-filter the pointless babbles [1] by applying the the linguistic quality judgements (e.g., OOV ratio) [24], and then remove inactive users that have less than one follower or followee and remove the users without any linkage to the remaining posting documents. We remove stopwords and URLs, perform stemming and segmentation (for Chinese texts), and build the graph after filtering, and run LBP to obtain the marginal probabilities. We establish the language model smoothed by the estimated influence.

Hashtag Clusters	Numbers	Notes
1. apple	42,528	Tech: apple products
2. nfl	40,340	Sport: American football
3. travel	38,345	General interst
4. mlb	38,261	Sport: baseball
5. fashion	30,053	General interest
1. 中国好声音	72,184	TV show: voice of China
2. 舌尖上的中国	71,169	Food: Chinese foods
3. 微博	63,154	Tech: Microblog service
4. 爱情公寓	57,783	TV drama: culture
5. 小米	49,428	Tech: smart phone

Table 2: Clusters of hashtag topics explored in our study.

4.2 Algorithms for Comparison

To illustrate the performance of our approach, we implement several alternative algorithms as baselines to compare with our method. The baselines include naive smoothing, smoothing by semantics, and positional smoothing from very recent studies. For fairness we conduct the same pre-processing procedures for all algorithms.

The first baseline is based on the traditional language model: **LM** is the language model without smoothing at all. We include the plain smoothing of **Additive** (also known as Add-δ) smoothing and **Absolute Discouting** which decreases the probability of seen words by subtracting a constant [22]. We also implement several classic strategies smoothed from the whole collection as background information: **Jelinek-Mercer** applies a linear interpolation, and **Dirichlet** employs a prior on collection influence [40, 14].

Beyond these simple heuristics, we also examine a series of semantic based language model smoothing. The most representative two semantic smoothing methods are the Cluster-Based Document Model (**CBDM**) proposed in [18], and the Document Expansion Language Model (**DELM**) in [30]. Both methods use semantically similar documents as a smoothing corpus for a particular document: CBDM clusters documents beforehand and smooths a document with the cluster where it belongs to, while DELM finds nearest neighbors dynamically for the document as the smoothing cluster. However, both methods are only based on document-level semantic similarity. We also include Positional Language Model (**PLM**) proposed in [19], which is the state-of-art positional proximity based language smoothing. PLM mainly utilizes positional information without semantic information. We implemented the best reported PLM configuration. We compare our proposed social language model smoothing (**SLMS**) against these baselines.

4.3 Evaluation Metric

It is generally difficult to examine the effect of language model directly. For most of the language model smoothing research, the performance is measured based on extrinsic evaluations (e.g., retrieval) [34, 30, 19]. We include two extrinsic evaluations in this study, i.e., standard posting document retrieval and recommendation, but first we aim to evaluate the information contained in the language itself. Hence we use language *perplexity* to evaluate the smoothed language model.

4.3.1 Intrinsic Evaluation

Our first set of experiments involved intrinsic evaluation of the "perplexity" approach based on a clustering scenario. The experimental procedure is as follows: we manually selected 10 topics (5 for each dataset) based on popularity (measured in the number of postings) and to obtain broad coverage of different types: sports, technology, cultures, and general interests. These topics are shown in Table 1. We group the posting documents with the same hashtag '#' into clusters, and then we remove the hashtags and compute its *perplexity* with respect to the current topic, defined as

$$\text{pow}\left[2, -\frac{1}{N}\sum_{w_i \in V} \log P(w_i)\right]$$

Perplexity is actually an entropy based evaluation. In this sense, the lower perplexity within the same topic cluster, the better performance in purity the topic cluster would have.

4.3.2 Extrinsic Evaluation

In addition to the intrinsic perplexity-based measurements on hashtag clusters, we also evaluate the effectiveness of our smoothed language models on the tasks of microblog search, and information recommendation. Here are a few more details about our experimental setups. For the retrieval task, to avoid the laborious work of building a test collection by hand, we focus our evaluation efforts on documents that contained at least one hashtag. Given the 10 topics mentioned above, we process all documents with hashtags as follows: first, the ground truth labels (i.e., the hashtags) are removed from the documents. We then use the hashtag terms as queries to search for relevant posting documents. The ones originally with the hashtag are regarded as relevant while others not. Note that, the retrieval performance under this experimental setting is to some extent a lower bound, since some of the retrieved documents could be false negative: they do not contain the hashtag but they are indeed relevant.

For the microblog recommendation task, we apply the recommendation framework described in [36] using the same experimental setups. For a specific user, we recommend posting documents based on their previous posting documents and their social behaviors using the graph co-ranking algorithm [36]. The language models for the documents are established after smoothing, and the recommendation list predicts which documents will be reposted in the test data. Again, to save the efforts by human evaluators, we use the automatic evaluation of microblog recommendation. The performance is evaluated by comparing with the ground truth, which indicates whether the posting has been retweeted or not. Also the automatic evaluation sketched above does not assess the full potential of the recommendation system. For instance, it is possible for the algorithm to recommend documents of interest to users but without recommendation, the documents are outside of their scope.

For both the retrieval and the recommendation task, we return the results as a ranking list given a search *query* or a designated *user*, and the ranking list is check by examining the *relevant* documents or the *reposted* documents. We measured ranking performance using the normalized Discounted Cumulative Gain (nDCG) [9].

$$nDCG(k) = \frac{1}{N_\Delta}\sum\nolimits_{|\Delta|}\frac{1}{Z_\Delta}\sum_{i=1}^{k}\frac{2^{r_i}-1}{\log(1+i)}$$

where N_Δ denotes the total numbers of queries or users (Δ=q for queries and Δ=u for users), k indicates the top-k positions in a ranked list, and Z_Δ is a normalization factor obtained from a perfect ranking for a particular query/user. r_i is the judge score (i.e., 1: relevant/reposted, 0: irrelevant/unretweeted) for the i-th posting document in the ranking list for the query/user.

Topic	EN-1	EN-2	EN-3	EN-4	EN-5	CN-1	CN-2	CN-3	CN-4	CN-5
LM	15851	11356	10676	7584	8257	22306	17441	10204	16887	9237
Additive	15195	10035	10342	7198	7924	19139	16221	10108	16342	9003
Absolute	15323	10123	10379	7230	8093	19403	16932	9984	16681	9111
Jelinek-Mercer	14115	10011	10185	9818	8003	20025	16201	10049	16001	8728
Dirichlet	13892	9516	10138	7124	7345	19712	16361	9119	15886	8550
PLM	13730	9925	10426	6913	7512	19965	15230	9865	14219	8981
CBDM	12931	9845	9311	6893	7510	19129	15194	9323	15113	7906
DELM	11853	9820	9513	7133	7348	18809	14165	9510	13985	7621
SLMS	$10788^\star$	$9539^\star$	$8408^\star$	$5817^\star$	$7109^\star$	$18169^\star$	$15375^\star$	$9194^\star$	$13212^\star$	$6919^\star$

Table 3: Perplexity of language models under different hashtag topic clusters. '$\star$' indicates that we accept the improvement hypothesis of SLMS over the best rival baseline by Wilcoxon test at a significance level of 0.01.

We also evaluate the system in terms of Mean Average Precision (MAP) [20] under a similar judge assumption as above:

$$MAP = \frac{1}{N_\triangle} \sum\nolimits_{|\triangle|} \frac{1}{Z_\triangle} \sum_{i=1}^{k} P_i \times r_i$$

Here $N_\triangle$ is the number of documents associated with the query or user, $Z_\triangle$ is the number of relevant documents retrieved or recommended, and P_i is the precision at i-th position for the query/user.

4.4 Overall Performance

We compare the performance of all methods of language model smoothing in the two datasets, measured in the intrinsic evaluation of perplexity, as well as the extrinsic evaluation of retrieval and recommendation. Table 3-5 list the overall results against all baseline methods. Our proposed method SLMS shows clearly better performance than the baseline methods. On average, SLMS achieves an average +23.8% improvement compared with other methods in terms of nDCG and MAP, and an average -18.6% improvement in terms of language perplexity in hashtag topic clustering. The advantage of our proposed method mainly comes from the two dimensions of textual influence and social influence propagated through the documents on social networks. We use a myriad of attribute factors and dependencies to control the influence propagation on the factor graph to make a more reliable estimation.

Language model without any smoothing performs worst as expected, and once again demonstrates the severe weakness of data sparsity on social networks - the *Achilles Heel*! Simple intuition based methods such as additive smoothing does not help a lot, since it only arbitrarily modifies the given term counts straightforward to avoid zero occurrence, which is proved to be insufficient. Absolute smoothing has a comparable performance as additive, due to the similar idea to reduce term counts naively. Jelinek-Mercer and Dirichlet methods are more useful since they include the information from the whole collection as background language models, but they fail to distinguish documents from documents and use all of them equally into smoothing. PLM offers a strengthened language model smoothing strategy within each posting document based on positions, and smooth the terms outside of the posting document formulating the background collection into a Dirichlet prior. The performance of CBDM and DELM indicates a prominent improvement, and proves that semantic attributes included into the smoothing process really make a difference. Both of the smoothing methods cluster documents, and use the clustered documents as a better background. However, none of these methods has made use of the social factors during the language model smoothing, while our method suggests social attributes, such as interactions and relationships, do have an impact on texts through influence propagation.

	nDCG@5	nDCG@25	nDCG@50	MAP
LM	0.271	0.298	0.319	0.328
Additive	0.295	0.320	0.331	0.385
Absolute	0.283	0.328	0.378	0.367
Jelinek-Mercer	0.331	0.376	0.361	0.503
Dirichlet	0.365	0.387	0.408	0.555
PLM	0.392	0.413	0.399	0.532
CBDM	0.388	0.397	0.426	0.546
DELM	0.404	0.438	0.489	0.566
SLMS	$0.463^\star$	$0.492^\star$	$0.503^\star$	$0.600^\star$

Table 4: Retrieval performance against baselines. '$\star$' indicates that we accept the improvement hypothesis of SLMS over the best baseline by Wilcoxon test at a significance level of 0.01.

	nDCG@5	nDCG@25	nDCG@50	MAP
LM	0.506	0.534	0.570	0.603
Additive	0.521	0.572	0.583	0.632
Absolute	0.525	0.581	0.585	0.635
Jelinek-Mercer	0.543	0.583	0.591	0.646
Dirichlet	0.562	0.609	0.621	0.639
PLM	0.558	0.602	0.619	0.653
CBDM	0.538	0.616	0.647	0.651
DELM	0.557	0.645	0.669	0.660
SLMS	$0.617^\star$	$0.662^\star$	$0.689^\star$	$0.673^\star$

Table 5: Recommendation performance. '$\star$' indicates that we accept the improvement hypothesis of SLMS over the best rival baseline by Wilcoxon test at a significance level of 0.01.

Having demonstrated the effectiveness of our proposed methods, we carry the next move to investigate more analysis on parameter settings, factor contributions, convergence and distributed learning analysis, which lead to some interesting observations.

4.5 Analysis and Discussions

4.5.1 Parameter Settings

In the experiments, as we crawled data from two consecutive months, we learn parameters θ=$\{\alpha, \beta\}$ on the data from the first month, using the labels by the reposting behavior, and examine the performance on the testing data from the next month. There is another free parameter λ in Equation (15) to balance the origi-

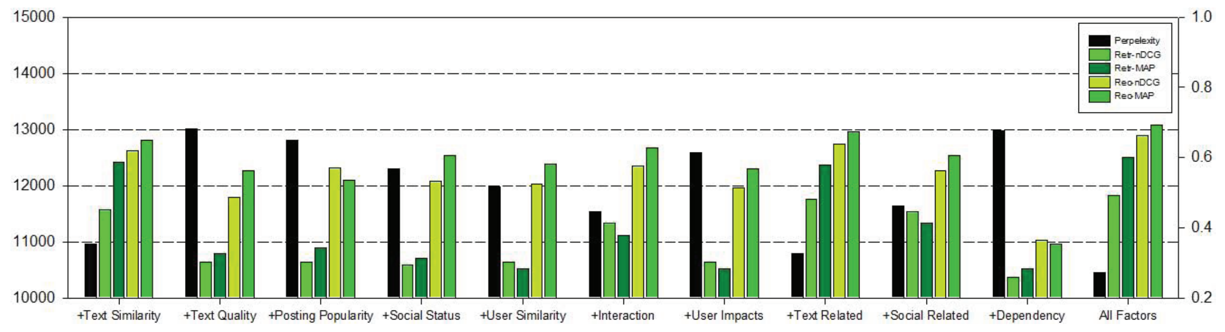


Figure 3: Performance comparison measured in perplexity, nDCG@25, and MAP in hashtag topic clustering, retrieval and recommendation tasks for feature analysis. "+factor(s)" means the performance of individual factor (group) in isolation.

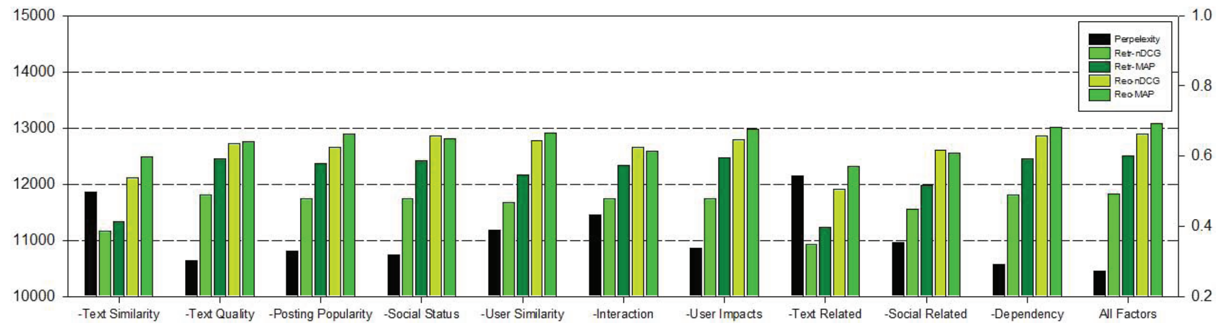


Figure 4: Performance comparison measured in perplexity, nDCG@25, and MAP in hashtag topic clustering, retrieval and recommendation tasks. "-factor(s)" means the performance of individual factor (group) when dropped out from the all-feature model.

nal language model and the smoothing language model. As we opt for more or less generic parameter value as we do not want to tune our method too much to suit the specific datasets at hand, we experiment with value ranging from 0 to 0.9, with a step size of 0.1. By examining the performance of perplexity, we find $\lambda \in$[0.3, 0.4] yields the best results for the two datasets, hence we set λ=0.35.

4.5.2 Factor Contributions

We further analyze the contribution or all factors. We conduct to a detailed experiment on all separate factors and visualize the result in Figure 3-4. In the factor graph for socialized language model smoothing, we consider 8 different attributes and factors: (i) text similarity, (ii) text quality, (iii) posting popularity, (iv) social status, (v) user similarity, (vi) social interactions, (vii) user impacts and (viii) variable dependency. Besides, we combine factors (i)-(iii) as *text related* ones and (iv)-(vii) as *social related* ones. We also list the performance of SLMS which employs all components here for comparison. Here we examine the contribution of the different factors defined in our model. To be specific, we show the performance of all the factors in isolation and then leave-one-out from the full combination of all features, one at a time.

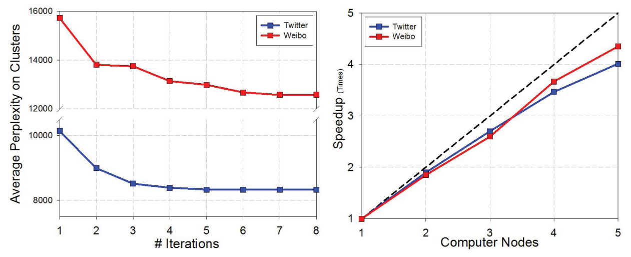
From Figure 3 and 4, we see that all of the individual factors have positive contributions to our evaluation tasks. The first result in Figure 3 is performed using the correspond component only and the second group of results in Figure 4 is performed using the full factor combination exempting the corresponding component, using a leave-one-out manner. For the individual factor analysis, we could see that on average *text similarity* still contributes most in isolation and its absence leads to unfavorable decrease. As to the social related features, *interaction* is the most important social factor for measuring the propagated influence, and gets a clear drop on the performance when left out from full factor combination. It is natural to see through the reposting behavior, the language model for a particular user is influenced by others. We also examine the three aspects of feature groups, i.e., text related factors, social related factors and variable dependencies. Text related factors are proved to be more useful while the social group yields better performance when integrate the factors together. Dependency factor seems to be the least powerful predictor. It is understandable that dependency factor is not deterministic but just to balance the label values. In general, the combination of all factors will be beneficial to improve the performance, as directly compared in Figure 3 and 4, which indicates that our method works well by combining the different factor functions and each factor in our method contributes to the overall improvements.

4.5.3 Convergence Property

We conduct an experiment to see the effect of the number of the loopy belief propagation iterations. Figure 5 (a) illustrates the convergence analysis results of the learning algorithm. We see on both test cases, the LBP-based learning algorithm can converges in less than 10 iterations. After only 6-7 learning iterations, the performance in terms of perplexity on both test cases becomes stable. This suggests that learning algorithm is very efficient and has a good convergence property.

4.5.4 Distributed Scalability

The distributed learning environment can typically achieve a significant reduction of the CPU time on the large-scale social networks. On the moderate scaled network after filtering and pre-



(a). Convergence performance. (b). Distributed computing speedups.

Figure 5: Experimental analysis illustrations.

processing, the speedup of the distributed influence learning is about 3-4 times. However, under the scenario of real world applications without human supervised filtering or pre-processing, the distributed learning framework can generally scale up with a speedup of 10-15 times or more [28].

In particular, we further conduct a scalability experiment with our distributed learning environment. We evaluate the speedup of the distributed learning algorithm on the 5 computer nodes using the two datasets. It can be seen from Figure 5 (b) that when the size of the data set is sufficiently large, the distributed learning shows a good parallel efficiency (speedup>3). In the figure, the dashed line denotes the perfect speedup ideally. The result confirms that the benefits to apply the distributed algorithm on the divisible subgraphs and that the distributed learning like many distributed learning algorithms is good on large-scale data sets.

5. RELATED WORK

Language models have been paid high attention to during recent years [25]. Many different ways of language modeling have been proposed to solve different research tasks. Better estimation of query language models [14, 15] and more accurate estimation of document language models [18, 30] have long been proved to be of great significance in information retrieval and text mining, etc. Language models are typically implemented based on traditional retrieval models, such as text weighting and normalization [40], but with more elegant mathematical and statistical foundations [26].

There is one problem for language models. Given limited data sampling, a language model estimation sometimes encounters with the zero count problem: the maximum likelihood estimator would give unseen terms a zero probability, which is not reliable. Language model smoothing is proposed to address this problem, and has been demonstrated to affect performance significantly [40, 14].

Many approaches have been proposed and tested. There are several ways of to smooth the original language model. The information of background corpus has been incorporated using linear combination [25, 40]. In contrast to the simple strategy which smoothes all documents with the same background, recently corpus contents have been exploited for more accurate smoothing. The basic idea is to smooth a document language model with the documents similar to the document under consideration through clustering [34, 18, 30]. Position information has also been used to enrich language model smoothing [41, 19] and the combination of both strategies of position and semantics [34]. In their work, the key idea is to define a language model for each position within a document, and score it based on the language models on all positions: hence the effect of positional adjacency is revealed. Beyond the semantic and/or position related smoothing intuitions, structural based language model smoothing is an alternative direction to investigate. A graph based language model smoothing method has been proposed utilizing structural adjacency only between neighboring nodes [21, 8].

There is a study in [17] which smoothes document language models of tweets for topic tracking in online text streams. Basically, it applies general smoothing strategies (e.g., Jelinek-Mercer, Dirichlet, Absolute Discounting, etc.) on the specific tracking task, while we have proposed a unified factor graph model based framework which incorporates a series of social factors as well as the text information on language model smoothing, which is a novel insight. To the best of our knowledge, we are the pilot study which maps the social influence onto the textual dimension and hence to estimate the influence between posting documents. Language model is smoothed by the influence propagated on the factor graph.

6. CONCLUSIONS

Online social networks are massive, useful and of great potentials, but have a severe bottleneck of textual data sparsity. To tackle such an Achilles Heel of social networks, we present an influence propagation based language model smoothing method to solve the zero count phenomenon for online social networks. The social influence is estimated based on a factor graph model, by utilizing a series of attributes and dependency factors from both textual and social dimensions. In this way, we propagate the term occurrence along the networks with a discounted counts according to the estimated pairwise influence between documents, and finally smooth the sparse language model accordingly.

We examine the effect of our proposed language model smoothing method on a series of intrinsic and extrinsic evaluation metrics based on the Twitter dataset (in English) and Weibo dataset (in Chinese). Our proposed method consistently and significantly outperforms the alternative baselines (with -18.6% improvement in terms of perplexity and +23.8% improvement in nDCG and MAP). Furthermore, we have investigated factor contributions as well as a series of experimental analysis for convergence and distributed learning. In general, all features facilitate the smoothing performance, while text similarity and social interaction are proved to have a stronger contribution. In the future, we will include more flexible social factors and make our model adaptive to diversified online social networks, e.g., structures with reciprocal linkage (e.g., Facebook) or documents without reposting behaviors (e.g., Foursquare).

7. ACKNOWLEDGMENTS

This work was partially done when the first author was in University of Pennsylvania. We thank all the anonymous reviewers for their valuable and constructive comments.

8. REFERENCES

[1] P. Analytics. Twitter study–august 2009. 15, 2009.

[2] L. Bottou. Online learning and stochastic approximations. *On-line learning in neural networks*, 17:9, 1998.

[3] M. A. Carreira-Perpinan and G. E. Hinton. On contrastive divergence learning. In *Artificial Intelligence and Statistics*, volume 2005, page 17, 2005.

[4] G. Casella and E. I. George. Explaining the gibbs sampler. *The American Statistician*, 46:167–174, 1992.

[5] K. Chen, T. Chen, G. Zheng, O. Jin, E. Yao, and Y. Yu. Collaborative personalized tweet recommendation. In *SIGIR '12*, pages 661–670, 2012.

[6] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. pages 107–113, 2004.

[7] J. Hopcroft, T. Lou, and J. Tang. Who will follow you back?: Reciprocal relationship prediction. In *CIKM '11*, pages 1137–1146, 2011.

[8] Y.-Y. Huang, R. Yan, T.-T. Kuo, and S.-D. Lin. Enriching cold start personalized language model using social network information. In *ACL '14*, pages 611–617, 2014.

[9] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, 2002.

[10] R. Kindermann, J. L. Snell, et al. *Markov random fields and their applications*, volume 1. American Mathematical Society Providence, RI, 1980.

[11] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *Information Theory, IEEE Transactions on*, 47(2):498–519, 2001.

[12] T.-T. Kuo, R. Yan, Y.-Y. Huang, P.-H. Kung, and S.-D. Lin. Unsupervised link prediction using aggregative statistics on heterogeneous social networks. In *KDD '13*, pages 775–783, 2013.

[13] H. Kwak, C. Lee, H. Park, and S. Moon. What is twitter, a social network or a news media? In *WWW '10*, pages 591–600, 2010.

[14] J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *SIGIR '01*, pages 111–119, 2001.

[15] V. Lavrenko and W. B. Croft. Relevance based language models. In *SIGIR '01*, pages 120–127, 2001.

[16] C. Lin, C. Lin, J. Li, D. Wang, Y. Chen, and T. Li. Generating event storylines from microblogs. In *CIKM '12*, pages 175–184, 2012.

[17] J. Lin, R. Snow, and W. Morgan. Smoothing techniques for adaptive online language models: Topic tracking in tweet streams. In *KDD '11*, pages 422–429, 2011.

[18] X. Liu and W. B. Croft. Cluster-based retrieval using language models. In *SIGIR '04*, pages 186–193, 2004.

[19] Y. Lv and C. Zhai. Positional language models for information retrieval. In *SIGIR '09*, pages 299–306, 2009.

[20] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*, volume 1. 2008.

[21] Q. Mei, D. Zhang, and C. Zhai. A general optimization framework for smoothing language models on graph structures. In *SIGIR '08*, pages 611–618, 2008.

[22] H. Ney, U. Essen, and R. Kneser. On the estimation ofsmall'probabilities by leaving-one-out. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 17(12):1202–1212, 1995.

[23] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: bringing order to the web. 1999.

[24] E. Pitler, A. Louis, and A. Nenkova. Automatic evaluation of linguistic quality in multi-document summarization. In *ACL '10*, pages 544–554, 2010.

[25] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *SIGIR '98*, pages 275–281, 1998.

[26] F. Song and W. B. Croft. A general language model for information retrieval. In *CIKM '99*, pages 316–321, 1999.

[27] J. Tang, T. Lou, and J. Kleinberg. Inferring social ties across heterogenous networks. In *WSDM '12*, pages 743–752, 2012.

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

[29] W. Tang, H. Zhuang, and J. Tang. Learning to infer social ties in large networks. In *ECML/PKDD '11*, pages 381–397. 2011.

[30] T. Tao, X. Wang, Q. Mei, and C. Zhai. Language model information retrieval with document expansion. In *HLT-NAACL '06*, pages 407–414.

[31] J. Teevan, D. Ramage, and M. R. Morris. #twittersearch: A comparison of microblog search and web search. In *WSDM '11*, pages 35–44, 2011.

[32] Z. Wang, J. Li, Z. Wang, and J. Tang. Cross-lingual knowledge linking across wiki knowledge bases. In *WWW '12*, pages 459–468, 2012.

[33] S. Wu, J. Sun, and J. Tang. Patent partner recommendation in enterprise social networks. In *WSDM '13*, pages 43–52, 2013.

[34] R. Yan, H. Jiang, M. Lapata, S.-D. Lin, X. Lv, and X. Li. Semantic v.s. positions: Utilizing balanced proximity in language model smoothing for information retrieval. In *IJCNLP'13*, pages 507–515, 2013.

[35] R. Yan, L. Kong, C. Huang, X. Wan, X. Li, and Y. Zhang. Timeline generation through evolutionary trans-temporal summarization. In *EMNLP '11*, pages 433–443, 2011.

[36] R. Yan, M. Lapata, and X. Li. Tweet recommendation with graph co-ranking. In *ACL '12*, pages 516–525, 2012.

[37] R. Yan, X. Wan, J. Otterbacher, L. Kong, X. Li, and Y. Zhang. Evolutionary timeline summarization: A balanced optimization framework via iterative substitution. In *SIGIR'11*, pages 745–754, 2011.

[38] Z. Yang, K. Cai, J. Tang, L. Zhang, Z. Su, and J. Li. Social context summarization. In *SIGIR '11*, pages 255–264, 2011.

[39] J. S. Yedidia, W. T. Freeman, Y. Weiss, et al. Generalized belief propagation. In *NIPS*, volume 13, pages 689–695, 2000.

[40] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *SIGIR '01*, pages 334–342, 2001.

[41] J. Zhao and Y. Yun. A proximity language model for information retrieval. In *SIGIR '09*, pages 291–298, 2009.

[42] W. X. Zhao, J. Jiang, J. Weng, J. He, E.-P. Lim, H. Yan, and X. Li. Comparing twitter and traditional media using topic models. In *ECIR '11*, pages 338–349. 2011.

[43] X. W. Zhao, Y. Guo, R. Yan, Y. He, and X. Li. Timeline generation with social attention. In *SIGIR '13*, pages 1061–1064, 2013.