Aspect-Level Influence Discovery from Graphs

Chuan Hu and Huiping Cao

Abstract—Graphs have been widely used to represent objects and object connections in applications such as the web, social networks, and citation networks. Mining influence relationships from graphs has gained increasing interests in recent years because providing information on how graph objects influence each other can facilitate graph exploration, graph search, and connection recommendations. In this paper, we study the problem of detecting influence aspects, on which objects are connected, and influence degree (or influence strength), with which one graph node influences another graph node on a given aspect. Existing techniques focus on inferring either the overall influence degrees or the influence types from graphs. In this paper, we propose a systematic approach to extract influence aspects and learn aspect-level influence strength. In particular, we first present a novel instance-merging based method to extract influence aspects from the context of object connections. We then introduce two generative models, Observed Aspect Influence Model (OAIM) and Latent Aspect Influence Model (LAIM), to model the topological structure of graphs, the text content associated with graph objects, and the context in which the objects are connected. To learn OAIM and LAIM, we design both non-parallel and parallel Gibbs sampling algorithms. We conduct extensive experiments on synthetic and real data sets to show the effectiveness and efficiency of our methods. The experimental results show that our models can discover more effective results than existing approaches. Our learning algorithms also scale well on large data sets.

Index Terms—Graph mining, machine learning, Gibbs sampling, knowledge extraction

1 Introduction

Graphs have been used to represent objects and interactions between objects in many applications. The connections among objects (i.e., edges between graph nodes) in graphs can represent the information that one object influences other objects. However, these connections are not equally important. For instance, strong and weak ties often exist in social networks [30]. Most existing graphs do not explicitly represent how strong the influence relationships happen in specific aspects.

Let us use *influence aspect* to denote the area (or context) in which one object influences another object, and use the *influence degree* to denote how strong the influence is. Mining and providing influence relationships with influence aspects and influence degrees to describe object connections can greatly improve the use of graphs in the process of graph exploration (e.g., [20], [32], [36]), graph search (e.g., [21]), and connection recommendations which are based on object relationships in graphs (e.g., [28], [33]). We give three examples to show the need of mining influence aspects and aspect-level influence degrees from graphs.

Motivating Example 1. The citation relationships among articles can provide researchers helpful information to find articles related to what (s)he is interested in. However, given a research article, there may be hundreds or thousands of research papers citing it. Although all these articles are influenced by the given article, only very few of them are strongly influenced by it regarding one aspect (e.g., methodology). Being able to discover on which aspect and how strong one article influences the others can greatly help a researcher explore or search publication collections.

Motivating Example 2. People in a social network are connected with and have influence to each other. However, even for directly connected people, their influence to each other is different. Discovering the strength of connections and the connection types can help identify strong ties (i.e., people with high influence relationship). Stronger influence relationships can provide more information for product recommendations. For instance, if a user named Alice is influenced by her friend Sarah more on *entertainment* aspect and influenced more by her friend Bob on *study* aspect, then Sarah's movie preference is more useful (than Bob's) when we make movie recommendations to Alice.

Motivating Example 3. In biological pathway databases (e.g., Kyoto Encyclopedia of Genes and Genomes (KEGG) [17], Reactome [16]), molecules (e.g., genes, proteins) are treated as graph objects whose content describes their functionalities, and interactions between molecules are modeled as graph edges. The molecule interactions (edges) are annotated with different contexts, which are experimental settings or conditions under which interactions happen. Discovering the influence relationships among molecules in different contexts (i.e., on different aspects) can greatly improve the understanding of biological pathways.

Much effort has been put to discover influence on different types of graphs. But most of these techniques [7], [20], [32], [36] focus on detecting influence degrees.

Our work is different in that we detect both *influence aspects* and *influence degrees at the aspect level* from graphs by utilizing the topological structure of graphs, the text content associated with graph nodes, and the context in which

- *The authors are with the Department of Computer Science, New Mexico State University, Las Cruces, NM 88003. E-mail: {chu, hcao}@cs.nmsu.edu.*

Manuscript received 20 May 2015; revised 28 Feb. 2016; accepted 29 Feb. 2016. Date of publication 3 Mar. 2016; date of current version 1 June 2016. Recommended for acceptance by A. Singh. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TKDE.2016.2538223

graph nodes are connected. Influence aspects denote how two nodes are connected and influence degrees at the aspect level describe how strongly two graph nodes are connected on the given aspect. In many graphs, aspect labels are not explicitly available. For example, in citation networks, section labels in articles can help us identify the aspects (context) of the content. However, in social networks, not every *follow* relation has explicit connection labels (aspects).

To discover the aspect-level influence relationships from graphs, we need to address several major challenges.

- Influence aspects and the aspect-level influence degrees need to be properly defined, represented, and modeled.
- Influence aspects need to be automatically extracted from the contexts of graph connections, which are short sentences.
- Strategies are needed to evaluate the effectiveness of the discovered aspects and the aspect-level influence degrees.
- Both the text and structure information of graphs need to be utilized in the process of influence discovery because both graph nodes and edges carry informative knowledge.
- The proposed solution should scale on large data sets in the real world.

To address the above challenges, we first propose a new semi-supervised method to automatically extract context labels, which act as influence aspects, from graph connections. Then, we propose two novel aspect influence models to capture the influence aspects and the influence degrees at the aspect level by utilizing both the text and structure information in a graph. In particular, we model the text information of graph nodes using latent topic states, which are introduced in topic models [4]. We model the graph edges (structure) by associating them with influence aspects (which can be observed, or extracted, or latent) and aspect-level degrees.

The contributions of this work are as follows.

- We formally define the problem of discovering aspect-level influence relationships, which consist of influence aspects and influence degrees on specific aspects from graphs.
- We propose a semi-supervised aspect extraction algorithm. This algorithm utilizes a very few number of context sentences with labeled aspects to learn a classifier and uses the classifier to predict the aspects for the unlabeled contexts.
- We propose two generative probabilistic models, Latent Aspect Influence Model (LAIM) and Observed Aspect Influence Model (OAIM). These two models capture the generative process of the contents of graph nodes by considering both the text and structure information in graphs.
- We design a blocking Gibbs sampling algorithm to learn both LAIM and OAIM.
- We design a parallel Gibbs sampling algorithm by utilizing a property of our problem and a lazy count updating strategy to further improve the efficiency of the algorithm.
- We conduct extensive experiments on synthetic and real data sets to show the effectiveness and efficiency of the proposed approaches. Quantitative comparisons of LAIM and OAIM with baseline approaches show that introducing influence aspects can discover more comprehensive influence relationships from graphs. The efficiency studies show that our Gibbs sampling algorithms scale well in the graph size and the complexity of the models.

This paper is organized as follows. Section 2 formally defines the problem and related notations. Section 3 reviews the related work in this area. Section 4 presents the solution framework. Section 5 presents the algorithm to extract influence aspects. Sections 6 and 7 explain in detail the two aspect influence models and the Gibbs sampling algorithms to learn the models. Section 8 shows the experimental results of the proposed approaches. Finally, Section 9 concludes this paper and shows future directions.

2 Problem Formulation

Our study takes a graph $G = (V, E)$ as input, where V is the set of graph objects (or graph nodes) o and E is a set of directed edges $o' \rightarrow o$. Each graph object o is associated with descriptive information. For instance, in a citation network of research articles, one research article is a graph object o and it contains a list of words; In social networks, a user is a graph object, which is associated with descriptive data capturing user activities. We use *object profile* to denote the descriptive information associated with each graph object and formally define it as follows.

Definition 2.1 (Object profile). *The profile of an object o is a sequence of tokens: $t_{o,1}, t_{o,2}, \ldots, t_{o,T(o)}$, where $T(o)$ is the number of tokens in the profile of o and $t_{o,pos}$ is the token of this object at position pos.*

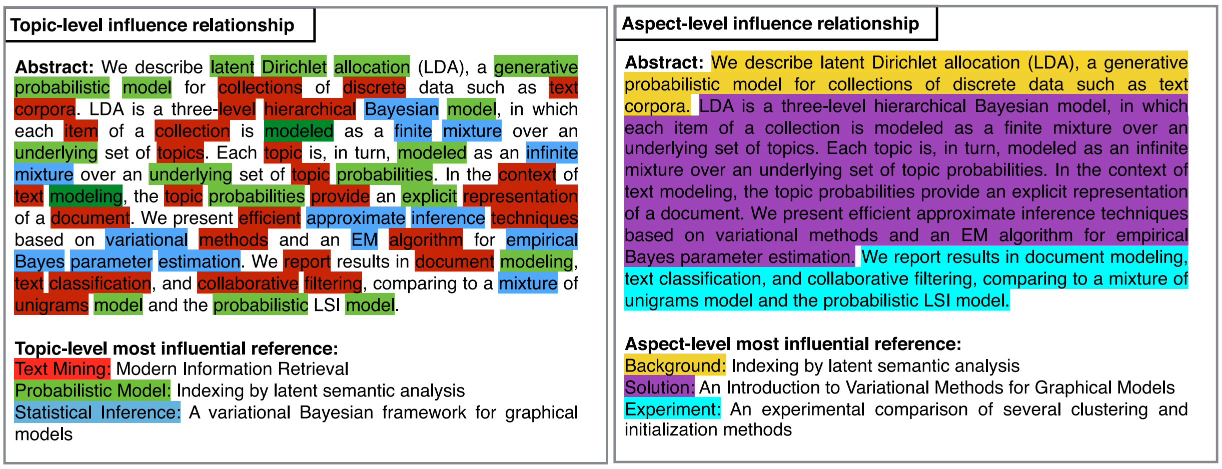
We use D and T to represent the total number of objects in graph G and the total number of distinct tokens in all the object profiles respectively. We also use $\mathcal{D}$ to denote the set of distinct tokens in the profiles of all the graph objects. A directed edge $o' \rightarrow o \in E$ implicitly denotes that object o' exists before and influences o.

2.1 Influence Aspects and Degrees

One object may influence another object with different degrees in different areas (or contexts) as described in the motivating examples in Section 1. These areas or contexts are denoted as influence aspects, which are formally defined as follows.

Definition 2.2 (Influence aspect). Influence aspects *are text labels representing specific semantic areas (or contexts) on which graph objects are connected.*

Structurally, influence aspects are different from topics [4] in their roles because influence aspects are associated with graph edges and topics are extracted from graph nodes.



Topic-level influence relationship

Abstract: We describe latent Dirichlet allocation (LDA), a generative probabilistic model for collections of discrete data such as text corpora. LDA is a three-level hierarchical Bayesian model, in which each item of a collection is modeled as a finite mixture over an underlying set of topics. Each topic is, in turn, modeled as an infinite mixture over an underlying set of topic probabilities. In the context of text modeling, the topic probabilities provide an explicit representation of a document. We present efficient approximate inference techniques based on variational methods and an EM algorithm for empirical Bayes parameter estimation. We report results in document modeling, text classification, and collaborative filtering, comparing to a mixture of unigrams model and the probabilistic LSI model.

Topic-level most influential reference:
Text Mining: Modern Information Retrieval
Probabilistic Model: Indexing by latent semantic analysis
Statistical Inference: A variational Bayesian framework for graphical models

Aspect-level influence relationship

Abstract: We describe latent Dirichlet allocation (LDA), a generative probabilistic model for collections of discrete data such as text corpora. LDA is a three-level hierarchical Bayesian model, in which each item of a collection is modeled as a finite mixture over an underlying set of topics. Each topic is, in turn, modeled as an infinite mixture over an underlying set of topic probabilities. In the context of text modeling, the topic probabilities provide an explicit representation of a document. We present efficient approximate inference techniques based on variational methods and an EM algorithm for empirical Bayes parameter estimation. We report results in document modeling, text classification, and collaborative filtering, comparing to a mixture of unigrams model and the probabilistic LSI model.

Aspect-level most influential reference:
Background: Indexing by latent semantic analysis
Solution: An Introduction to Variational Methods for Graphical Models
Experiment: An experimental comparison of several clustering and initialization methods

Fig. 1. Comparison of aspect-level influence and topic-level influence [20] (use the abstract of LDA [4] as an example).

Semantically, influence aspects are more general than topics. Influence aspects are user defined and application specific. In some situations, aspects capture the context in which influence happens. For example, if researchers exploring articles are interested in whether one article influences the other on *methodology*, then researchers can denote *methodology* as one influence aspect. In this case, topics cannot represent aspects. In other situations, aspects may capture the theme (or topic) on which influence happens. We still use the article exploration application as an example. When a researcher is interested in whether one article influences another article on *the sampling methodology*, which can be learned as a topic, this researcher can denote *sampling methodology* as one aspect. There are also situations that users denote the specific reason for influence, while this specific reason can be interpreted as either context or topic. Social network applications can lend us many examples on this. A user may denote that he follows (and is influenced by) another user on aspect *president election*, which can be interpreted as either a context or a topic.

We illustrate the specific differences in the content semantic between the aspect (-level influence) and the topic (-level influence) [20] in Fig. 1 by using the abstract of LDA [4] as an example. Directly applying topic-level influence discovery method (e.g., [20]) cannot find aspect-level influence because aspects are explicit labels associated with graph edges and existing topic-level influence discovery models do not capture such edge labels.

In some graph mining problems, topics discovered from graph nodes cannot capture the connection contexts. In *Motivating example 3*, directly applying topic models on molecules (i.e., graph objects) in Reactome [16] cannot find meaningful aspects because the topics are about molecule functions, while the context information (e.g., annotated labels provided by researchers) is more about interaction conditions (experimental settings, temperature requirements) on graph edges.

Influence aspects can be observed or be latent. For the aspects that are explicit on graphs, we denote them as observed aspects. When the number of observed aspects is large, it is hard to visualize the influence at the aspect-level. Then, latent aspects are introduced to summarize the large number of observed aspects. Let t^a be an observed influence aspect and $\mathcal{A}$ be the set of t^as.

A latent influence aspect a is a probabilistic distribution among all the distinct observed aspects in $\mathcal{A}$. The definition of the latent influence aspect is similar to the topic definition in topic modeling [4], where each latent topic is a distribution over the observed profile tokens. However, they are different. The observed aspects for a latent aspect are from the context set $\mathcal{A}$ while the tokens for a latent topic are from the distinct token set $\mathcal{D}$.

Definition 2.3 (Aspect-level influence degree). *When an object o' influences another object o on a latent aspect a or an extracted/explicit aspect t^a, we use $I(o' \xrightarrow{a} o)$ or $I(o' \xrightarrow{t^a} o)$, which is a number in the range of $[0, 1]$, to denote the influence degree that o' has over o on the aspect a or t^a.*

Influence degree captures the strength of the influence. Note that the influence is directional, and influence degrees are not symmetric, i.e., $I(o' \xrightarrow{a} o) \neq I(o \xrightarrow{a} o')$ and $I(o' \xrightarrow{t^a} o) \neq I(o \xrightarrow{t^a} o')$.

2.2 Problem Definition

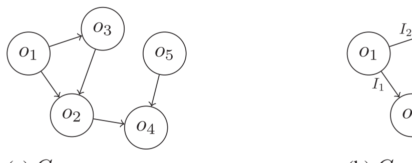
Our research problem is formally defined as follow.

Definition 2.4 (Learning aspect-level influence). *Given a graph $G = (V, E)$, the problem of learning aspect-level influence is (i) to extract influence aspects t_a and (ii) to learn $I(o' \xrightarrow{t^a} o)$ or $I(o' \xrightarrow{a} o)$ for each directed edge $o' \rightarrow o \in E$.*

For example, given a graph G_s in Fig. 2a, the discovery algorithm is to output another graph G_t in Fig. 2b which is associated with vectors of influence aspects and influence degrees on the corresponding aspects in Fig. 2c.

3 Related Work

Topic models such as PLSI [13] and Latent Dirichlet Allocation (LDA) model [4] are designed to model single document sets. In topic models, every object is associated with a multinomial distribution over latent topics, which generates a latent topic for each token. There is a corpus level topic to token mixture, each of which is a distribution over the observed tokens.



(a) G_s (b) G_t

I_1	I_2	I_3	I_4	I_5
b:0.6	p:0.5	p:0.6	e:0.7	m:0.7
p:0.3	b:0.26	e:0.3	p:0.2	b:0.2
m:0.95	e:0.14	b:0.1	b:0.1	p:0.1
e:0.05	m:0.1	m:0.0	m:0.0	e:0.0

(c) Influence vectors for G_t

Fig. 2. Input graphs G_s (without influence relationships) and output graph G_t (with influence relationships): the influence aspects that can be observed/extracted on the edges are: b (ackground), e (xperiment), m (ethodology), and p (roblem).

Much effort has been put to discover connections among graph nodes. The first group of works that are highly related to our research targets to discover influence degrees among objects. Dietz et al. [7] proposed one of the first few articles that leverage both the text and links in a citation graph into a probabilistic model to infer influence strength between articles by utilizing the topic model. In [7] a graphical model is proposed to capture the overall influence among graph objects. A further step from [7], our work can discover the influence among graph objects on different aspects. Liu et al. [20] proposed a similar generative graphical model to infer both direct and indirect topic-level influence strengths between objects from heterogeneous graphs. Our work is different in that we discover aspect-level influence, which is more general than topic-level influence [20] as we have discussed in Section 2.1.

The second group of works is to identify the relationship types among objects in graphs. Diehl et al. [6] introduced the problem of identifying the types of relationships between communication parties from their communication content (e.g., messages), where relationship types are predefined. The focus of the analysis is on the communication content associated with graph edges. Our work is different in that we leverage the structure of graphs. Wang et al. [36] discussed techniques to detect one specific type of influence relationships, adviser-advisee relationship, from publication networks. Tang et al. [30] presented a framework to learn the social relationship types across heterogeneous graphs. Barbieri et al. [3] presented a model to predict links and explain the links on social networks using predefined relationship types. The relationship types in [3], [30] are similar to influence aspects in our work. However, our work differs from [3], [30] in that we do not assume the existence of known relationships from any graphs.

There are other works that discover different types of influence relationships in graphs. Kataria et al. [18] studied the problem of predicting the existence of citation relationships among documents. Shen and Jin [28] and Tang et al. [33] introduced generative process approaches to do social recommendations for the possible creation of new graph edges. Miao et al. [22] presented Latent Association Analysis (LAA) model to discover the potential links in bipartite graphs. Iwata et al. [15] utilized cascade Poisson processes to learn the influence strength among users by analyzing special event sequences which consist of information about when users adopt products. Kutzkov et al. [19] designed online algorithms to calculate influence strength among users by using a continuous stream of tweets.

Several commonly recognized influence factors in social sciences include homophily, contagion, and causality [27], [29] where homophily captures the similarities among social users, contagion represents the temporal order of user behavior, and causality finds the reason behind user behavior. In this work, we consider the homophily and contagion factors by utilizing object-profile similarities (homophily) and the directed edges representing the sequential order of object behavior (contagion).

Last but not the least, the influence relationship we study in this work is different from influence measurements such as impact factor [8] or H-index [12]. These existing measurements measure the overall influence of a researcher (H-index) or a journal (impact factor) in a global manner. They do not measure the localized influence between researchers or between journals (e.g., the influence of one researcher/journal over another researcher/journal). These measurements are calculated at a coarse granularity. Our work discovers the localized influence at user-defined finer granularities.

4 Proposed Framework

The first problem is to extract aspects from the profile of graph objects if aspects are not available on graphs.

Given a graph $G = (V, E)$ and profiles of each object $o \in V$, the problem of aspect extraction is to find a labeling function $f : \mathcal{D}_{T(o)} \rightarrow (\mathcal{D} \times \mathcal{A})_{T(o)}$, which takes the tokens $(t_1, t_2, \ldots)$ of the context for an directed edge $o' \rightarrow o$ as input, and outputs $((t_1, t_1^a), (t_2, t_2^a), \ldots)$. Context tokens can be from the profiles of o where o mentions or references o', or are input from users. After extracting aspect labels from graphs, we design probabilistic models to discover aspect-level influence degrees from graphs where the aspect can be explicit/extracted or be latent.

5 Semi-Supervised Aspect Extraction

Aspects may explicitly exist in the graph (edges). E.g., on social networks, the observed aspects are the tags people put for connections. However, such aspects may not be explicitly available and they need to be learned on some graphs.

In this section, we present a semi-supervised method to annotate the profiles of graph objects with aspects. Existing text annotation techniques generally provide Post-of-Speech tags [35], sentiment polarity [24], or topic labels [4]. However, the labels generated by those techniques are different from the aspects that this work targets to discover, which represent the context that the graph objects are connected on.

Intuitively, the profile tokens within a certain context share the same aspect. This intuition comes from the observation that one sentence generally represent one major point. Based on this intuition, we treat one sentence (instead of one token) as an instance when we label the objects' profiles with aspects.

```
Merging-based Aspect Extraction Algorithm
Input: a classification algorithm, window size $W$, training instances $\mathbf{X}$ with their aspect labels $\mathbf{y}$
1) Initialize augmented instance matrix $\mathbf{X}'$ and new instance label vector $\mathbf{y}' = \mathbf{0}$
2) Group instances in $\mathbf{X}$ by their aspect labels and get $T_a$ groups: $\mathbf{X}_1, \mathbf{X}_2, \dots \mathbf{X}_{T_a}$
3) For each observed aspect $t_i^a (i \in \{1, 2, \dots T_a\})$
    a) while $|\mathbf{X}_i| \geq W$
        i) Randomly select $W$ distinct instances $\mathbf{x}_{i_1}, \mathbf{x}_{i_2}, \dots, \mathbf{x}_{i_W}$ from $\mathbf{X_i}$
        ii) Create a new instance $\mathbf{x}' = \sum_{j=1}^{W} \mathbf{x}_{i_j}$ and assign $t_i^a$ as its aspect
        iii) Append $\mathbf{x}'$ and $t_i^a$ to $\mathbf{X}'$ and $\mathbf{y}'$ respectively
        iv) Remove $\mathbf{x}_{i_1}, \mathbf{x}_{i_2}, \dots, \mathbf{x}_{i_W}$ from $\mathbf{X}_i$
4) Train a classifier using the given classification algorithm and training data $\begin{pmatrix} \mathbf{X} \\ \mathbf{X}' \end{pmatrix}$ and $\begin{pmatrix} y \\ y' \end{pmatrix}$
5) Return the trained classifier
```

Fig. 3. Merging-based aspect extraction algorithm.

Recall that $\mathcal{D}$ contains T distinct tokens in all the object profiles. For each short sentence, let the ith token occur x_i times. A zero x_i means that the ith token does not occur in this sentence. Then, each short sentence can be represented as a vector $\mathbf{x} = (x_1, x_2, \dots, x_T)$. The vectors of all the N short sentences form a sentence-token matrix $\mathbf{X} = \begin{pmatrix} \mathbf{x}_1 \\ \cdots \\ \mathbf{x}_N \end{pmatrix}$. The matrix $\mathbf{X}$ is extremely sparse because the number of tokens in a sentence is much smaller than the number of distinct tokens T in all the object profiles. Each short sentence is manually labeled with an aspect y. All the manually labeled aspects form the set of observed aspects $\mathcal{A}$.

The aspect extraction problem is modeled as a classification problem, where the training instances are N labeled sentences represented with $\mathbf{X}$ and their corresponding aspect labels are $\mathbf{y} = (y_1, y_2, \dots, y_N)$.

Many classification algorithms were proposed to classify long articles [1]. However, such algorithms may not work well to classify $\mathbf{X}$ because the text features of short sentences are not significant for linear and kernel methods [37]. Existing classification methods classify short sentences by using extra external knowledge, such as knowledge graph [37], latent topic features [26] to fill the sentence-term matrix and make the training data denser. Utilizing extra external knowledge in such classification algorithms requires more workload from users.

We propose a merging-based approach for short sentence classification (or aspect extraction from short sentences). This method does not require and utilize external domain knowledge. Instead, it creates more training instances from existing training instances in $\mathbf{X}$. Let the newly created training instances be augmented instances. An augmented instance $\mathbf{x}'$ is created by merging $W\ (\geq 2)$ instances $\mathbf{x}_{i_1}, \mathbf{x}_{i_2}, \dots, \mathbf{x}_{i_W}$ that have the same aspects t^a. In particular, $\mathbf{x}' = \sum_{j=1}^{W} \mathbf{x}_{i_j}$. It means that the frequency of a token in the augmented instance is the summation of the frequencies of this token in the W instances. The new augmented instance $\mathbf{x}'$ is assigned with aspect t^a.

The *merging-based aspect extraction algorithm* is shown in Fig. 3. It takes as input (i) a basic classification algorithm, (ii) a window size W, and (iii) the training instances $\mathbf{X}$ together with their labels $\mathbf{y}$. It outputs a trained classifier. This algorithm puts all the augmented instances to $\mathbf{X}'$ (Step 1). For each observed aspect t_i^a (Step 3), that algorithm creates augmented instances using W original training instances and assigns t_i^a to be its aspect label. The merging process generates an augmented data set $\mathbf{X}'$ and aspect labels $\mathbf{y}'$. The original training instances and the augmented data set are used to train a classifier using a specific classification algorithm (e.g., Decision Tree). This trained classifier is then used to predict the aspect labels of new coming instances.

6 Aspect Influence Models

We design models to learn the influence degrees on observed, extracted, or latent aspects. Objects' profiles and their influence relationships are governed by three major factors. The first factor is the set of latent topic states that are associated with each object. An object's latent topic states determine the internal theme of this object. Two objects with similar internal themes tend to influence each other more compared with objects that do not share internal themes. E.g., an article with theme "bioinformatics" is more likely influenced by articles with "biology" theme than by articles with "politics" theme. This factor has been taken into consideration in research of topic modeling [4]. The second factor is the set of links that connect to the objects (i.e., directed edges) [7], [20]. The explicitly linked objects have natural influence relationships, so the model should take the explicit links into consideration. The third factor is the context in which the objects link to each other [18]. Such context reflects the influence aspect. For example, the link connecting one article to another article appears in the context of "experimental results" is more likely to reflect influence aspect "experiment" than the aspect "problem definition". Our solution framework is to learn these factors and derive the influence relationships from these factors. In our approaches, we adopt the definition of latent topic states which are probabilistic distributions over tokens (or words). We define two types of influence aspects in our models as discussed in Section 2.1. The model parameters are listed in Table 1.

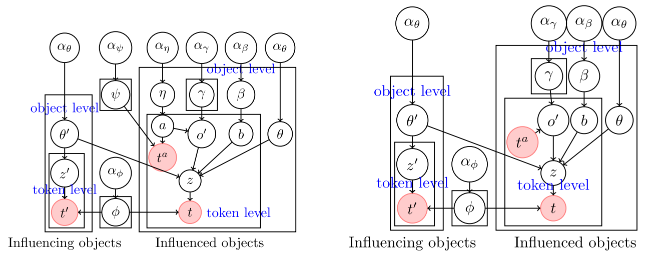
6.1 Latent Aspect Influence Model

The creation of an object profile can be envisioned as being generated from several fundamental ideas. From the perspective of generating an object profile, when an object is not influenced by any other objects, its tokens are generated by its own latent topics. The Latent Aspect Influence Model is introduced mainly to govern the generation of profile tokens for objects that are influenced by other objects. When an object o is influenced by another object o', part of o's idea (i.e., topics) is new, but the other part of its idea is borrowed from o'. When o borrows idea from o', the tokens are affected by both the major idea of o' and the influence aspects from o' to o. Inspired by the generation of tokens in the topic modeling [4], where each profile token is assumed to be associated with a latent topic and is drawn from a topic-token distribution, LAIM assumes that every profile token of o is not only associated with latent topics from o', but also associated with influencing aspects from its influencing objects. Thus, its generation is controlled by both the influence aspects (observed, latent) and the influencing object's latent topics.

Fig. 4a shows the aspect influence model LAIM which models the influence aspect as a latent state a. Fig. 5 shows

TABLE 1
Parameters for Aspect Influence Models

Symbol	meaning
o, o'	an influenced object and an influencing object
t, t'	tokens for o and o' respectively
t^a	observed aspects for a token
b	latent boolean variable
a	latent aspect variable
z, z'	latent topic states for o and o' respectively
$T(o)$	the number of tokens in the profile of object o
$R(o)$	the set of objects that influence o
D	the total number of objects
$\mathcal{D}$	the set of all the observed tokens in object profiles and contexts
T	the total number of distinct tokens
T^a	the total number of distinct observed aspects
Z	the total number of latent topic states
A	the total number of latent influence aspects
$\mathcal{A}$	the set of all the observed aspects
$\vec{\alpha}_\phi$	length-T hyperparameters $=(\alpha_\phi,\cdots,\alpha_\phi)$ to generate ϕ
$\vec{\alpha}_\theta$	length-Z hyperparameters $=(\alpha_\theta,\cdots,\alpha_\theta)$ to generate θ
$\vec{\alpha}_\eta$	length-A hyperparameters $=(\alpha_\eta,\cdots,\alpha_\eta)$ to generate η
$\vec{\alpha}_\psi$	length-T^a hyperparameters $=(\alpha_\psi,\cdots,\alpha_\psi)$ to generate ψ
$\vec{\alpha}_\gamma$	length-D hyperparameters $=(\alpha_\gamma,\cdots,\alpha_\gamma)$ to generate γ
$\vec{\alpha}_\beta$	hyperparameters$=(\alpha_{old},\alpha_{new})$ to generate β
ϕ	a $Z\times T$ matrix
θ, θ'	a $D\times Z$ matrix
γ	a $D\times T^a\times L(D)$ matrix, where $L(D)=max_o(R(o))$
η	a $D\times A$ matrix
β	a $D\times 2$ matrix
ψ	a $A\times T^a$ matrix



(a) Latent Aspect Influence Model (LAIM)

(b) Observed Aspect Influence Model (OAIM)

Fig. 4. Aspect influence models.

the detailed generative process for LAIM. In LAIM, there are two types of observed variables, object-profile tokens t and influence aspects t^a. In LAIM, we incorporate two types of objects: the objects that explicitly influence other objects (i.e., graph nodes with outgoing edges) and the objects that are explicitly influenced by other objects (i.e., graph nodes with incoming edges).

For objects that explicitly influence other objects, we model that their profile tokens t' arise from latent topic states z' by using LDA model. The topic mixture θ' is used to represent the probabilistic distribution of an object over all the possible latent topic states. θ' is generated using the Dirichlet distribution with the hyperparameter $\vec{\alpha}_\theta=(\alpha_\theta,\ldots,\alpha_\theta)$.

For objects that can be influenced by other objects (i.e., graph nodes with incoming edges), we model their profiles by considering their own content and connections from other objects. The profile of an object o is formed with both new information and inherited information from other objects that influence o. An object o's latent topic z (which is used to draw token t) can be drawn from its own topic mixture θ or its influencing objects' topic mixture θ'. This choice is modeled with a binary variable b which follows a

```
1) For every latent topic state $z$, draw $\vec{\phi}_z \sim dirichlet(\vec{\alpha}_\phi)$
2) For every latent aspect $a$, draw $\vec{\psi}_a \sim dirichlet(\vec{\alpha}_\psi)$
3) For every object $o$ that **can influence other objects**
   a) Draw $\vec{\theta}'_o \sim dirichlet(\vec{\alpha}_\theta)$
   b) For each position $pos$ in the profile of object $o$
      i) Generate $z'$ with $p(z'|o;\vec{\theta'}_o) \sim multi(\vec{\theta}'_o)$
      ii) Generate $t'$ with $p(t'|z';\vec{\phi}_{z'}) \sim multi(\vec{\phi}_{z'})$
4) For each object $o$ that **can be influenced by other objects**
   a) Draw $\vec{\theta}_o \sim dirichlet(\vec{\alpha}_\theta)$
   b) Draw $\vec{\eta}_o \sim dirichlet(\vec{\alpha}_\eta)$
   c) For each latent aspect $a$, draw $\vec{\gamma}_{o,a} \sim dirichlet(\vec{\alpha}_\gamma)$
   d) Draw the proportion between newly generated values and values generated
         from its influencing object
         $\beta_o \sim beta(\alpha_{old},\alpha_{new})$
   e) For each position $pos$ of the profile for object $o$, generate its token from
         either an influencing object or innovative state.
      i) Draw a coin $b$ with $p(b|o) \sim bernoulli(\beta_o)$
      ii) if $b=0$,
         A) Draw $z$ with $p(z|o;\vec{\theta}_o) \sim multi(\vec{\theta}_o)$
         B) Generate $t$ with $p(t|z,\vec{\phi}_z) \sim multi(\vec{\phi}_z)$
      iii) if $b=1$, draw an influence aspect $a$, an influencing object $o'$, and a topic
            state $z'$ for $o$ from $o'$
         A) Draw $a$ with $p(a|o,\vec{\eta}_o) \sim multi(\vec{\eta}_o)$
         B) Generate $t^a$ with $p(t^a|a,\vec{\psi}_a) \sim multi(\vec{\psi}_a)$
         C) Draw an interacting object $o'$ with $I(o \xrightarrow{a} o') \sim multi(\vec{\gamma}_{o,a})$
         D) Draw $z$=$z'$ with $p(z'|o',\vec{\theta'}_{o'}) \sim multi(\vec{\theta'}_{o'})$
         E) Generate $t$ = $t'$ with $p(t'|z',\vec{\phi}_{z'}) \sim multi(\vec{\phi}_{z'})$
```

Fig. 5. Generative process for LAIM: z', z, t', t, t^a, b, a, o' represent $z'_{o,pos}$, $z_{o,pos}$, $t'_{o,pos}$, $t_{o,pos}$, $t^a_{o,pos}$, $b_{o,pos}$, $a_{o,pos}$, $o'_{o,pos}$.

Bernoulli distribution with parameter β. β is generated using a Beta prior $\vec{\alpha}_\beta=(\alpha_{old},\alpha_{new})$. The choice of the influencing object o' follows a multinomial distribution with parameters $\vec{\gamma}_{o,a}$. $\vec{\gamma}_{o,a}$ captures the different influence degrees among objects on different aspects.

6.2 Observed Aspect Influence Model

The LAIM models an aspect as a latent state a and uses it to decide how o' affects the generation of o. Each a is represented as a distribution over the observed influence aspects. With the intuition that models with more variables are less efficient compared with models with less variables. We propose a second generative probabilistic model by treating observed influence aspects t^a as the influencing aspects. Fig. 4b shows this probabilistic model, which is denoted as OAIM. Fig. 6 shows the generative process for OAIM. This model contains less variables. It takes less time to learn OAIM than to learn LAIM given the same input. However, OAIM's results may not be as effective as LAIM. We compare these two different models in Section 8.5.

7 Model Learning

7.1 Gibbs Sampling Algorithm

We utilize the Gibbs sampling approach to learn the model parameters and latent states that control the generative process of LAIM and OAIM. Gibbs sampling is one of the Markov Chain Monte Carlo (MCMC) methods and is widely used to estimate the values of a set of latent variables. Gibbs sampling [10] allows to learn a model by iteratively updating each latent variable when fixing the remaining variables. The procedure of a typical Gibbs sampling algorithm is shown in Fig. 7.

The latent variables in the generative process of Fig. 5 are b, a, o', z', and z. The latent variables in the generative process of Fig. 6 are b, o', z', and z. To facilitate the

```
1) For every latent topic state $z$, draw $\vec{\phi}_z \sim dirichlet(\vec{\alpha}_\phi)$
2) For every object $o$ that **can influence other objects**
  a) Draw $\vec{\theta}'_o \sim dirichlet(\vec{\alpha}_\theta)$
  b) For each position $pos$ in the profile of object $o$
    i) Generate $z'$ with $p(z'|o;\vec{\theta'}_o) \sim multi(\vec{\theta}'_o)$
    ii) Generate $t'$ with $p(t'|z';\vec{\phi}_{z'}) \sim multi(\vec{\phi}_{z'})$
3) For each object $o$ that **can be influenced by other objects**
  a) Draw $\vec{\theta}_o \sim dirichlet(\vec{\alpha}_\theta)$
  b) For each observed aspect $t_a$, draw $\vec{\gamma}_{o,t_a} \sim dirichlet(\vec{\alpha}_\gamma)$
  c) Draw the proportion between newly generated values and values generated
        from its influencing object $\beta_o \sim beta(\alpha_{old}, \alpha_{new})$
  d) For each position $pos$ of the profile for object $o$, generate its token from
        either an influencing object or innovative state.
    i) Draw a coin $b$ with $p(b|o) \sim bernoulli(\beta_o)$
    ii) if $b = 0$,
      A) Draw $z$ with $p(z|o;\vec{\theta}_o) \sim multi(\vec{\theta}_o)$
      B) Generate $t$ with $p(t|z,\vec{\phi}_z) \sim multi(\vec{\phi}_z)$
    iii) if $b = 1$,
      A) Draw an interacting object $o'$ with $I(o' \xrightarrow{t^a} o) \sim multi(\vec{\gamma}_{o,t^a})$
      B) Draw $z$=$z'$ with $p(z'|o',\vec{\theta'}_{o'}) \sim multi(\vec{\theta'}_{o'})$
      C) Generate $t$ = $t'$ with $p(t'|z',\vec{\phi}_{z'}) \sim multi(\vec{\phi}_{z'})$
```

Fig. 6. Generative process for OAIM.

sampling process, we need to maintain the sampling counts of tokens that are assigned to different configurations. For objects that can be influenced by other objects, we use $N_{dv_1,\ldots,dv_i}[val_1,\ldots,val_i]$ to denote the number of tokens assigned to the configuration with $dv_1 = val_1, \ldots, dv_i = val_i$ during the sampling process. For example, $N_{o,a,o',b}[1,3,2,1]$ denotes the total number of tokens that are assigned to object o_1 and that come from object o_2 on aspect a_3 when the latent variable b is 1. For objects that can influence others, the count cache is denoted as N'. The meaning of each different count is detailed in [14].

The Gibbs sampling algorithm needs to sample the latent variables using update equations. The update equations used to learn LAIM are given in Eqs. (1)-(6). For a detailed derivation of all the equations, readers can refer to [14].

In each iteration i, the profile tokens for object o may be newly self-generated or is borrowed from some other objects. o's tokens come from other objects that influence o when b is 1. Let $T'(a)$ be the total number of tokens coming from other objects for influence aspect a, and let $T'(o',a)$ be the total number of tokens coming from a specific influencing object o' on aspect a. Then, the ratio $\frac{T'(o',a)}{T'(a)}$ can estimate the strength that o' influences o on aspect a, i.e., $I(o' \xrightarrow{a} o)$. The expected value of any parameter variables can be approximated by averaging over all the samples after the sampling chain converges [20]. The influence that o_k has over o_j on latent aspect a, $I(o_k \xrightarrow{a} o_j)$ can be estimated as $\frac{1}{M} \cdot \sum_{i=1}^{M} \frac{N_{o,a,o',b}(o_j,a,o_k,1)^{(i)}+\alpha_\gamma}{N_{o,a,b}(o_j,a,1)^{(i)}+|R(o_j)|\alpha_\gamma}$ where M is the total number of iterations of the converged sampling chain after burn-in, and the superscript $^{(i)}$ denotes the ith iteration. Similarly the influence that o_k has over o_j on the observed aspect t^a can be estimated as follows:

$$I(o_k \xrightarrow{t^a} o_j) = \frac{1}{M} \cdot \sum_{i=1}^{M} \frac{N_{o,t^a,o',b}(o_j,t^a,o_k,1)^{(i)} + \alpha_\gamma}{N_{o,t^a,b}(o_j,t^a,1)^{(i)} + |R(o_j)|\alpha_\gamma}. \tag{7}$$

OAIM. In learning OAIM, the update Eqs. (1), (2), (5), and (6) are still used. However, Eq. (3) is not needed for

```
Gibbs Sampling Algorithm
Input: Random variables to estimate $v_1, v_2, \ldots, v_n$
1) Randomly initialize $v_1, v_2, \ldots v_n = v_1^0, v_2^0, \ldots, v_n^0$
2) For iteration $i = 1, \ldots M$ or until converge
  a) Sample $v_1^i \sim p(v_1|v_2^{i-1}, v_3^{i-1}, \ldots, v_n^{i-1})$
  b) Sample $v_2^i \sim p(v_2|v_1^i, v_3^{i-1}, \ldots, v_n^{i-1})$
  c) ...
  d) Sample $v_n^i \sim p(v_n|v_1^i, v_2^i, \ldots, v_{n-1}^i)$
Output: $v_1, v_2, \ldots v_n = v_1^i, v_2^i, \ldots, v_n^i$
```

Fig. 7. Gibbs sampling algorithm.

OAIM because OAIM does not model latent influence aspect a anymore. In addition, Eq. (4) is changed to Eq. (8),

$$\begin{aligned} &p(o'_{pos}|\vec{t}, \vec{t'}, \vec{z}, \vec{z'}, \vec{o'}_{\neg pos}, \vec{t^a}, \vec{b}) \\ &\quad \propto \rho_{o=o',b1} \cdot \frac{N_{o,t^a,o',b}(o,t^a,o'_{pos},1) + \alpha_\gamma - 1}{N_{o,t^a,b}(o,t^a,1) + |R(o)|\alpha_\gamma - 1}. \end{aligned} \tag{8}$$

To implement the Gibbs sampling algorithm, we adopt the *blocking Gibbs sampling* strategy [2], which samples several variables together as a block to speed up the Gibbs sampling process for complex probabilistic models. In particular, to sample token t_{pos} for influenced objects, all the latent variables associated with position pos (i.e., b_{pos}, a_{pos} (for *LAIM*), o'_{pos}, z_{pos}) are sampled together.

Time complexity. Let M be the total number of iterations before the sampling process converges. For *influencing objects*, the sampling procedure is the same for both the LAIM and OAIM models. In each sampling iteration, the algorithm samples a new latent state $z_{pos} \in \{1,\ldots,Z\}$ for each token t_{pos}. The time complexity for sampling influencing objects in both LAIM and OAIM is $O(M \cdot D \cdot \bar{T}(o) \cdot Z)$, where $\bar{T}(o)$ is the average number of tokens in object o.

For the *influenced objects*, the algorithm in each iteration draws a new block of latent variables b_{pos}, a_{pos} (only for LAIM), o'_{pos}, and z_{pos} for each token t_{pos} in every object. Since we adopt the blocked Gibbs sampling strategy, the candidate size for a new sample at each position is the product of the domain size of latent variables. Thus, the time complexity of sampling influenced objects is $O(M \cdot D \cdot \bar{T}(o) \cdot Z \cdot \bar{R}(o) \cdot A)$ for LAIM and is $O(M \cdot D \cdot \bar{T}(o) \cdot Z \cdot \bar{R}(o))$ for OAIM, where $\bar{R}(o)$ is the average number of objects that influence the object o.

7.2 Parallel Gibbs Sampling Algorithm

The serial Gibbs sampling algorithm described in Section 7.1 cannot directly run in parallel. In one iteration, the serial Gibbs sampling algorithm calculates the conditional probability of every variable using the counts in Fig. 8. The counts are updated after drawing samples for each variable. These counts need to be shared in the calculation of multiple variables' conditional probabilities. Let us use shared counts to denote the counts that are needed in the calculation of multiple conditional probabilities. If some variables are sampled by directly applying this sampling principle in parallel, other variables' sampling processes need to wait until the shared counts are updated. We call the procedure of updating shared counts *synchronization*. The synchronization of the shared counts blocks the calculation of the conditional probability of the next variable.

$$p(b_{pos} = 0|\vec{t}, \vec{t'}, \vec{z}, \vec{z'}, \vec{o'}, \vec{a}, \vec{b}_{\neg pos}) \propto \rho_{o,z,b0} \cdot \frac{N_{o,b}(o,0) + \alpha_{new} - 1}{N_o(o) + \alpha_{old} + \alpha_{new} - 1} \quad (1)$$

$$p(b_{pos} = 1|\vec{t}, \vec{t'}, \vec{z}, \vec{z'}, \vec{o'}, \vec{a}, \vec{b}_{\neg pos}) \propto \rho_{o=o',b1} \cdot \frac{N_{o,b}(o,1) + \alpha_{old} - 1}{N_o(o) + \alpha_{old} + \alpha_{new} - 1} \quad (2)$$

$$p(a_{pos}|\vec{t}, \vec{t'}, \vec{z}, \vec{z'}, \vec{o'}, \vec{a}_{\neg pos}, \vec{t^a}, \vec{b}) \propto \frac{N_{o,a,o',b}(o, a_{pos}, o'_{pos}, 1) + \alpha_\gamma - 1}{N_{o,a,b}(o, a_{pos}, 1) + |R(o)|\alpha_\gamma - 1} \cdot \frac{N_{a,t^a,b}(a_{pos}, t^a_{pos}, 1) + \alpha_\psi - 1}{N_{a,b}(a_{pos}, 1) + T^a\alpha_\psi - 1} \cdot \frac{N_{o,a,b}(o, a_{pos}, 1) + \alpha_\eta - 1}{N_{o,b}(o,1) + A\alpha_\eta - 1} \quad (3)$$

$$p(o'_{pos}|\vec{t}, \vec{t'}, \vec{z}, \vec{z'}, \vec{o'}_{\neg pos}, \vec{a}, \vec{b}) \propto \rho_{o=o',b1} \cdot \frac{N_{o,a,o',b}(o, a_{pos}, o'_{pos}, 1) + \alpha_\gamma - 1}{N_{o,a,b}(o, a_{pos}, 1) + |R(o)|\alpha_\gamma - 1} \quad (4)$$

$$p(z_{pos}|\vec{t}, \vec{t'}, z_{\vec{\neg pos}}, \vec{z'}, \vec{o'}, \vec{a}, \vec{b}_{\neg pos}, b_{pos} = 0) \propto \rho_{z,t} \cdot \rho_{o,z,b0} \quad (5)$$

$$p(z_{pos}|\vec{t}, \vec{t'}, \vec{z}_{\neg pos}, \vec{z'}, \vec{o'}, \vec{a}, \vec{b}_{\neg pos}, b_{pos} = 1) \propto \rho_{z,t} \cdot \rho_{o=o',b1} \quad (6)$$

Given the following count ratio ρ,

$$\rho_{o=o',b1} = \frac{N'_{o',z',b=1}(o'_{pos}, z_{pos}, 1) + N'_{o,z}(o'_{pos}, z_{pos}) + \alpha_\theta - 1}{N'_{o',b=1}(o'_{pos}, 1) + N'_o(o'_{pos}) + Z\alpha_\theta - 1}$$

$$\rho_{z,t} = \frac{N_{z,t}(z_{pos}, t_{pos}) + N'_{z,t}(z_{pos}, t_{pos}) + \alpha_\phi - 1}{N_z(z_{pos}) + N'_z(z_{pos}) + T\alpha_\phi - 1}$$

$$\rho_{o,z,b0} = \frac{N_{o,z,b}(o, z_{pos}, 0) + \alpha_\theta - 1}{N_{o,b}(o,0) + Z\alpha_\theta - 1}$$

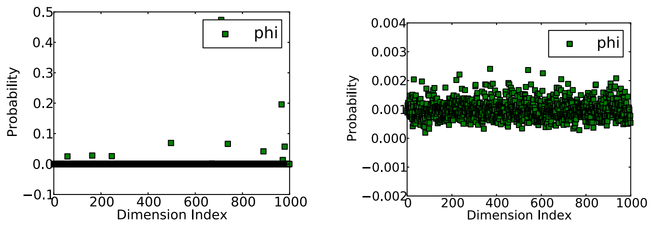
Fig. 8. Gibbs sampling updating equations.

Object-dependent property. We observe that a good property exists in the conditional probabilities of many variables. A conditional probability is *object-dependent* when its calculation depends only on one object's counts, but does not depend on shared counts.

We identify that the conditional probabilities that have object-dependent property are those calculated using Eqs. (1), (2), (4), and (8). These conditional probabilities can be calculated in parallel without synchronization. We also observe that the counts $N_{z,t}, N'_{z,t}, N_z, N'_z, N'_{o',z',b'}$ and $N_{a,t_a,b}$ are shared counts and need to be synchronized.

Lazy-update strategy for shared counts. For the shared counts, we propose and utilize a lazy-update strategy. Typical Gibbs sampling algorithms immediately update the counts after drawing samples for *each variable*. This updating operation creates much synchronization overhead in parallel implementation. Motivated by the blocked Gibbs sampling algorithm, which draws samples for a block of variables together to improve sampling efficiency, we propose a new strategy, which updates the shared counts in a lazy way. In particular, the shared counts are not updated after drawing samples for each variable. Instead, they are updated after finishing drawing samples for *all* the variables. Experimentally (Section 8.6.2, Figs. 20c, 20d), this strategy can achieve very similar effect as a serial Gibbs sampling algorithm.

Our parallel Gibbs sampling algorithm utilizes the object-dependent property and the lazy-update strategy for shared counts. For each object o, a thread is created to sample the variables for all the tokens in its profile. The threads for all the objects are submitted to a thread pool with a fixed size (# of cores). For a graph containing D objects, the parallel algorithm creates D threads, which are virtually running concurrently. Among these D threads, only a few number (# of cores) of threads are actually running. The other threads are either terminated or waiting. In running the serial Gibbs sampling in one thread, the counts in object-dependent conditional probabilities are updated after each variable is sampled. However, the shared counts are synchronized only after all the threads for one sampling iteration terminate.



(a) W=1000, $\alpha_\phi = 0.001$ (sparse distribution) (b) W=1000, $\alpha_\phi = 10$ (dense distribution)

Fig. 9. ϕ distribution generated using different α_ϕ.

8 Experiments

We implement both the non-parallel and parallel Gibbs sampling algorithms in Java. The aspect extraction algorithm is implemented in Python. The experiments are conducted on a workstation configured with 48 Intel(R) Xeon (R) cores (2.40 GHz) and 2,56G RAM, running GNU/Linux.

8.1 Data Sets

We use one synthetic data set and two real data sets to test the proposed methods. All the data sets are available at http://www.cs.nmsu.edu/dbdm/data/aspectinfluencedata.tar.

Synthetic data set. We generate synthetic data using the RMAT algorithm in [5] since it can generate graphs similar to real world networks. The synthetic graph has $\sim$1,500 nodes and $\sim$2,000 edges. The object profiles is generated using the generative process in Fig. 6 with $W = 1{,}000$, $Z = 10$, and $T^a = 100$. The hyperparameter values are set as $\alpha_\theta = 0.1$, $\alpha_\tau = 0.01$, $\alpha_\phi = 0.0001$, $\alpha_\beta = 0.5$, $\alpha_\gamma = 0.05$, $\alpha_\eta = 0.01$, $\alpha_\psi = 0.01$. The hyperparameters are set to be less than 1.0 because the Dirichlet distribution with small parameters (smaller than 1.0) generates a sparse multinomial distribution (Fig. 9a) while large parameters generate dense distributions (Fig. 9b). A sparse multinomial distribution means that most words have very small probability (close to zero) to occur in a document, which is consistent with the observations in topic models [4].

DBLP citation data set. We use the DBLP data set from [34]. The DBLP data set is preprocessed by removing the articles without abstracts. After the preprocess, the DBLP data set contains $\sim 325K$ abstracts (nodes) and $\sim 1M$ citations (edges). In total, there are $\sim 40M$ tokens among which $\sim 25K$ are distinct tokens.

CiteMisc data set. We extract a subset of the DBLP citation dataset and manually label sentences with aspects. We define four aspects, "background", "problem definition", "solution" and "experiment". Then, for the sentences that act as contexts, two researchers discuss and choose the aspect (out of the four aspects) that best captures the context semantic. We denote this data set as *CiteMisc*. *CiteMisc* contains 90 research articles and 71 citations. The total number of distinct tokens, T, is $\sim 1K$ and each article cites (i.e., is influenced by) ~ 5 articles on average. The articles in *CiteMisc* are selected on eight major research topics, including association rules, clustering, and topic modeling, etc.

Twitter50000 data set is crawled from the Twitter website in the period of Oct. 1 2014 and May 18 2015. It contains 50,000 Twitter users and their recent 200 tweets. A Breadth-First Search (BFS) strategy is utilized to crawl Twitter data. First, a seed set of Twitter users is randomly chosen. Then, for each user in the seed set, his/her friends are crawled and added to the seed set. Once a user's friends are crawled, this user is removed from the seed set. This BFS crawling strategy can guarantee that all the collected Twitter users form a connected graph. The number of tokens, distinct tokens, and follow relationships are $\sim 90M$, $\sim 200K$, and $\sim 50K$ respectively. Each Twitter user in this data set follows ~ 200 users on average. This data set embeds ~ 400 observed aspects, which are the categories tagged to the tweets.

8.2 Baseline Approaches

We use Latent Dirichlet Allocation [4] as our first baseline model. Our second baseline model is Citation Influence Model (CIM) [7]. All the baseline approaches are used for modeling the generative process of graphs.

8.3 Evaluation Measurement

8.3.1 Effectiveness

Effectiveness of aspect extraction. We use the classification accuracy to measure the aspect extraction method in Section 5.

Effectiveness of influence learning. We examine the *convergence and the correctness* of our learning algorithm on the synthetic data set using a similar idea in [11]. After generating the synthetic data, we treat the parameters that are used to generate the synthetic data as ground truth. Then, we run the learning algorithms to learn the parameters. The learned parameters are compared with the ground truth parameters to verify the correctness and the convergence of our Gibbs sampling algorithm.

(1) KL Divergence. KL divergence measures the difference between two probability distributions. We use it to measure the difference between ground truth parameters and the learned parameters. Let ϕ denote the ground truth values for one parameter ϕ ($p(t|z)$) and $\hat{\phi}$ be the learned values. We calculate the KL divergence of $\hat{\phi}_i$ from ϕ_j for every pair (i,j) where $i,j \in \{1,\dots Z\}$. The KL divergence of $\hat{\phi}_i$ from ϕ_j is denoted as $D_{KL}(\phi_j||\hat{\phi}_i)$ and is calculated as $D_{KL}(\phi_j||\hat{\phi}_i) = \sum_{t=1}^{T} \phi_{jt} \ln \frac{\phi_{jt}}{\hat{\phi}_{it}}$.

(2) Log-likelihood. The likelihood over unseen data is calculated on tokens in the profiles of both influencing and influenced objects to compare the generalization capabilities of the baseline approaches and our models, LAIM and OAIM. The formulas we use to calculate likelihood for influenced objects of CIM, OAIM, LAIM are Eqs. (10), (11), and (12) in Fig. 10 respectively. The likelihood for influencing objects of OAIM, LAIM, CIM, and LDA models can be calculated using Eq. (9) in Fig. 10.

$$\begin{aligned} p(\vec{t'_o}) &= \prod_{i=1}^{T(o')} \Bigg(\sum_{z'=1}^{Z} p(t'_i|z_i = z') \cdot p(z_i = z'|o') \Bigg) \\ &= \prod_{t'=1}^{T(o')} \Bigg(\sum_{z'=1}^{Z} \phi_{z',t'} \cdot \theta'_{o',z'} \Bigg) \end{aligned} \tag{9}$$

$$\begin{aligned} p(\vec{t}_o)_{CIM} = \prod_{t=1}^{T(o)} \Bigg(& \beta_{o,new} \cdot \big(\sum_{z=1}^{Z} \phi_{z,t} \cdot \theta_{o,z} \big) \\ & + \beta_{o,old} \cdot \big(\sum_{o' \in R(o)} \sum_{z'=1}^{Z} \gamma_{o,o'} \cdot \theta'_{o',z'} \cdot \phi_{z',t} \big) \Bigg) \end{aligned} \tag{10}$$

$$\begin{aligned} p(\vec{t}_o)_{OAIM} = \prod_{t=1}^{T(o)} \Bigg(& \beta_{o,new} \cdot \big(\sum_{z=1}^{Z} \phi_{z,t} \cdot \theta_{o,z} \big) \\ & + \beta_{o,old} \cdot \big(\sum_{o' \in R(o)} \sum_{z'=1}^{Z} \gamma_{o,t_i^a,o'} \cdot \theta'_{o',z'} \cdot \phi_{z',t} \big) \Bigg) \end{aligned} \tag{11}$$

$$\begin{aligned} & p(\vec{t}_o)_{LAIM} \\ & = \prod_{t=1}^{T(o)} \Bigg(\beta_{o,new} \cdot \big(\sum_{z=1}^{Z} \phi_{z,t} \cdot \theta_{o,z} \big) \\ & + \beta_{o,old} \cdot \big(\sum_{a=1}^{A} \sum_{o' \in R(o)} \sum_{z'=1}^{Z} \eta_{o,a} \cdot \gamma_{o,a,o'} \cdot \theta'_{o',z'} \cdot \phi_{z',t} \big) \Bigg) \end{aligned} \tag{12}$$

Fig. 10. Predictive likelihood functions.

(3) Perplexity. Perplexity measures how well a probability distribution predicts a sample. The perplexity of an influencing object or an object in the LDA model o' is calculated as $perp(o') = \exp\{-\frac{\log p(\vec{t'_o})}{T(o')}\}$ [4]. The perplexity of an influenced object o in CIM, OAIM, and LAIM are $perp_{CIM}(o) = \exp\{-\frac{\log p(\vec{t}_o)_{CIM}}{T(o)}\}$, $perp_{OAIM}(o) = \exp\{-\frac{\log p(\vec{t}_o)_{OAIM}}{T(o)}\}$, and $perp_{LAIM}(o) = \exp\{-\frac{\log p(\vec{t}_o)_{LAIM}}{T(o)}\}$ respectively.

(4) Precision@K. To quantitatively evaluate the aspect-level influence, we manually rank the papers that are cited by a paper on each aspect according to the influence degrees. Then the aspect-level influence degrees, which are calculated using Eq. (7), are compared with the manual rank. Precision@3 is calculated on each aspect.

(5) Case studies. We subjectively verify the results of LAIM and OAIM by showing the learned results of several graph nodes on *Twitter50000* and *CiteMisc* data sets.

8.3.2 Efficiency

Running time. We compare the running time of both the non-parallel and the parallel Gibbs sampling algorithms to demonstrate that the actual running time is consistent with our time complexity analysis. We also test the learning algorithms on data sets of various sizes to show the scalability of both OAIM and LAIM.

Speedup ratio. To evaluate our parallel Gibbs sampling algorithm, we conduct three sets of experiments to compare its speedup ratio with the ideal case by varying the number of cores, the graph size, and the model parameters.

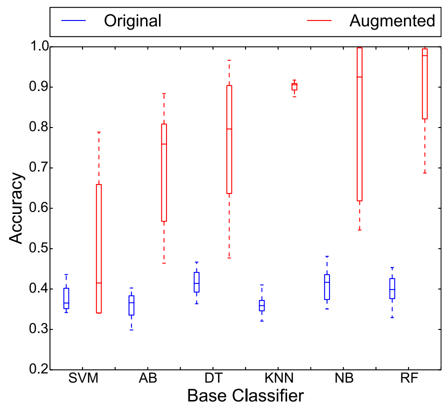


Fig. 11. Ten-fold cross validation result of six base classifiers ($W = 10$, *CiteMisc* data set).

8.4 Effectiveness of Aspect Extraction

We evaluate the effectiveness of the merging-based aspect extraction algorithm, presented in Section 5, using the *CiteMisc* data set, for which we manually label the aspects of each sentence in the abstracts. We run 10-fold cross validation and calculate the averaged accuracy. The 10-fold cross validation is conducted as follows. First, it partitions the *CiteMisc* data set to 10 partitions. Then, it chooses one partition as testing set and uses the original instances and the augmented instances from the other remaining nine partitions as training set to train a classifier. The classifier is then used to predict the aspects of testing instances. The accuracy of the classifier is then calculated. This training-testing process is conducted 10 times with each time uses a different partition (from the 10 partitions) as testing set. The averaged accuracy from the 10 executions is reported.

We utilize six classification algorithms implemented in:

- Support Vector Machine (SVM) with Radial Basis Function (RBF) kernel (kernel parameter $\gamma = 1$),
- Decision Tree (DT) with entropy as splitting criteria,
- AdaBoost (AB) with Decision Tree as the base estimator where the number of estimators is 100 and the learning rate is 1,
- Random Forest (RF) with Decision Tree as the base estimator where the number of estimators is 100,
- Naive Bayesian (NB) with parameter $\alpha = 1$,
- K-Nearest Neighbor (KNN) with $k = 1$.

We test the merging-based strategy by varying the window size from 2 to 20. Our results indicate that the aspect extraction algorithm achieves the best performance when the window size is 10. We report the results for $W = 10$, using which the algorithm generates 3,104 augmented instances.

We plot the accuracies using a box plot in Fig. 11. In this figure, the blue box and the red box represent the quartile statistics of the extraction accuracy on the original and the augmented data sets respectively. For each box, the three short horizontal lines from top to bottom represent the 75th percentile, median, and 25th percentile of the 10 accuracy values. The symbols "-" at the end of the dashed line are the minimum and maximum accuracies.

Fig. 11 shows that classifiers trained on the augmented data sets have significant higher accuracies than classifiers trained on the original data sets. This improvement comes from the merging-based aspect extraction. On the augmented data set the critical features are augmented. We also observe that the most robust classifier is KNN. Its accuracy variance is much smaller than the variances of other classifiers. We thus choose KNN as the base classifier of the Merging-based Aspect Extraction method when learning the aspect-level influence.

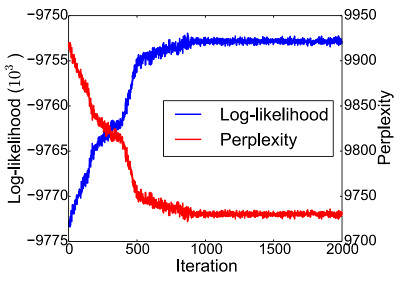


Fig. 12. Log-likelihood and perplexity (*Synthetic* data set).

8.5 Effectiveness of Models and Algorithms

This section presents the results of our proposed influence models and model learning algorithms. Section 8.5.1 shows the correctness and convergence of our Gibbs sampling algorithm. Section 8.5.2 presents the log-likelihood of the learned models. Section 8.5.3 presents the performance of OAIM on *CiteMisc* data set using the Precision@3 measurement. Section 8.5.4 subjectively demonstrates the usefulness of the results from OAIM and LAIM through case studies on both *Twitter50000* and *DBLP* data sets.

8.5.1 *Correctness and Convergence of the Gibbs Sampling Algorithm*

Convergence. We check the convergence of the learning algorithm by examining the trend of the log-likelihood and the perplexity (Fig. 12). The process converges after about iteration 900 since the log-likelihood and perplexity stabilize after that.

Correctness. To show that our Gibbs sampling algorithm can correctly learn the ground truth parameters, we examine the differences between the ground truth values and the learned values on one parameter ϕ ($p(t|z)$). The KL divergence of all the $\hat{\phi}_i$s (learned parameters) from all the ϕ_js (ground truth parameters), where $i, j \in \{1, \ldots Z\}$, is shown in Fig. 13. The x and y axes are the indexes for ϕ and $\hat{\phi}$ respectively. The darker the pixel at (i, j) is, the smaller the KL-divergence of $\hat{\phi}_i$ from ϕ_j is.

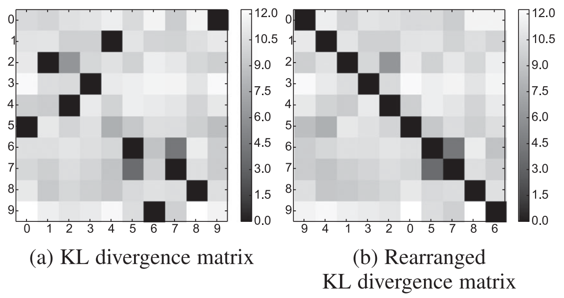
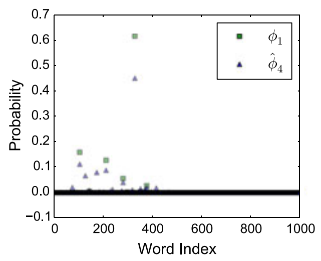
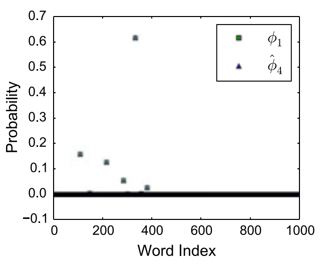

(a) KL divergence matrix
(b) Rearranged KL divergence matrix

Fig. 13. Comparison of learned parameters and ground truth.

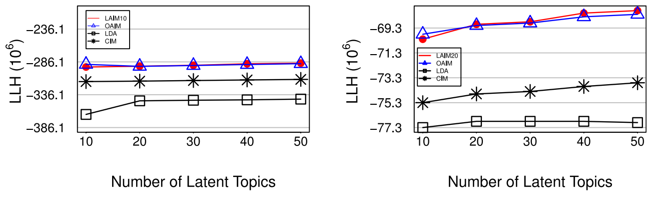


(a) Iteration 10



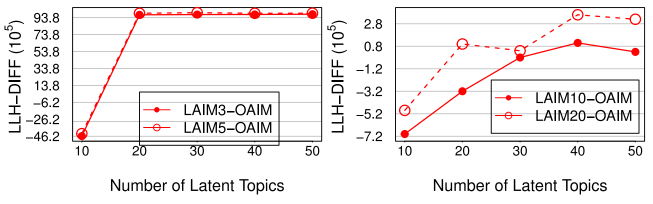
(b) Iteration 500

Fig. 14. Comparison of learned $\hat{\phi}_4$ and ground truth ϕ_1.



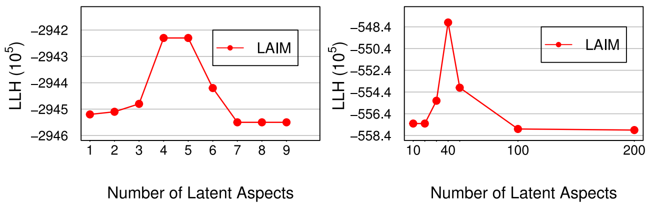
(a) *DBLP* (b) *Twitter50000*

Fig. 15. Compare baseline approach with OAIM and LAIM.



(a) *DBLP* (b) *Twitter50000*

Fig. 16. Comparing OAIM and LAIM: log-likelihood difference versus Z.



(a) *DBLP* (b) *Twitter50000*

Fig. 17. LAIM log-likelihood versus A ($Z = 10$).

Fig. 13a shows the KL-divergence between each pair of $\hat{\phi}_i$ and ϕ_j. After rearranging Fig. 13a by swapping columns, we get Fig. 13b. For each ϕ_j, its corresponding learned parameter is $\hat{\phi}_i$ with the minimal KL divergence. As shown in Fig. 13, there is a one to one mapping between ϕ_j and $\hat{\phi}_i$. These results indicate that our algorithm can correctly learn the parameters.

We further examine the details of the learning process by plotting $\hat{\phi}_4$ and its matching ground truth ϕ_1 in Fig. 14. At the beginning of the learning process, $\hat{\phi}_4$ is very random and does not match with ϕ_1. As the learning process continues, $\hat{\phi}_4$ quickly stabilizes and converges to ϕ_1, which is consistent with the trend of log-likelihood and perplexity in Fig. 12. After the model converges, most values are close to zero because the hyperparameter values that are used to generate the ground truth parameter are 0.01. Other parameters show similar behavior.

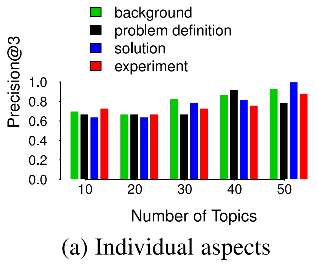
8.5.2 Log-Likelihood of the Learned Models

The log-likelihood for LAIM, OAIM, and baseline methods (Figs. 15, 16, 17) are calculated. The reported log-likelihood of each model is calculated after the model converges. In these figures, LAIMx means that the number of latent aspects A is x.

We first compare baseline approaches with our LAIM and OAIM models and show the results in Fig. 15. On the *DBLP* and *Twitter50000*, in which the number of observed aspects T^a is 4 and ~400 respectively, OAIM and LAIM get much better log-likelihood than baseline approaches. The OAIM, LAIM, and CIM perform better than LDA because LDA ignores graph structures while the influence models capture graph structures as a part of the generative process. The OAIM and LAIM outperform CIM because OAIM and LAIM model aspects which CIM does not capture. These results show that baseline approaches are not as effective as our Aspect Influence Models in modeling the generative process of the graphs.

The second set of experiments is conducted to compare the LAIM and OAIM models because the difference between the performance of OAIM and LAIM is not obvious in Fig. 15. For this set of tests, we plot the difference of the log-likelihood values (LLH-DIFF) between LAIM and OAIM. Fig. 16 shows the results when we vary the number of latent topic states (Z). When *LLH-DIFF* is smaller than zero, it means that OAIM gets bigger log-likelihood value than LAIM, and OAIM performs better than LAIM accordingly. Fig. 16 shows that when the number of latent topic states Z is small ($Z = 10$), OAIM performs slightly better than LAIM (with negative *LLH-DIFF* value). When Z is big ($Z > 10$), LAIM performs better than OAIM (with positive *LLH-DIFF* value). This trend is observed because, when the number of topics is small, OAIM can capture the graph generative process very well and LAIM uses more variables which may overfit the graph; when the number of topics increases, the complexity of the model increases and OAIM starts to underfit the generative process of graphs. This trend is also observed in [31].

The third set of experiments is to evaluate LAIM's performance when changing the number of latent aspects, A. Fig. 17 plots the results when we vary A. The performance of LAIM improves with the increase of A when A is small. We can observe that LAIM's log-likelihood value on unseen data increases as A increases from 1 to 4 in Fig. 17a, and as A increases from 10 to 40 in Fig. 17b. This is because latent variables form a set of features extracted from observed variables; a model using more latent variables loses less information, thus shows higher likelihood in the testing samples. LAIM's performance stabilizes or slightly decreases once it reaches a point. For the *DBLP* data set, which has four observed aspects, LAIM's performance is the best when A is 5. For *Twitter50000* data set, which has ~400 observed aspects, the best performance is achieved when A is 40. When there are more latent variables, the parameter matrices are sparser, which decreases the prediction capability of the model. This is consistent with the trend observed in [4], which demonstrates that proper number of latent variables give the best performance, and less or more latent variables lead to underfitting or overfitting.



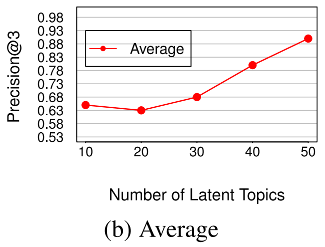


Fig. 18. Precision@3 for *CiteMisc* (OAIM model).

8.5.3 Precision@K

We measure the precision of the top-K results on *CiteMisc* data set. For each citing article, its corresponding domain experts (Ph.D. students who work in that domain) rank the article's citations. For each aspect $t^a \in \mathcal{A}$, domain experts give a list of ranked citations, in which the ranking criterion is the influence degree from the cited article to the citing article on the aspect t^a. The manually ranked citation lists are used as ground truth. After OAIM converges, Eq. (7) is used to estimate the influence degree on each aspect. Then, we calculate the Precision@3 of the results discovered by the OAIM model on each aspect. The precisions on the four aspects and the average precisions are plotted in Fig. 18. The results show that OAIM can find results with high precision for different numbers of latent topics, and the precision increases with the number of latent topics.

8.5.4 Case Studies

This section *subjectively* verifies the Aspect Influence Models. After running our models on both the *Twitter50000* and *CiteMisc* data sets, we select several graph objects from these two graphs and show their most influencing objects on several aspects. Note that the experiments were run on the whole data sets. The results for case studies are only reported on several objects which can be easily verified. We cannot show the results on all the nodes because of space limitation.

For the *Twitter50000* data set, we choose a few famous users and analyze the aspects on which they are influenced. These famous users are selected because it is easy to subjectively verify the findings from our models. Table 3 shows a sample of results from our OAIM model. For instance, a user *elisefoley*, who is a reporter of a newspaper Huffington Post, is influenced on the aspect *Election Day/Night 2012* by user *TeamRomney* the most. This information shows that this reporter's interest is greatly influenced by Romney's team in the election. In addition, *elisefoley* is influenced on the aspect *National Security* by another user *marcambinder*, who is inside the Government Secrecy Industry. Similarly, a user *dcharlesReuters*, who is Reuters correspondent in Washington, covering homeland security issues, is influenced on the aspect of *national politics* by the *WSJ* (i.e., Wall Street Journal) the most. We observe that OAIM model is returning us meaningful results from several Twitter users whom we can verify subjectively.

We also report the topic-level influence among the users in Table 3 to show the superiority of our aspect-level influence. We ran our topic discovery using Z=100 to match the number of observed aspects. Then, for each aspect in Table 3, we chose several topics that best match the aspects by manually checking the semantics of the topics. The third column of Table 3 shows the topics and Table 2 shows the top representative words of these topics. For user *elisefoley*, her aspect *Election Day/Night 2012* is best matched to topic z_2 which is related to the election. This aspect and topic represent similar semantics. Similarly, user *WestWingReport*'s second aspect represents similar semantics as topic z_5. However, results also show that *many aspects captures the influence dimension more accurately*. For example, user *elisefoley*'s second aspect *National Security* is more specific than its best matched topic z_1, which can be summarized as *cyber security*. User *dcharlesReuters*'s results also show that aspects capture the semantics more accurately than topic $z4$, which is hard to summarize. Aspects such as *thinkprocess-staff* cannot be matched with any discovered topics. These case studies demonstrate that topics are not able to capture the *specific* dimensions on which users are connected on social networks. This is consistent with our modeling process which uses the explicit aspects; explicit aspects can better capture the reasons of connections.

To give a concrete idea about the results of *LAIM*, we show several users and their most influencing users on two latent influence aspects in Table 4. E.g., user *dcharlesReuters* is most influenced by *GStephanopoulos* (a chief political correspondent in ABC) on latent aspect A_8. As shown in Table 4b, A_8 has higher probability on observed aspects related to news. This makes sense because *dcharlesReuters* is a correspondent and his Twitter account activity shows that he is influenced more on *news* aspect.

For the case studies of Table 4, we find the best matched topics to the latent aspects. The latent aspects A_4 and A_8 can be summarized as *foreign* and *news* respectively. However, they are both matched to topic

TABLE 2
Topics and Representative Words

Topic	Representative words
z_1	cyberattack, reject, communicate, cognition
z_2	obama, us, report, vote, today, new
z_3	job, manage, hire, business
z_4	practition, gmail cleanup, newborn, gun, ambience
z_5	us, news, world, people, apple

TABLE 3
Top Influence Relationships Discovered by OAIM on *Twitter50000*

User u	Top aspects	Best matched topic	Top influencing users
elisefoley	Election Day/Night 2012	z_2	TeamRomney, samyoungman
	National Security	z_1	marcambinder, rozen
dcharlesReuters	national politics	z_4	WSJ, Lis Smith
	US politics	z_4	WSJ, HuffPostPol
aterkel	thinkprocess-staff	NA	brainstelter, ThinkProgress
	self_created	NA	David_Ingram, jaketapper
ReutersPolitics	Reporters	z_5	ZekeJMiller, mattspetalnick
	self_created	NA	NewsHour, ReutersWorld
WestWingReport	self_created	NA	JimPethokoukis, BillGates
	Journalism	z_5	mitchellreports, nytimes

TABLE 4
Top Influence Relationships Discovered by LAIM on *Twitter50000* (There are Overall A=10 Latent Influence Aspects)

A user u	Aspect	Best matched topic	Top influencing user
dcharlesReuters	latent aspect 8	z_5	GStephanopoulos
WestWingReport	latent aspect 3	z_3	brianstelter
rickklein	latent aspect 4	z_5	GStephanopoulos

(a) Three users with the top-1 most influencing latent aspects

A_3 (economy)	A_4 (foreign)	A_8 (news)
WSJ	NGB	National Journal on Twitter
OFA-States	Foreign Office on Twitter	Debate Watch 2012
WSJ staff	FCO Ministers	HotlineOnCall
Debate Watch 2012	London2012	newzhoundz
HotlineOnCall	Sochi 2014	Election Night

(b) Latent aspects A_3, A_4, and A_8

z_5. This shows that aspects capture more specific information (with finer granularities) due to the usage of observed aspects.

Table 5 shows the results on *CiteMisc* data set with the parameter setting $Z = 10$. There are four observed aspects (i.e., $T^a = 4$). For each influenced seed article, we show the most influencing (i.e., top-1) article on two aspects, *background* and *solution*, which OAIM discovers from the *CiteMisc* data set. For instance, the article p_3 *Scalable Algorithms for Association Mining* is most influenced by the article *Set-Oriented Mining for Association Rules in Relational Databases* on the *Background* aspect because they all study the association rule mining problem. But when it comes to the *Solution* aspect, the article *Sampling Large Databases for Association Rules* influences p_3 more because they all work on scalable large data processing. These discoveries are consistent with our subjective understanding on these research articles.

8.6 Efficiency of the Learning Algorithms

When we test the efficiency of the learning algorithms, we extract nine smaller graphs from *Twitter50000*: *Twitter100*, *Twitter500*, *Twitter1000*, *Twitter2000*, *Twitter5000*, *Twitter10000*, *Twitter20000*, *Twitter30000*, *Twitter40000*. The total number of tokens in these nine sub-graphs varies from $\sim 0.5M$ to $\sim 50M$.

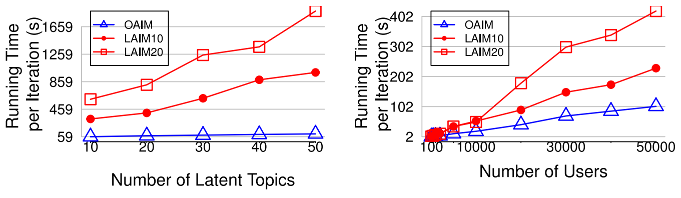
8.6.1 Learning Algorithms Comparison for LAIM and OAIM

We first compare the running time of the serial Gibbs sampling algorithms on learning two models LAIM and OAIM. The per-iteration running time of LAIM10, LAIM20, and OAIM with different number of latent topics (Z) is collected on *Twitter50000* data set. The result is shown in Fig. 19a. We can see that OAIM is much faster than LAIM no matter whether the number of latent aspects A is 10 or 20. The running time of both LAIM and OAIM grows linearly in the number of latent topics.

We show how these two models scale in graph sizes. The per-iteration running time of LAIM10, LAIM20, and OAIM on *Twitter50000* and its nine sub-graphs is shown in Fig. 19b. Fig. 19b shows that both models scale linearly in the graph size. These results are consistent with our complexity analysis in Section 7.1.

TABLE 5
Top Influencing Articles Discovered by OAIM for the Three Seed Articles in the *CiteMisc* Data Set

id	Article	Top-2 aspects	Top influencing article
p_1	Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and its Applications	Background	BIRCH: An Efficient Data Clustering Method for Very Large Databases
		Solution	A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise
p_2	Unsupervised Prediction of Citation Influences	Background	The Author-Topic Model for Authors and Documents
		Solution	Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks
p_3	Scalable Algorithms for Association Mining	Background	Set-Oriented Mining for Association Rules in Relational Databases
		Solution	Sampling Large Databases for Association Rules



(a) Time vs. Z (b) Time vs. graph size

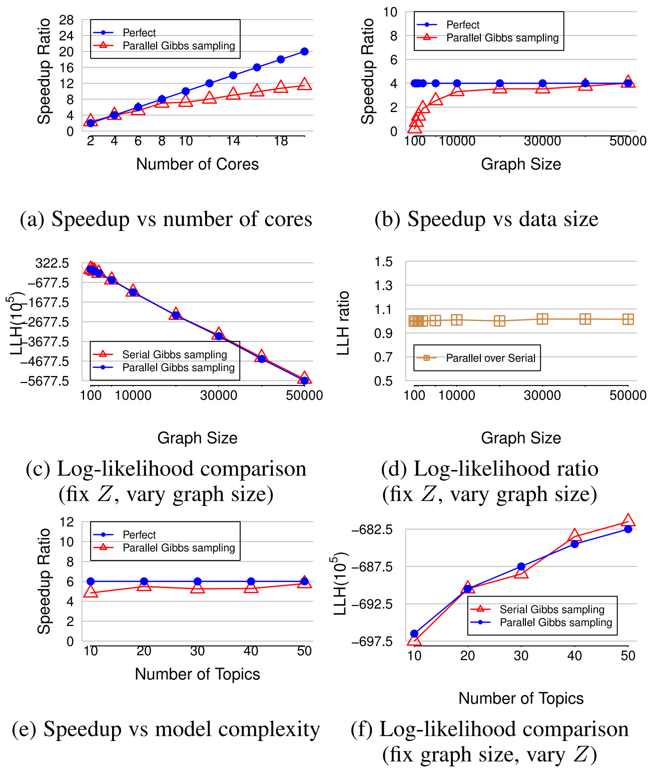
Fig. 19. Comparing running time of LAIM and OAIM.

We also check the convergence of LAIM and OAIM on the *Twitter50000* data set, as what we have done on the *Synthetic* data sets in Section 8.5.1. The convergence criterion we adopt is $\hat{R}$ value [9]. The smaller the $\hat{R}$ value is, the closer the sampling chain is to convergence. We set the number of iterations in the initial *burnin* phase to be 100. When the $\hat{R}$ value of the sampling chain is less than 1.01 [23], the Gibbs sampling algorithm is treated as converged. The number of iterations before the convergence for LAIM and OAIM on different sets of parameters and different sizes of *Twitter* data sets varies in the range of $[20, 50]$.

8.6.2 Parallel Gibbs Sampling Algorithm

We design experiments to show the performance of the parallel Gibbs sampling algorithm (to learn OAIM). Fig. 20 shows the efficiency of the parallel algorithm that we describe in Section 7.2. Both the non-parallel and the parallel Gibbs sampling algorithms are tested on the *Twitter50000* data set and its nine sub-graphs. We compare the speedup ratio of our parallel algorithm with the ideal parallel speedup ratio by testing the algorithms with different number of cores, graph sizes, and model complexities.

The speedup ratio of a parallel algorithm with n cores over its corresponding serial algorithm has an upper bound, which is n, according to Amdahl's Law [25]. In the ideal parallel case, every computation step of a serial algorithm can be parallelized, which produces the perfect speedup ratio n. When evaluating our parallel Gibbs sampling, we



(a) Speedup vs number of cores (b) Speedup vs data size

(c) Log-likelihood comparison (fix Z, vary graph size)

(d) Log-likelihood ratio (fix Z, vary graph size)

(e) Speedup vs model complexity (f) Log-likelihood comparison (fix graph size, vary Z)

Fig. 20. Parallel Gibbs sampling results.

compare the speedup ratio of the parallel Gibbs sampling with the perfect speedup ratio.

Fig. 20a shows the speedup ratio of the parallel Gibbs sampling algorithm when increasing the number of cores used in running the parallel algorithm. When the number of threads is small (2-6), the speedup ratio is close to perfect. When the number of threads is more than six, the speedup ratio is a little less than the perfect ratio. This decrease of performance is due to the overhead of thread scheduling when the number of threads increases. Despite the decrease, it can be observed that the speedup ratio still grows linearly in the number of cores. Overall, the trend in Fig. 20a shows that the speedup ratio of our parallel Gibbs sampling is close to the ideal speedup ratio, which indicates that our algorithm can make good use of the multiple cores in a system.

We also test the performance of the parallel sampling algorithm on various sizes of data to demonstrate how the algorithm scales in graph sizes. For this test, we fix the number of cores to be four, and utilize the *Twitter50000* and its nine sub-graphs. The results are shown in Fig. 20b. The figure shows that the parallel algorithm does not outperform the serial one on small data sets. It is because the thread scheduling overhead takes more time than computations on small data sets. When the data set size grows, the speedup ratio of the parallel algorithm becomes close to perfect. This indicates that the parallel algorithm has better speedup on larger data sets. This figure also shows that the parallel implementation scales well on real world large data sets.

We report the log-likelihood trend of the parallel and serial Gibbs sampling algorithms when graph sizes vary to verify the correctness of our parallel Gibbs sampling algorithm. Fig. 20c compares the log-likelihood values of the parallel and serial Gibbs sampling algorithms. We can see that there is no major difference between the log-likelihood values of the two algorithms for the same graph size. To further investigate their differences, we plot the ratio of the log-likelihood of the parallel Gibbs sampling algorithm over the serial Gibbs sampling algorithm in Fig. 20d. All the ratios are close to 1. These results show that the parallel algorithm can speed up the learning process, and at the same time guarantees the effectiveness.

We also explore how the parallel algorithm scales in the model complexity by changing the number of latent topics (Z). For this experiment, we fix the number of cores to be six and the data set to be *Twitter50000*. The results are shown in Fig. 20e. It shows that when the model becomes more complicated (with a greater Z), the parallel Gibbs sampling can achieve better speedup ratio. This demonstrates that the parallel Gibbs sampling algorithm can achieve better speedup ratio when the model complexity increases.

Fig. 20f compares the log-likelihood of OAIM learned using the parallel Gibbs sampling (with lazy-update) and the serial Gibbs sampling on the *Twitter50000* data set. For different numbers of latent topics, the log-likelihood values of two Gibbs sampling algorithms do not differ much. This result shows that the parallel Gibbs sampling algorithm can speed up the learning of the Aspect Influence Model without sacrificing effectiveness.

9 Conclusions and Future Work

We study the problem of detecting influence relationships at aspect level from graphs. In particular, these influence relationships capture in which context (influence aspect) and how strong (influence degree) that one object influences another. We first propose a semi-supervised Merging-based Aspect Extraction Algorithm to automatically extract aspects from graphs. We then design two probabilistic models, OAIM and LAIM, to capture and represent these influence relationships. We design blocking Gibbs sampling algorithms to learn the probabilistic models. We further design a parallel Gibbs sampling algorithm by utilizing the object-dependent property and the lazy-update strategy. Extensive experiments on synthetic and real data sets show that the aspect extraction algorithm can label influence aspects very accurately, and the two probabilistic models can generate meaningful results. Efficiency tests of the serial and parallel Gibbs sampling show that the learning algorithms scale linearly in the size of data sets, the complexity of the models, and the number of cores.

In the future, we will extend our aspect-level model to biological pathway study by considering a special characteristic of such data. The characteristic is that multiple molecules *together* influence another molecule under a certain context. This type of influence at the aspect level is very meaningful. However, to discover the aspect-level influence from such data, we need to consider the combined nature of multiple graph nodes.

Acknowledgments

This work is supported by US National Science Foundation (NSF) #1337884 and #1345232.

REFERENCES

[1] C. C. Aggarwal and C. Zhai, "A survey of text classification algorithms," *Mining Text Data*. New York, NY, USA: Springer, 2012, pp. 163–222.

[2] C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan, "An introduction to MCMC for machine learning," *Mach. Learn.*, vol. 50, no. 1-2, pp. 5–43, 2003.

[3] N. Barbieri, F. Bonchi, and G. Manco, "Who to follow and why: Link prediction with explanations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2014, pp. 1266–1275.

[4] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, 2003.

[5] D. Chakrabarti, Y. Zhan, and C. Faloutsos, "R-MAT: A recursive model for graph mining," in *Proc. SIAM Int. Conf. Data Mining*, 2004, pp. 442–446.

[6] C. P. Diehl, G. Namata, and L. Getoor, "Relationship identification for social network discovery," in *Proc. 22nd Nat. Conf. Artif. Intell.*, 2007, pp. 546–552.

[7] L. Dietz, S. Bickel, and T. Scheffer, "Unsupervised prediction of citation influences," in *Proc. Int. Conf. Mach. Learn.*, 2007, pp. 233–240.

[8] E. Garfield, "Citation indexes for science. A new dimension in documentation through association of ideas," *Int. J. Epidemiol.*, vol. 35, no. 5, pp. 1123–1127, 2006.

[9] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin, *Bayesian Data Analysis*. London, U.K.: Chapman and Hall, 2003.

[10] W. Gilks, S. Richardson, and D. Spiegelhalter, *Markov Chain Monte Carlo in Practice*. London, U.K.: Chapman & Hall, 1996.

[11] T. L. Griffiths and M. Steyvers, "Finding scientific topics," *Proc. Nat. Acad. Sci.*, vol. 101(suppl. 1), pp. 5228–5235, 2004.

[12] J. E. Hirsch, "An index to quantify an individual's scientific research output," *Proc. Nat. Acad. Sci.*, vol. 102, no. 46, pp. 16569–16572, 2005.

[13] T. Hofmann, "Probabilistic latent semantic indexing," in *Proc. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 1999, pp. 50–57.

[14] C. Hu and H. Cao (2014). Detecting influence relationships from graphs. New Mexico State Univ., Las Cruces, NM, USA. [Online]. Available: http://www.cs.nmsu.edu/wp13/nmsu-files/uploads/2014/01/TR-CS-NMSU-2014-1-13.pdf

[15] T. Iwata, A. Shah, and Z. Ghahramani, "Discovering latent influence in online social activities via shared cascade poisson processes," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2013, pp. 266–274.

[16] G. Joshi-Tope, M. Gillespie, I. Vastrik, P. D'Eustachio, E. Schmidt, B. de Bono, B. Jassal, G. Gopinath, G. Wu, L. Matthews, S. Lewis, E. Birney, and L. Stein, "Reactome: A knowledgebase of biological pathways," *Nucleic Acids Res.*, vol. 33(suppl 1), pp. D428–D432, 2005.

[17] M. Kanehisa and S. Goto, "Kegg: Kyoto encyclopedia of genes and genomes," *Nucleic Acids Res.*, vol. 28, no. 1, pp. 27–30, Jan. 2000.

[18] S. Kataria, P. Mitra, and S. Bhatia, "Utilizing context in generative Bayesian models for linked corpus," in *Proc. Assoc. Advancement Artif. Intell.*, 2010, pp. 1340–1345.

[19] K. Kutzkov, A. Bifet, F. Bonchi, and A. Gionis, "Strip: Stream learning of influence probabilities," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2013, pp. 275–283.

[20] L. Liu, J. Tang, J. Han, M. Jiang, and S. Yang, "Mining topic-level influence in heterogeneous networks," in *Proc. 19th ACM Int. Conf. Inf. Knowl. Manage.*, 2010, pp. 199–208.

[21] Y. Liu, H. Cao, Y. Hao, P. Han, and X. Zeng, "Discovering context-aware influential objects," in *Proc. SIAM Int. Conf. Data Mining*, 2012, pp. 780–791.

[22] G. Miao, Z. Guan, L. E. Moser, X. Yan, S. Tao, N. Anerousis, and J. Sun, "Latent association analysis of document pairs," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2012, pp. 1415–1423.

[23] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. Cambridge, MA, USA: MIT Press, 2012.

[24] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up?: Sentiment classification using machine learning techniques," in *Proc. ACL-02 Conf. Empirical Methods Natural Lang. Process.*, 2002, pp. 79–86.

[25] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design, Fourth Edition: The Hardware/Software Interface* (The Morgan Kaufmann Series in Computer Architecture and Design), 4th ed. San Mateo, CA, USA: Morgan Kaufman, 2008.

[26] X. H. Phan, M. L. Nguyen, and S. Horiguchi, "Learning to classify short and sparse text & web with hidden topics from large-scale data collections," in *Proc. 17th Int. Conf. World Wide Web*, 2008, pp. 91–100.

[27] C. R. Shalizi and A. C. Thomas, "Homophily and contagion are generically confounded in observational social network studies," *Sociological Methods Res.*, vol. 40, no. 2, pp. 211–239, May 2011.

[28] Y. Shen and R. Jin, "Learning personal + social latent factor model for social recommendation," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2012, pp. 1303–1311.

[29] G. V. Steeg and A. Galstyan, "Statistical tests for contagion in observational social network studies," in *Proc. 16th Int. Conf. Artif. Intell. Statist.*, Apr. 29 - May 1 2013, pp. 563–571.

[30] J. Tang, T. Lou, and J. M. Kleinberg, "Inferring social ties across heterogenous networks," in 5th ACM Int. Conf. Web Search Data Mining, 2012, pp. 743–752.

[31] J. Tang, Z. Meng, X. Nguyen, Q. Mei, and M. Zhang, "Understanding the limiting factors of topic modeling via posterior contraction analysis," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 190–198.

[32] J. Tang, J. Sun, C. Wang, and Z. Yang, "Social influence analysis in large-scale networks," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2009, pp. 807–816.

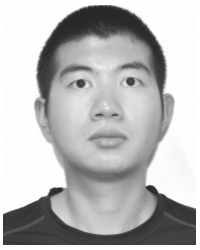
[33] J. Tang, S. Wu, J. Sun, and H. Su, "Cross-domain collaboration recommendation," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2012, pp. 1285–1293.

[34] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, "Arnetminer: Extraction and mining of academic social networks," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2008, pp. 990–998.

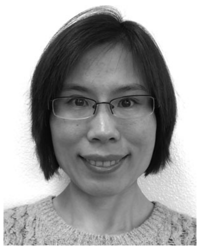
[35] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer, "Feature-rich part-of-speech tagging with a cyclic dependency network," in *Proc. Conf. North Am. Ch. Assoc. Comput. Linguistics Human Lang. Technol.*, 2003, pp. 173–180.

[36] C. Wang, J. Han, Y. Jia, J. Tang, D. Zhang, Y. Yu, and J. Guo, "Mining advisor-advisee relationships from research publication networks," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2010, pp. 203–212.

[37] F. Wang, Z. Wang, Z. Li, and J. Wen, "Concept-based short text classification and ranking," in *Proc. ACM Int. Conf. Inf. Knowl. Manage.*, 2014, pp. 1069–1078.



Chuan Hu received the bachelor's degree in management information systems from the Southwestern University of Finance and Economics, China, in 2012. He is currently working toward the PhD degree in computer science at New Mexico State University (NMSU). His research interests are graph mining and probabilistic models.



Huiping Cao received the PhD degree in computer science from the University of Hong Kong. She is an assistant professor in computer science at New Mexico State University (NMSU). Her research interests are in the general areas of data mining and data engineering. In particular, she is interested in creating effective and efficient methods for the discovery of useful knowledge from complex data (e.g., sequence data and graphs) either through pre-defined mining requests or through ad-hoc queries.