Improving Automated Bug Triaging with Specialized Topic Model: Technical Report

Xin Xia*, David Lo†, Ying Ding†, Jafar M. Al-Kofahi‡, Tien N. Nguyen‡, and Xinyu Wang*
*College of Computer Science and Technology, Zhejiang University, Hangzhou, China
†School of Information Systems, Singapore Management University, Singapore
‡Electrical and Computer Engineering Department, Iowa State University, Ames, USA
xxkidd@zju.edu.cn, {davidlo,ying.ding.2011}@smu.edu.sg, {jafar,tien}@iastate.edu, and wangxinyu@zju.edu.cn

I. Topic Extraction Using LDA

Here, we describe how we use LDA to extract topics from bug reports.

A. Modeling a Bug Report

Using LDA, all unique terms (i.e., words) in bug reports are collected into a common **vocabulary** Voc of size V. A **topic** k is expressed as a collection of terms from Voc. LDA uses a **topic-word vector** ϕ_k of size V to represent a topic k. Each element of the vector ϕ_k represents the probability of the corresponding term in Voc to describe the topic.

For a bug report m containing L_m terms, LDA considers it as a textual document with K technical aspects (i.e., topics). LDA would infer the values of the following two key parameters/variables for m:

1. Topic Assignment Vector z_m. Each term in m belongs to one topic. Thus, a topic assignment vector z_m is of length L_m, and each element of z_m is an index to one topic (i.e., 1 to K).

2. Topic Distribution Vector θ_m. A bug report m could have multiple topics, and different topics have different weights in describing m. Thus, LDA assigns a bug report m a topic distribution vector θ_m to represent the weights of the K topics. θ_m is of length K, and each element of θ_m represents the weight of the corresponding topic. We denote the weight of topic k in θ_m as $\theta_m[k]$, and the higher $\theta_m[k]$ is, the more terms in the bug report m are assigned to topic k.

B. Graphical Model and Generative Process

LDA can be represented as a graphical model, which is shown in Figure 1. A circle represents a variable in the graphical model and a rectangle represents a variable that repeats a certain number of times. The arrows represent dependencies between variables. K refers to the number of topics which need to be input by end users. M refers to the number of documents in the corpus. L_m refers to the number of words in the m^{th} document. The shaded circles are observed variables, which are the words in a bug report. The other circles refer to latent variables. $w_m[n]$ refers to the n^{th} word in the m^{th} document. $z_m[n]$ refers to the topic of the n^{th} word in the m^{th} document. ϕ_k refers to topic-word vector for each topic k – there are in total K such ϕ_k. θ_m is the topic distribution vector – there are in total M such θ_m. α and β are the parameters of the Dirichlet prior for the topic distribution vector, and topic-word vector, respectively [1].

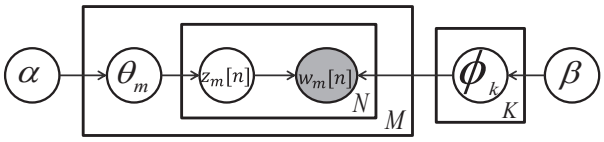

Fig. 1. The Graphical Model of LDA

LDA is a generative probabilistic model of a textual corpus. A generative probabilistic model assumes that the data (i.e., bug reports) is generated based on a certain process/model with the aforementioned sets of variables: θ_m, z_m, and ϕ_k, for each report m and each topic k. Given a bug report m of size L_m, LDA first generates its topic distribution vector θ_m according to a specific distribution (i.e., Dirichlet distribution [2]). Next, LDA generates the topic assignment vector z_m to describe the topic of each of the L_m positions in m according to its topic distribution vector θ_m. Finally, for each position in m, LDA generates a term (word) according to the topic k, which is assigned to the position, and the topic-word vector ϕ_k corresponding to topic k. In this way, a bug report is generated by LDA.

LDA works on two phases: training and inference. In the training phase, the terms in the training bug reports are used to learn the values of the sets of variables, θ_m, z_m, and ϕ_k, which best fit the training bug reports. In the inference phase, given a new bug report new, based on the values of the sets of variables that have been learned from training bug reports, LDA infers the topics that are assigned to terms in new (i.e., z_{new}), and the topic distribution vector of new (i.e., θ_{new}). In our framework, during the model construction phase, we employ the training step of LDA; during the recommendation phase, we employ the inference step of LDA.

C. Algorithms

Here we describe the training phase and prediction phase of LDA in detail.

Training Phase: In the training phase, we estimate the values of the variables: z_m (topic assignment vector), ϕ_k (topic-word vector), and θ_m (topic distribution vector), for each bug report m and each topic k, that best fit the bug reports in the training

data. Notice that LDA only observes the words in the bug reports; thus, the optimal estimated values of these variables will be the ones that have the largest posterior probability, conditioned on the observed data (i.e., words in bug reports). Gibbs sampling is one of the solutions to estimate z_m, ϕ and θ_m [2]. Gibbs sampling is a generic procedure used to infer values of variables of a statistical model. It consists of many iterations where the estimated values of the variables are refined progressively. In each iteration, the value of each variable is estimated, one at a time, conditioned on the values of the other variables. We describe how the values of LDA's sets of variables are inferred using Gibbs sampling in the following paragraphs.

Step 1. Estimating the topic assignment vector z_m for each bug report m in the training data.

Initially, each vector z_m of a bug report m, is assigned random values. Next, the algorithm iterates many times. In each iteration, it estimates every element of z_m based on the current values of the other elements of z_m and other vectors of other bug reports in the training data. The iteration process would terminate after a large number of iterations. In this work, following [1], we set the number of iterations to 500.

For each iteration, for each bug report m and each topic k, LDA estimates the probability of k being assigned to the i^{th} position of m (i.e., $z_m[i]$). This probability (i.e., $p(z_m[i] = k)$) is computed as follows:

$$p(z_m[i] = k) = \frac{(N_m^M[-i,k] + \alpha)}{(N_m^M - 1 + K\alpha)} \times \frac{(N_k^V[-i,w_i] + \beta)}{(N_k^V - 1 + V\beta)} \quad (1)$$

In the above equation, $N_m^M[-i,k]$ is the number of words (excluding the i^{th} word) in bug report m that are assigned to topic k; N_m^M is the number of words in bug report m; w_i is the i^{th} word of bug report m; $N_k^V[-i,w_i]$ is the number of times the word w_i (excluding its appearance in the i^{th} position of m) being assigned to topic k in all bug reports; N_k^V is the number of words assigned to topic k in all bug reports.

After the probability of each topic k is estimated using the above equation, the algorithm randomly chooses a topic, from the K topics, based on the estimated probabilities. The chosen topic is assigned as the topic of the i^{th} position of m. This assignment is refined in the subsequent iterations. At the end of this step, we have a topic assignment vector z_m for every bug report m in the training data.

Step 2. Estimating the topic distribution vector θ_m for each bug report m in the training data.

Once the topic assignment vector z_m of bug report m has been computed, considering K topics, we compute its topic distribution vector θ_m based on the topics assigned to its constituent words as:

$$\theta_m = \langle t_1, \ldots, t_K \rangle, \ where$$
$$t_i = \frac{\#\ words\ assigned\ the\ i^{th}\ topic\ in\ m}{\#\ words\ in\ m} \quad (2)$$

In this way, we map the words in the original bug reports into topics.

Step 3. Estimating topic-word vector ϕ_k for all K topics.

In the final step, the topic-word vector ϕ_k would be estimated for each topic k. We denote $\phi_k[i]$ as the probability of the term in the position i of *Voc* to represent topic k. $\phi_k[i]$ is estimated by computing the ratio between the number of times the i^{th} term of *Voc* is assigned to topic k, and the total number of times terms in *Voc* are assigned to topic k.

Prediction Phase: To infer the topic distribution of a new bug report new, we input its terms, make use of the values of the sets of variables (i.e., z_m, ϕ_k , and θ_m, for each bug report m and each topic k) estimated in the training phase, and employ Gibbs sampling to iterate through the terms in the new bug report enough number of times to infer their corresponding topics. At the end of prediction phase, we get the topic distribution vector θ_{new} of the new bug report which would be processed by *TopicMiner* (see Section **??**). In the next paragraphs, we first describe how z_{new} is estimated for a new bug report. Next, we describe how θ_{new} is estimated.

Step 1. Estimating the topic assignment vector z_{new} for a new bug report new.

Initially, we randomly assign topics to z_{new}. We then perform many (i.e., 500) iterations to refine the topics in z_{new}. For each iteration and each position i in z_{new}, given the topic assignments of bug reports in the training set and the current assignments of topics to words in the new bug reports, except for the i^{th} word, we can compute the probability of assigning topic k to the i^{th} word of the new bug report new as follows:

$$p(z_{new}[i] = k) = \frac{(N_{new}[-i,k] + \alpha)}{(N_{new} - 1 + K\alpha)} \times \frac{(N_k^V[-i,w_i] + \beta)}{(N_k^V - 1 + V\beta)} \quad (3)$$

In the above equation, $N_{new}[-i,k]$ is the number of words (excluding the current position i) in the new bug report new that are assigned to topic k; N_{new} is the number of words in the new bug report; w_i is the word at position i in bug report m; $N_k^V[-i,w_i]$ is the number of times the word w_i (excluding its appearance in the i^{th} position of m) is being assigned to topic k in all bug reports; N_k^V is the number of words in all bug reports which are assigned to topic k.

After the probability of each topic k is estimated using the above equation, the algorithm randomly chooses a topic, from the K topics, based on the estimated probabilities. The chosen topic is assigned as the topic of the i^{th} position of new. This assignment is refined in the subsequent iterations. At the end of this step, we have a topic assignment vector z_{new} for new.

Step 2. Estimating the topic distribution vector θ_{new} for the new bug report new.

To infer the topic distribution vector θ_{new}, we use Equation (2) to calculate θ_{new}, and the detailed step is the same as step 2 of the training phase.

II. Topic Extraction with MTM

Here, we describe our proposed topic model, named Multi-feature Topic Model (MTM), and how we use it to extract topics from bug reports.

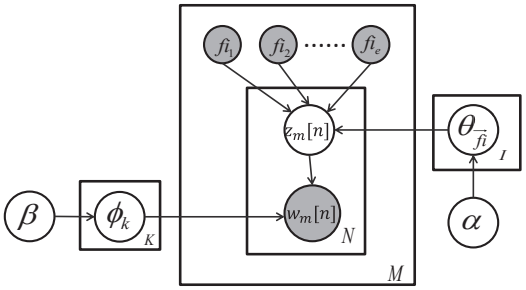

Fig. 2. The Graphical Model of Multi-feature Topic Model (MTM)

A. Modeling a Bug Report

LDA provides a way to estimate the topic distribution of a textual document. However, LDA is a general purpose topic model, which might not be very suitable for our bug triaging problem. Only words in the bug reports are observed and used as input to LDA. However, in addition to a textual description, a typical bug report contains many different fields (e.g., product and component). We refer to them as features of a bug report, and denote them as fi_1, fi_2, $\cdots$ fi_e. In this section, we present our *multi-feature topic model (MTM)* which considers these features to better capture topics in bug reports.

In MTM, the topic distribution vector θ_m of a bug report m would be affected by the features of m. We denote the features of a bug report m as $\overrightarrow{fi}^m$, i.e., $\overrightarrow{fi}^m = (fi_1^m, \cdots, fi_e^m)$. In this work, we only use an instance of *MTM* with two features: product and component – we set fi_1 to be the product, and fi_2 to be the component. A specific feature combination $\overrightarrow{fi}_i$ (e.g., fi_1 = `CDT` and fi_2 = `cdt-core`) could be associated to multiple topics, and each of the topics could be assigned a weight to describe the strength of its relationship with $\overrightarrow{fi}_i$. Thus, MTM assigns to a specific feature combination $\overrightarrow{fi}_i$ a **feature-topic vector** $\theta_{\overrightarrow{fi}_i}$ to represent the weights of all of the K topics. $\theta_{\overrightarrow{fi}_i}$ has the length of K, and each element of $\theta_{\overrightarrow{fi}_i}$ represents the weight of the corresponding topic at the element's position. We denote the value of topic k in $\theta_{\overrightarrow{fi}_i}$ as $\theta_{\overrightarrow{fi}_i}[k]$, and the higher $\theta_{\overrightarrow{fi}_i}[k]$ is, the more terms in bug reports, whose feature combination is $\overrightarrow{fi}_i$, are assigned topic k.

B. Graphical Model and Generative Process

The graphical model of *MTM* is shown in Figure 2. I refers to the number of different feature combinations. $\theta_{\overrightarrow{fi}_i}$ is the feature-topic vector – there are in total I such $\theta_{\overrightarrow{fi}_i}$. The structure is similar to LDA's graphical model with a few exceptions: first, we now have additional observed variables which are the features of bug reports such as the product and component of the reports. Also, rather than having a topic distribution per bug report θ_m, we now have a topic distribution per feature combination (denoted as $\theta_{\overrightarrow{fi}_i}$ in the graphical model).

Similar to LDA, MTM is also a generative probabilistic model of a textual corpus. The generative process of MTM uses 3 sets of variables feature-topic vector $\theta_{\overrightarrow{fi}_i}$, topic assignment vector z_m, and topic-word vector ϕ_k, for each feature combination $\overrightarrow{fi}_i$, each bug report m, and each topic k. Given a bug report m, and its feature combination $\overrightarrow{fi}^m$. MTM first generates its feature-topic vector $\theta_{\overrightarrow{fi}^m}$ according to the Dirichlet distribution. Next, MTM generates the topic assignment vector z_m to describe the topic of each position in m according to its feature-topic vector $\theta_{\overrightarrow{fi}^m}$. Finally, in each position of the bug report, MTM generates a term (word) according to the topic k assigned to this position, and the topic-word vector θ_k corresponding to topic k. In this way, a bug report is generated by leveraging MTM. Notice that in LDA, the topic assignment vector z_m is generated by the **topic distribution vector** θ_m, while in MTM, z_m is generated by the **feature-topic vector** $\theta_{\overrightarrow{fi}_i}$.

Similar to LDA, MTM works on two phases: training and inference. In the training phase, the terms and the features in the training bug reports are used to learn the values of the sets of variables, $\theta_{\overrightarrow{fi}}$, z_m, and ϕ_k, which best fit the training bug reports. Since *TopicMiner* takes as input the topic distribution vector θ_m, we also derive θ_m from z_m for each bug report m. In the inference phase, given a new bug report n, based on the values of the sets of variables that have been learned from training bug reports, MTM infers the topic assignment vector z_n and the topic distribution vector θ_n of n. In our framework, during the model construction phase, we employ the training step of MTM; during the recommendation phase, we employ the inference step of MTM.

C. Algorithms

Here we describe the training phase and prediction phase of MTM in detail.

Training Phase: In the training phase, we aim to estimate the values of the sets of variables: z_m, ϕ_k, and $\theta_{\overrightarrow{fi}}$ for each bug report m, each topic k and each feature combination $\overrightarrow{fi}$, that best fit bug reports in a training set. The optimal estimated values of these variables are the ones that have the largest posterior probability conditioned on the observed data (i.e., words and features of bug reports). Since the graphical model and generative process of MTM are different from those of LDA, to estimate the values of z_m, ϕ_k, and and $\theta_{\overrightarrow{fi}}$ using Gibbs sampling, we need to re-derive the equations to estimate these values. The detailed derivation steps are long and complicated are in Section III. In this paper, we simply present how the resultant formulas are used in the Gibbs sampling iterations. Also, since *TopicMiner* takes as input the topic distribution vector θ_m, for every bug report m, θ_m is also derived from the topic assignment vector z_m. We describe how the values of MTM's sets of variables are inferred using Gibbs sampling in the following paragraphs.

Step 1. Estimating the topic assignment vector z_m for each bug report m in the training data.

Initially, the variable z_m for each bug report m and ϕ_k for each topic k, and $\theta_{\overrightarrow{fi}}$ for each feature combination $\overrightarrow{fi}$ are all assigned with random values. Next, similar to LDA, the algorithm iterates many times. In each iteration, it estimates every element of z_m based on the current values of the other elements of z_m and other vectors of other bug reports in the training data. The iteration process would terminate after 500 iterations.

The major difference is that MTM estimates the probability of topic k being assigned to the i^{th} position of bug report m (i.e., $z_m[i]$) with feature combination f, in each iteration, using the following equation:

$$p(z_m[i] = k) = \frac{(N_f^F[-i,k] + \alpha)}{(N_f^F - 1 + K\alpha)} \times \frac{(N_k^V[-i,w_i] + \beta)}{(N_k^V - 1 + V\beta)} \quad (4)$$

In the above equation, $N_f^F[-i,k]$ is the number of words that are assigned to topic k in bug reports with feature combination f (excluding the i^{th} word of m); N_f^F is the number of words in bug reports with feature combination f; w_i is the i^{th} word of m; $N_k^V[-i,w_i]$ is the number of times the word w_i (excluding its appearance in the i^{th} position of m) is assigned to topic k; N_k^V is the number of words assigned to topic k. Notice that Equation 4 (for MTM) is different from Equation 1 (for LDA). For LDA, $N_m^M[-i,k]$ is used; while for MTM, $N_f^F[-i,k]$ is used. $N_f^F[-i,k]$ is used to take the feature combination into consideration when estimating the probability of a topic assignment.

Step 2. Estimating the feature-topic vector $\theta_{\overrightarrow{fi}}$ for each feature combination fi.

We first seperate bug reports into different groups according to their feature combinations. Bug reports are in the same group if their feature combinations are the same. Under each group corresponding to a feature combination fi, for a topic k, we denote $\theta_{\overrightarrow{fi}}[k]$ as the probability that topic k represents fi. $\theta_{\overrightarrow{fi}}[k]$ is approximately computed by computing the ratio between the number of terms which are assigned to topic k in bug reports with feature combination fi, and the total number of terms in bug reports with feature combination fi.

Step 3. Estimating topic-word vector ϕ_k for all K topics.

The step of estimating topic-word vector ϕ_k of MTM is the same as that of LDA. The detailed step is the same as step 3 of the training phase of LDA.

Step 4. Estimating the topic distribution vector θ_m for each bug report m in the training data.

To estimate the topic distribution vector θ_m, we use Equation (2), and the detailed step is the same as step 2 of the training phase of LDA.

Prediction Phase: To infer the topic distribution of a new bug report, we input its terms and multiple features, make use of the values of the sets of variables (i.e., z_m, ϕ_k , and $\theta_{\overrightarrow{fi}}$) estimated in the training phase, and employ Gibbs sampling to iterate through the terms in the new bug report enough number of times to get their corresponding topics. At the end of prediction phase, we get the topic distribution vector θ_{new} which would be processed by *TopicMiner*.

Step 1. Estimating the topic assignment vector z_{new} for a new bug report new.

The process is similar to that of LDA (Step 1 of the prediction phase of LDA), except to compute the probability of assigning topic k to position i of a new bug report with feature combination f, we make use of the following equation:

$$p(z_{new}[i] = k) = \frac{(N_f^{F+}[-i,k] + \alpha)}{(N_f^{F+} - 1 + K\alpha)} \times \frac{(N_k^V[i,w_i] + \beta)}{(N_k^V - 1 + V\beta)} \quad (5)$$

In the above equation, $N_f^{F+}[-i,k]$ is the number of words that are assigned to topic k in the new bug report (excluding its i^{th} word) and bug reports with feature combination f in the training set; N_f^{F+} is the number of words in the new bug report and bug reports with feature combination f in the training set; w_i is the i^{th} word of new; $N_k^V[-i,w_i]$ is the number of times the word w_i (excluding its appearance in the i^{th} position of new) is assigned topic k in all bug reports; N_k^V is the number of words which are assigned topic k in all bug reports.

Step 2. Estimating the topic distribution vector θ_{new} for the new bug report new.

To infer the topic distribution vector θ_{new}, we use Equation (2) to calculate θ_{new}, and the detailed step is the same as step 4 of the training phase of MTM.

III. Inference of MTM

In this section, we describe the detail steps to derive the Gibbs sampling function of MTM. We first describe the generative process of MTM by using more formal description. Next, we present the training phase of MTM, and the formula derivation. Finally, we present the inference of topics for a new bug report.

A. Generative Process

Our proposed *multi-feature topic model (MTM)* is a generative model that considers the features of a bug report. The general graphical model of *MTM* is shown in Figure 2. The structure is similar to LDA with a few exceptions: first, we now have additional observed variables which are the features of bug reports such as the products and components of the reports. Also, rather than having a topic distribution per document, we now have a topic distribution per feature combination (denoted as $\theta_{\overrightarrow{fi}_i}$ in the graphical model).

Various features (e.g., product, component, reporter, version, platform, etc.) can be considered in *MTM*. We denote the features as fi_1, fi_2, $\cdots$ fi_e. The generative process of *MTM* is as follows:

1) For a feature combination $\overrightarrow{fi} = (fi_1, fi_2, \cdots, fi_e)$ appearing in the corpus, sample the feature-topic distribution $\theta_{\overrightarrow{fi}}$ from **Dir**(α);
2) For each topic $t = 1, 2, \cdots, K$, sample the topic-word distribution ϕ_t from **Dir**(β);
3) For each document (i.e., bug report) $m = 1, 2, \cdots, M$, with feature combination $\overrightarrow{fi}^m$:
   For each word w_{mn} in document m:
   Sample a topic z_{mn} from **Mult**($\theta_{\overrightarrow{fi}^m}$);
   Sample a word w_{mn} from **Mult**($\phi_{z_{mn}}$).

B. Training Phase

Table I summarizes the notations used in the remaining part of this section. Our goal is to generate the topics (i.e., z_{mn}) for all the words in the M documents (i.e., bug reports). Gibbs sampling is often used to recover the topics given a textual corpus and a graphical model [2]. Gibbs sampling is a Markov Chain Monte Carlo method, which is widely used in parameter estimation. In Gibbs sampling, we perform many iterations. For each iteration, we perform many steps. At each step, we estimate the value of one latent variable (e.g., the topic of a word in a document (i.e., bug report)) while keeping the values of all other latent variables constant (e.g., the topics of all other words in the documents). The process continues until a certain number of iterations have been performed or the process converges (i.e., there is no more change to the values assigned to the latent variables).

At each step of Gibbs sampling, we need to calculate the probability of assigning topic k to the i^{th} position of bug report m (i.e., $z_m[i]$) based on the topic assignments of all other words and all other variables of the graphical model:

$$p(z_m[i] = k) = p(z_m[i] = k|\boldsymbol{Z}_{-mi}, \boldsymbol{W}, \boldsymbol{FI}, \alpha, \beta)$$

Here, $\boldsymbol{Z}$ denotes the topic assignment of all words, $\boldsymbol{Z}_{-mi}$ denotes the topic assignment of all words excluding the i^{th} position in document m, $\boldsymbol{W}$ denotes all the words in the bug report collections, and $\boldsymbol{FI}$ denotes the feature information collection of all reports (i.e., feature combinations of each bug report in the collection). We refer to this probability as the *Gibbs sampling function*. The derivation of the Gibbs sampling functions used in the training phase (i.e., Equation (17)) is as follows.

The Gibbs sampling function for the training phase can be calculated by Bayes theorem as follows:

$$\begin{aligned} & p(z_m[i] = k|\boldsymbol{Z}_{-mi}, \boldsymbol{W}, \boldsymbol{FI}, \alpha, \beta) \\ = & \frac{p(z_m[i] = k, \boldsymbol{Z}_{-mi}, \boldsymbol{W}|\boldsymbol{FI}, \alpha, \beta)}{p(\boldsymbol{Z}_{-i}, \boldsymbol{W}|\boldsymbol{FI}, \alpha, \beta)} \\ = & \frac{p(\boldsymbol{Z}, \boldsymbol{W}|\boldsymbol{FI}, \alpha, \beta)}{p(\boldsymbol{Z}_{-mi}, \boldsymbol{W}_{-mi}|\boldsymbol{FI}, \alpha, \beta)p(w_{mi}|\boldsymbol{Z}_{-mi}, \boldsymbol{W}_{-mi}, \boldsymbol{FI}, \alpha, \beta)} \\ \propto & \frac{p(\boldsymbol{Z}, \boldsymbol{W}|\boldsymbol{FI}, \alpha, \beta)}{p(\boldsymbol{Z}_{-mi}, \boldsymbol{W}_{-mi}|\boldsymbol{FI}, \alpha, \beta)} \end{aligned} \quad (6)$$

Since $p(w_{mi}|\boldsymbol{Z}_{-mi}, \boldsymbol{W}_{-mi}, \boldsymbol{FI}, \alpha, \beta)$ is not related to the value of $z_m[i]$, we simplify it. Based on our graphical model, the characteristic joint distribution of multi-feature topic model is:

$$p(\boldsymbol{Z}, \boldsymbol{W}|\alpha, \beta, \boldsymbol{FI}) = p(\boldsymbol{W}|\boldsymbol{Z}, \beta)p(\boldsymbol{Z}|\alpha, \boldsymbol{FI}) \quad (7)$$

The first term at the right part of the above equation is the same as the first term of the characteristic joint distribution of LDA (c.f. [2]), which is:

$$p(\boldsymbol{W}|\boldsymbol{Z}, \beta) = \prod_{k=1}^{K} \frac{\prod_{w=1}^{V} \Gamma(N_k^V[w] + \beta)}{\Gamma(N_k^V + \beta V)} \frac{\Gamma(\beta V)}{\Gamma(\beta)^V} \quad (8)$$

where $N_k^V[w]$ is the number of times word w is assigned to topic k, N_k^V is the number of words assigned to topic k and $\Gamma(z)$ is a function which can be calculated as

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt \quad (9)$$

and it also satisfies the functional equation:

$$\Gamma(z + 1) = z\Gamma(z) \quad (10)$$

Similarly, $p(\boldsymbol{Z}|\alpha, \boldsymbol{FI})$ is derived as follows,

$$\begin{aligned} p(\boldsymbol{Z}|\alpha, \boldsymbol{FI}) &= \int p(\boldsymbol{Z}, \theta|\alpha, \boldsymbol{FI}) \, d\theta \\ &= \int p(\boldsymbol{Z}|\theta, \boldsymbol{FI}) p(\theta|\alpha, \boldsymbol{FI}) \, d\theta \end{aligned} \quad (11)$$

and

$$\begin{aligned} p(\theta|\alpha, \boldsymbol{FI}) &= \prod_{\overrightarrow{fi}} p(\theta_{\overrightarrow{fi}}|\alpha) \\ &= \prod_{\overrightarrow{fi}} \frac{\Gamma(\alpha K)}{\Gamma(\alpha)^K} \times \prod_{k=1}^{K} \theta_{\overrightarrow{fi},k}^{\alpha-1} \end{aligned} \quad (12)$$

$$\begin{aligned} p(\boldsymbol{Z}|\theta, \boldsymbol{FI}) &= \prod_{m=1}^{M} \prod_{n=1}^{L_m} \prod_{k=1}^{K} \theta_{\overrightarrow{fi}^m,k}^{I(z_{mn}=k)} \\ &= \prod_{m=1}^{M} \prod_{k=1}^{K} \theta_{\overrightarrow{fi}^m,k}^{C(m,k)} \end{aligned} \quad (13)$$

where L_m is the length of m^{th} document and $I(\cdot)$ is an indicator function which returns 1 when the condition is true, $C(m, k)$ is document-topic count, i.e., the number of times topic k is assigned to the words in the m^{th} document. For the sake of similarity, we will use f to represent the feature combination $\overrightarrow{fi}$. Combining the equations (11), (12) and (13), we derive:

$$\begin{aligned} p(\boldsymbol{Z}|\alpha, \boldsymbol{FI}) &= \int \prod_f \frac{\Gamma(\alpha K)}{\Gamma(\alpha)^K} \prod_{k=1}^{K} \theta_{f,k}^{N_f^F[k]+\alpha-1} d\theta \\ &= \prod_f \frac{\prod_{k=1}^{K} \Gamma(N_f^F[k] + \alpha)}{\Gamma(N_f^F + \alpha K)} \frac{\Gamma(\alpha K)}{\Gamma(\alpha)^K} \end{aligned} \quad (14)$$

where $N_f^F[k]$ is the number of words that are assigned to topic k in bug reports with feature combination f, N_f^F is the number of words in bug reports with feature combination f.

By combining equations 7, 8, 14, we can get

$$\begin{aligned} p(\boldsymbol{Z}, \boldsymbol{W}|\alpha, \beta, \boldsymbol{FI}) = & \prod_{k=1}^{K} \frac{\prod_{w=1}^{V} \Gamma(N_k^V[w] + \beta)}{\Gamma(N_k^V + \beta V)} \frac{\Gamma(\beta V)}{\Gamma(\beta)^V} \times \\ & \prod_f \frac{\prod_{k=1}^{K} \Gamma(N_f^F[k] + \alpha)}{\Gamma(N_f^F + \alpha K)} \frac{\Gamma(\alpha K)}{\Gamma(\alpha)^K} \end{aligned} \quad (15)$$

TABLE I
SYMBOLS ASSOCIATED WITH MULTI-FEATURE TOPIC MODEL

Notation	Type	Description
M	scalar	Numbers of documents (i.e., bug reports) in the document collection.
K	scalar	Numbers of topics.
V	scalar	Number of unique terms in the documents.
e	scalar	Number of features.
α	scalar	Dirichlet prior, hyperparameter for the topic distribution for each feature combination.
β	scalar	Dirichlet prior, hyperparameter for the word distribution for each topic.
$\overrightarrow{fi}$	vector	Vector representation of a feature combination, i.e., $\overrightarrow{fi} \in E$, and $\overrightarrow{fi} = (fi_1, fi_2, \cdots fi_e)$
$\boldsymbol{Z}$	vector	Topic assignment of all words.
$\boldsymbol{FI}$	vector	Feature combinations of bug reports in the document collection.
$\boldsymbol{W}$	vector	All words in the bug reports in the document collection.
ϕ_k	vector	Word distribution for topic k.
$\theta_{\overrightarrow{fi}}$	vector	Topic distribution for feature combination $\overrightarrow{fi}$.
z_{mn}	scalar	Topic of the n^{th} word in the m^{th} bug report.
w_{mn}	scalar	n^{th} word in the m^{th} bug report.

Similarly, we can get the denominator of Equation (6):

$$p(\boldsymbol{Z}_{-mi}, \boldsymbol{W}_{-mi}|\alpha, \beta, \boldsymbol{FI}) = \prod_{k=1}^{K} \frac{\prod_{w=1}^{V} \Gamma(N_k^V[-i,w]+\beta)}{\Gamma(N_k^V - 1 + \beta V)} \frac{\Gamma(\beta V)}{\Gamma(\beta)^V} \times \prod_{f} \frac{\prod_{k=1}^{K} \Gamma(N_f^F[-i,k]+\alpha)}{\Gamma(N_f^F - 1 + \alpha K)} \frac{\Gamma(\alpha K)}{\Gamma(\alpha)^K} \quad (16)$$

where $N_f^F[-i,k]$ is the number of words that are assigned to topic k in bug reports with feature combination f (excluding the i^{th} word of m), $N_k^V[-i,w_i]$ is the number of times the word w_i (excluding its appearance in the i^th position of m) is assigned to topic k.

According to Equation 10, we can get the final Gibbs sampling function:

$$p(z_m[i] = k|\boldsymbol{Z}_{-mi}, \boldsymbol{W}, \boldsymbol{FI}, \alpha, \beta) \propto \frac{N_f^F[-i,k]+\alpha}{N_f^F - 1 + K\alpha} \times \frac{N_k^V[-i,w_i]+\beta}{N_k^V - 1 + V\beta} \quad (17)$$

The Gibbs sampling algorithm for the training phase of *MTM* is shown in Algorithm 1. In the algorithm, we have counting variables N_f^F, $N_f^F[k]$, N_k^V and $N_k^V[w]$. A topic is assigned to a feature combination if it is assigned to a word inside a document with that feature combination. In the initialization part (Lines 1 to 6), we first randomly assign a topic to each word in our corpus and update all the counting variables according to this random assignment. In the sampling part (Lines 7 to 13), we perform the following steps iteratively:

1) For each word w_{mn}, we assign a topic randomly to it according to the probability computed using Equation (17). (Line 10).

Algorithm 1 The Training Phase of MTM.

Input: Document collection $\boldsymbol{C}$, Feature information collection of all reports $\boldsymbol{FI}$, Hyperparameters α, β, Topic number K, Maximum number of iteration $maxIters$.
Output: Topic assignment for all of the words in $\boldsymbol{C}$: $\boldsymbol{Z}$
Initialization:
1: **for** $m \leftarrow 1, \cdots, M$ **do**
2: **for** $n \leftarrow 1, \cdots, L_m$ **do**
3: Sample topic z_{mn} from **Mult**$(1/K, \cdots, 1/K)$
4: **end for**
5: **end for**
6: Initialize all N_f^F, $N_f^F[k]$, N_k^V and $N_k^V[w]$
Sampling:
7: **repeat**
8: **for** $m \leftarrow 1, \cdots, M$ **do**
9: **for** $n \leftarrow 1, \cdots, L_m$ **do**
10: Randomly assign a topic to w_{mn} according to Equation (17)
11: **end for**
12: **end for**
13: Update all N_f^F, $N_f^F[k]$, N_k^V and $N_k^V[w]$
14: **until** Converge or iterated more than $maxIters$ steps

2) After going through all words in our corpus, we update all the counting variables (i.e., N_f^F, $N_f^F[k]$, N_k^V and $N_k^V[w]$) according to the new topic assignments (Line 13).
3) This process repeats many times until convergence or a predefined number of iterations has been reached (Line 14).

C. Inference Phase

In the inference phase, we infer the topic distribution of a new bug report m_{new}. We fix the topic assignments of words in the training bug reports, and use Gibbs sampling to iterate through the terms in the new bug report enough number of times to get their topic assignments. At each step of Gibbs sampling, we need to calculate the probability of assigning topic k to each word in m_{new}. In the inference part, we estimate the topic assignments of words in a new document. For a new document m_{new} with feature combination f, we

can estimate the topic assignment of its i^{th} word. The joint probability of words and topics for m_{new} is:

$$\begin{aligned} & p(\boldsymbol{Z}^{new}, \boldsymbol{W}^{new}, \boldsymbol{Z}, \boldsymbol{W} | \alpha, \beta, f, \boldsymbol{FI}) \\ = \; & p(\boldsymbol{W}^{new}, \boldsymbol{W} | \boldsymbol{Z}^{new}, \boldsymbol{Z}, \beta) p(\boldsymbol{Z}^{new}, \boldsymbol{Z} | \alpha, f, \boldsymbol{FI}) \end{aligned} \quad (18)$$

Similar to the derivation of the Gibbs sampling function for the training phrase (i.e., Equation (17)), we can easily derive the sampling function for the inference phrase, which is shown in Equation (19).

$$p(z_{new}[i] = k) = \frac{N_f^{F+}[-i, k] + \alpha}{N_f^{F+} - 1 + K\alpha} \times \frac{N_k^V[i, w_i] + \beta}{N_k^v - 1 + V\beta} \quad (19)$$

Algorithm 2 The Inference Phase of MTM.

Input: Words of training data $\boldsymbol{W}$, Topic assignments of words in training data $\boldsymbol{Z}$, Feature information collection of the training data $\boldsymbol{FI}$, Features of the new document f, Words in the new document (i.e., bug report) $\boldsymbol{W}^{new}$, Maximum number of iteration $maxIters$.
Output: Topic assignment of all the words in the new bug report: $\boldsymbol{Z}^{new}$
Initialization:
1: **for** $n \leftarrow 1, 2, \cdots, L_{new}$ **do**
2: Sample topic z_n from **Mult**$(1/K, \cdots, 1/K)$.
3: **end for**
4: Initialize counting variables
Sampling:
5: **repeat**
6: **for** $n \leftarrow 1, 2, \cdots, L_{new}$ **do**
7: Randomly assign a topic to n^{th} word according to Equation (19)
8: **end for**
9: Update counting variables
10: **until** Converge or iterated more than $maxIters$ steps

In the above equation, $N_f^{F+}[-i, k]$ is the number of words that are assigned to topic k in the new bug reports (excluding the i^{th} word) and bug reports with feature combination f in the training set; N_f^{F+} is the number of words in the new bug report and bug reports with feature combination f in the training set; w_i is the i^{th} word of m_{new}; $N_k^V[-i, w_i]$ is the number of times the word w_i (excluding its appearance in the i^{th} position of m_{new}) is assigned topic k in all bug reports; N_k^V is the number of words which are assigned topic k in all bug reports.

Algorithm 2 presents the inference phase of MTM by using Gibbs sampling. In the algorithm, we have counting variables(i.e., N_f^{F+}, $N_f^{F+}[k]$, N_k^V and $N_k^V[w]$) which are updated at each Gibbs sampling step. In the initialization part (Lines 1 to 4), we first randomly assign a topic to each word in the new bug report and update the counting variables according to this random assignment. In the sampling part (Lines 5 to 10), we perform the following steps iteratively:

1) For the n^{th} word, we assign a topic randomly to it according to the probability computed using Equation (19) (Line 7).
2) After going through all words in the new bug report, we update the counting variables(i.e., N_f^{F+}, $N_f^{F+}[k]$, N_k^V and $N_k^V[w]$) according to the new topic assignments (Line 9).
3) This process repeats many times until convergence or a predefined number of iterations has been reached (Line 10).

REFERENCES

[1] D. Blei, A. Ng, and M. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[2] G. Heinrich, "Parameter estimation for text analysis," *Web: http://www.arbylon.net/publications/text-est.pdf*, 2005.