Model Derivation

MATTHEW DENNY THURSDAY 13TH APRIL, 2017

Contents

1 Variable Definitions

To prevent confusion, we define all variables used in the main derivation below:

1.1 Latent Variables

1. $\Phi = \{\boldsymbol{\phi}^{(t)}\}_{t=1}^{T}$ – the topic-specific distributions over word types.
2. $\Theta = \{\boldsymbol{\theta}^{(d)}\}_{d=1}^{D}$ – the document-specific distributions over topics.
3. $\mathcal{Z} = \{\boldsymbol{z}^{(d)}\}_{d=1}^{D}$ – the latent topic assignments for each token.
4. $\mathcal{L} = \{l_t\}_{t=1}^{T}$ – the interaction pattern assignments for the topics.
5. $\mathcal{B} = \{b^{(c)}\}_{c=1}^{C}$ – the intercept terms for the interaction patterns.
6. $\Gamma = \{\boldsymbol{\gamma}^{(c)}\}_{c=1}^{C}$ – the coefficients for the interaction patters.
7. $\mathcal{S} = \{\{\boldsymbol{s}_a^{(c)}\}\}_{c=1}^{C}$ – the actor's latent positions for the interaction patterns.

1.2 Observed Variables

1. $\mathcal{W} = \{\boldsymbol{w}^{(d)}\}_{d=1}^{D}$ – the tokens.
2. $\mathcal{Y} = \{\boldsymbol{y}^{(d)}\}_{d=1}^{D}$ – the emails' recipients.

1.3 Metadata

1. $\mathcal{X} = \{\{\boldsymbol{x}^{(ar)}\}_{r=1}^{A}\}_{a=1}^{A}$ – the fixed covariates for the sender-recipient dyads.
2. $\mathcal{A} = \{a^{(d)}\}_{d=1}^{D}$ – These are the (fixed) sender identities for the emails.

1.4 Hyper-Parameters

1. α – the concentration parameter on the Dirichlet prior over document-topic distributions.
2. $\mathbf{m}$ – the base measure on the Dirichlet prior over document-topic distributions.
3. β – the concentration parameter on the Dirichlet prior over topic-word distributions.
4. $\mathbf{n}$ – the base measure on the Dirichlet prior over topic-word distributions.

2 Generative Process

In this section, we discuss the generative process for our model. We begin by describing the generative process we might use if we wanted to sample all data and latent variables from scratch just once, or make independent draws. From this first version of the generative process, it is most clear how we will need to go about deriving inference for this model. Next, we discuss a version of the generative process we use for backward sampling in Geweke's "Getting it Right" test. We discuss each in turn below.

2.1 Standard Generative Process

First we generate the global (corpus-wide) variables. There are two main sets of global variables: those that describe the topics people talk about and those that describe how people interact (interaction patterns). These variables are linked by a third set of variables that associate each topic with the pattern that best describes how people interact when talking about that topic. There are T topics. Each topic $\boldsymbol{\phi}^{(t)}$ is a discrete distribution over V word types **[See Algorithm 1]**. There are C interaction patterns. Each interaction pattern consists of an intercept $b^{(c)} \in \mathbb{R}$, a coefficient vector $\boldsymbol{\gamma}^{(c)} \in \mathbb{R}^P$, and a set of A positions $\{\boldsymbol{s}_a^{(c)} \in \mathbb{R}^K\}_{a=1}^{A}$—one for each person. We also associate each sender–recipient pair with an observed P-dimensional vector of covariates $\boldsymbol{x}^{(ar)}$; however, we assume that our generative process is conditioned on these covariates **[See Algorithm 2]**. The topics and interaction patterns are tied together via a set of T categorical variables (one per topic). These variables associate each topic with a single interaction pattern **[See Algorithm 3]**. Then, we generate the local variables. There are D emails. We assume that each email's sender $a^{(d)} \in [A]$ and length $N^{(d)} \in \mathbb{N}_0$ are observed; we do not generate these variables **[See Algorithm 4]**.

Algorithm 1: Topic Word Distributions

for $t = 1$ *to* T **do**
  draw $\boldsymbol{\phi}^{(t)} \sim \text{Dir}(\beta, \mathbf{n})$
end

Algorithm 2: Interaction Patterns

for $c = 1$ *to* C **do**
  draw $b^{(c)} \sim \mathcal{N}(\mu, \sigma_1^2)$
  draw $\boldsymbol{\gamma}^{(c)} \sim \mathcal{N}(\mathbf{0}, \sigma_2^2 \, \mathrm{I}_P)$
  for $a = 1$ *to* A **do**
    draw $\boldsymbol{s}_a^{(c)} \sim \mathcal{N}(\mathbf{0}, \sigma_2^2 \, \mathrm{I}_K)$
  end
  for $a = 1$ *to* A **do**
    for $r = 1$ *to* A **do**
      if $r \neq a$ **then**
        set $p_{ar}^{(c)} = \sigma(b^{(c)} + \boldsymbol{\gamma}^{(c)\top} \boldsymbol{x}^{(ar)} - ||\boldsymbol{s}_a^{(c)} - \boldsymbol{s}_r^{(c)}||)$
      end
      else
        set $p_{ar}^{(c)} = 0$
      end
    end
  end
end

Algorithm 3: Topic-Interaction Pattern Assignments

for $t = 1$ *to* T **do**
  draw $l_t \sim \text{Unif}(1, C)$
end

Algorithm 4: Tokens

for $d = 1$ *to* D **do**
  draw $\boldsymbol{\theta}^{(d)} \sim \text{Dir}(\alpha, \mathbf{m})$
  set $\bar{N}^{(d)} = \max(1, N^{(d)})$
  for $n = 1$ *to* $\bar{N}^{(d)}$ **do**
    draw $z_n^{(d)} \sim \boldsymbol{\theta}^{(d)}$
    if $N^{(d)} \neq 0$ **then**
      draw $w_n^{(d)} \sim \boldsymbol{\phi}^{(z_n^{(d)})}$
    end
  end
  for $t = 1$ *to* T **do**
    set $\bar{N}^{(t|d)} = \sum_{n=1}^{\bar{N}^{(d)}} \delta(z_n^{(d)}{=}t)$
  end
  for $r = 1$ *to* A **do**
    draw $y_r^{(d)} \sim \text{Bern}(\sum_{t=1}^{T} \frac{\bar{N}^{(t|d)}}{\bar{N}^{(d)}} p_{a^{(d)}r}^{(l_t)})$
  end
end

2.2 Backward Generative Process (For "Getting it Right")

We need to define a "backward" generative process in order to perform Geweke's "Getting it Right" test because when we are generating our "backwards" samples, we only want to resample the token word types and message recipients, and not any of our latent variables. This means we will take as input the latent variables we got by running our inference procedure (token

topic assignments, topic interaction pattern assignments, and interaction pattern parameters), and simply condition on these to draw new data (message recipients and token word types).

This "backward" version of the generative process is detailed below **[See Algorithm 5]**. Let NTV be a $V \times T$ dimensional matrix where each entry will record the count of the number of tokens of word-type v that are currently assigned to topic t. Also let NT be a T dimensional vector recording the total count of tokens currently assigned to topic t. These data structures, along with the latent variables we got by running our inference procedure (token topic assignments, topic interaction pattern assignments, and interaction pattern parameters) are passed in to the backward generative process, which then outputs new message recipients and token word types.

Algorithm 5: Generate data with backward sampling.

for $d = 1$ *to* D **do**
  set $\bar{N}^{(d)} = \max(1, N^{(d)})$
  for $n = 1$ *to* $\bar{N}^{(d)}$ **do**
    if $N^{(d)} \neq 0$ **then**
      $NTV_{w_n^{(d)}, z_n^{(d)}}$ -= 1
      for $v = 1$ *to* V **do**
        $\texttt{token\_word\_type\_distribution}_n^{(d)}[\text{v}] = \frac{(NTV_{v, z_n^{(d)}} + \beta \mathbf{n}[v_n])}{(NT_{z_n^{(d)}} + \beta)}$
      end
      draw $w_n^{(d)} \sim (\texttt{token\_word\_type\_distribution}_n^{(d)})$
      $NTV_{w_n^{(d)}, z_n^{(d)}}$ += 1
    end
  end
  for $r = 1$ *to* A **do**
    draw $y_r^{(d)} \sim \text{Bern}(\sum_{t=1}^{T} \frac{\bar{N}^{(t|d)}}{\bar{N}^{(d)}} p_{a^{(d)}r}^{(l_t)})$
  end
end

3 Factoring the Joint Distribution and Integrating Out Φ and Θ

We start by writing down the big joint distribution.

$$P(\Phi, \Theta, \mathcal{Z}, \mathcal{W}, \mathcal{L}, \mathcal{B}, \Gamma, \mathcal{S}, \mathcal{Y} \,|\, \mathcal{X}, \mathcal{A}, \beta, \mathbf{n}, \alpha, \mathbf{m}) \tag{1}$$

Our generative process implies a particular way to factor this joint distribution as a product of conditional and unconditional probabilities involving **observed variables**: $\mathcal{W}$ and $\mathcal{Y}$, **latent variables**: $\Phi, \Theta, \mathcal{Z}, \mathcal{L}, \mathcal{B}, \Gamma$, and $\mathcal{S}$, **metadata**: $\mathcal{X}$, and $\mathcal{A}$, and **hyper-parameters** $\beta, \mathbf{n}, \alpha$, and $\mathbf{m}$.

$$\begin{aligned} &P(\Phi, \Theta, \mathcal{Z}, \mathcal{W}, \mathcal{L}, \mathcal{B}, \Gamma, \mathcal{S}, \mathcal{Y} \,|\, \mathcal{X}, \mathcal{A}, \beta, \mathbf{n}, \alpha, \mathbf{m}) \\ &= P(\mathcal{W} \,|\, \mathcal{Z}, \Phi) P(\mathcal{Y} \,|\, \mathcal{Z}, \mathcal{X}, \mathcal{L}, \mathcal{B}, \Gamma, \mathcal{S}, \mathcal{A}) P(\mathcal{Z} \,|\, \Theta) P(\Phi \,|\, \beta, \mathbf{n}) P(\Theta \,|\, \alpha, \mathbf{m}) P(\mathcal{L}) P(\mathcal{B}) P(\Gamma) P(\mathcal{S}) \end{aligned} \tag{2}$$

Previous work (Griffiths, 2002) has shown that we can integrate out Φ and Θ in latent Dirichlet allocation so that we only have to perform inference over the latent topic-token assignments (using collapsed Gibbs sampling). Here we are going to do the same thing for our new model. Fortunately, this will involve steps that are identical to the process for LDA because we can ignore all terms that do not include Φ or Θ in performing this integration. First I am going to start with the joint (with the terms not involving Φ or Θ dropped for convenience) and expand it into distributions just so see what is going on:

$$\begin{aligned} &= P(\mathcal{Z}, \mathcal{W}, \Phi, \Theta \,|\, \alpha, \mathbf{m}, \beta, \mathbf{n}) \\ &= P(\mathcal{Z} \,|\, \Theta) P(\mathcal{W} \,|\, \mathcal{Z}, \Phi) P(\Phi \,|\, \beta, \mathbf{n}) P(\Theta \,|\, \alpha, \mathbf{m}) \end{aligned} \tag{3}$$

$$= \prod_{d=1}^{D} \prod_{n=1}^{N^{(d)}} P(z_n^{(d)} \,|\, \theta^{(d)}) \prod_{d=1}^{D} \prod_{n=1}^{N^{(d)}} P(w_n^{(d)} \,|\, \phi_{z_n^{(d)}}) \prod_{t=1}^{T} P(\phi_t \,|\, \beta, \mathbf{n}) \prod_{d=1}^{D} P(\theta^{(d)} \,|\, \alpha, \mathbf{m}) \tag{4}$$

$$= \prod_{d=1}^{D} \prod_{n=1}^{N^{(d)}} \text{Disc. Mult.}(z_n^{(d)} \,|\, \theta^{(d)}) \prod_{d=1}^{D} \prod_{n=1}^{N^{(d)}} \text{Disc. Mult.}(w_n^{(d)} \,|\, \phi_{z_n^{(d)}}) \prod_{t=1}^{T} \text{Dirichlet}(\phi_t \,|\, \beta, \mathbf{n}) \prod_{d=1}^{D} \text{Dirichlet}(\theta^{(d)} \,|\, \alpha, \mathbf{m}) \tag{5}$$

Fortunately this ends up being a big product of standard distributions. The goal is to work with this particular factorization implied by the generative process and integrate out Φ and Θ. We can start by writing out the integration at a high level:

$$\int_{\Phi}\int_{\Theta} P(\mathcal{Z}\,|\,\Theta)P(\mathcal{W}\,|\,\mathcal{Z},\Phi)P(\Phi\,|\,\beta,\mathbf{n})P(\Theta\,|\,\alpha,\mathbf{m})d\Phi d\Theta \tag{6}$$

Since some of the terms only depend on Φ and others only depend on Θ, we know we can split the integral as follows:

$$= \int_{\Phi} P(\mathcal{W}\,|\,\mathcal{Z},\Phi)P(\Phi\,|\,\beta,\mathbf{n})d\Phi \times \int_{\Theta} P(\mathcal{Z}\,|\,\Theta)P(\Theta\,|\,\alpha,\mathbf{m})d\Theta \tag{7}$$

Now we can rewrite these probabilities as a product over topic/document level probabilities:

$$= \int_{\Phi} \prod_{t=1}^{T} P(\phi_t\,|\,\beta,\mathbf{n}) \prod_{d=1}^{D}\prod_{j=1}^{N^{(d)}} P(w_n^{(d)}\,|\,\phi_{z_n^{(d)}})d\Phi \times \int_{\Theta} \prod_{d=1}^{D} P(\mathbf{z}^{(d)}\,|\,\theta^{(d)})P(\theta^{(d)}\,|\,\alpha,\mathbf{m})d\Theta \tag{8}$$

For the case of Φ, all of the terms involving $t' \neq t$ will be constant when we are evaluating the t'th term in the product, meaning we can push the integral inside the product. Similarly, for the case of Θ, all of the terms involving $d' \neq d$ will be constant when we are evaluating the d'th term in the product, so we can push the integral inside this product as well. This works out because if we look at the generative process we see that each ϕ_t is generated independently, as is each $\theta^{(d)}$.

$$= \prod_{t=1}^{T}\int_{\phi_t} P(\phi_t\,|\,\beta,\mathbf{n}) \prod_{d=1}^{D}\prod_{j=1}^{N_d} P(w_j^{(d)}\,|\,\phi_{z_j^{(d)}})d\phi_t \times \prod_{d=1}^{D}\int_{\theta^{(d)}} P(\mathbf{z}^{(d)}\,|\,\theta^{(d)})P(\theta^{(d)}\,|\,\alpha,\mathbf{m})d\theta^{(d)} \tag{9}$$

From here, we can substitute in the actual functional forms of these distributions. For notational simplicity, I will represent the multinomial likelihood $P(w_j^{(d)}\,|\,\phi_{z_j^{(d)}})$ as $\prod_{j=1}^{N_d}\phi_{z_j^{(d)},w_j^{(d)}}$ and the multinomial likelihood $P(\mathbf{z}^{(d)}\,|\,\theta^{(d)})$ as $\prod_{j=1}^{N_d}\theta_{z_j}^{(d)}$. Furthermore, we use V to denote the number of words in the vocabulary, and we end up with:

$$= \prod_{t=1}^{T}\int_{\phi_t} \frac{\Gamma\left(\sum_{i=1}^{V}\beta n_i\right)}{\prod_{i=1}^{V}\Gamma(\beta n_i)} \prod_{i=1}^{V}(\phi_{i,t})^{(\beta n_i)-1} \prod_{d=1}^{D}\prod_{j=1}^{N_d}\phi_{z_j^{(d)},w_j^{(d)}}d\phi_t \times \prod_{d=1}^{D}\int_{\theta^{(d)}} \frac{\Gamma\left(\sum_{t=1}^{T}\alpha m_t\right)}{\prod_{t=1}^{T}\Gamma(\alpha m_t)} \prod_{t=1}^{T}\left(\theta_t^{(d)}\right)^{(\alpha m_t)-1} \prod_{j=1}^{N_d}\theta_{z_j}^{(d)}d\theta^{(d)} \tag{10}$$

Next, we can notice that the fractions of Gamma functions do not involve Φ or Θ respectively, so we can move them outside of the integral.

$$= \prod_{t=1}^{T} \frac{\Gamma\left(\sum_{v=1}^{V}\beta n_v\right)}{\prod_{v=1}^{V}\Gamma(\beta n_v)} \int_{\phi_t} \prod_{v=1}^{V}(\phi_{v,t})^{(\beta n_v)-1} \prod_{d=1}^{D}\prod_{j=1}^{N_d}\phi_{z_j^{(d)},w_j^{(d)}}d\phi_t \times \prod_{d=1}^{D} \frac{\Gamma\left(\sum_{t=1}^{T}\alpha m_t\right)}{\prod_{t=1}^{T}\Gamma(\alpha m_t)} \int_{\theta^{(d)}} \prod_{t=1}^{T}\left(\theta_t^{(d)}\right)^{(\alpha m_t)-1} \prod_{j=1}^{N_d}\theta_{z_j}^{(d)}d\theta^{(d)} \tag{11}$$

Now the goal is to get rid of the integrals. We will want to collect terms to do this, and we can make use of the product rule for exponents ($a^n a^m = a^{n+m}$) if we adopt an alternate functional form for our multinomial likelihoods. First, it will be helpful to define so more terms which will make doing so easier. Lets define $N_{v,t}^{(d)}$ as the number of tokens assigned to topic t and word type v in document d:

$$N_{v,t}^{(d)} = \sum_{j=1}^{N_d} \mathbb{1}\left(z_j^{(d)} = t \;\&\; w_j^{(d)} = v\right) \tag{12}$$

where $\mathbb{1}$ is the indicator function. Then we can also define the following quantities which will be useful later:

$$N_{*,t}^{(d)} = \sum_{v=1}^{V} N_{v,t}^{(d)} \qquad N_{v,t}^{*} = \sum_{d=1}^{D} N_{v,t}^{(d)} \qquad N_{*,t}^{*} = \sum_{v=1}^{V}\sum_{d=1}^{D} N_{v,t}^{(d)} \tag{13}$$

We can rewrite the innermost products over N_d as products over T by making careful use of these terms as exponents:

$$= \prod_{t=1}^{T} \frac{\Gamma\left(\sum_{v=1}^{V}\beta n_v\right)}{\prod_{v=1}^{V}\Gamma(\beta n_v)} \int_{\phi_t} \prod_{v=1}^{V}(\phi_{v,t})^{(\beta n_v)-1} \prod_{v=1}^{V}\phi_{v,t}^{N_{v,t}^{*}}d\phi_t \times \prod_{d=1}^{D} \frac{\Gamma\left(\sum_{t=1}^{T}\alpha m_t\right)}{\prod_{t=1}^{T}\Gamma(\alpha m_t)} \int_{\theta^{(d)}} \prod_{t=1}^{T}\left(\theta_t^{(d)}\right)^{(\alpha m_t)-1} \prod_{t=1}^{T}\left(\theta_t^{(d)}\right)^{N_{*,t}^{(d)}}d\theta^{(d)} \tag{14}$$

Here, $\prod_{v=1}^{V} \phi_{v,t}^{N^*_{v,t}}$ is the product over all word types of probability of word type v being drawn from ϕ_t (the distribution over word types for topic t), and this is exponentiated to the count of the number of tokens assigned to topic t and word type v in all documents. This is equivalent to $\prod_{d=1}^{D}\prod_{j=1}^{N_d} \phi_{z_j^{(d)},w_j^{(d)}}$ in equation 11. Similarly, $\prod_{t=1}^{T}\left(\theta_t^{(d)}\right)^{N^{(d)}_{*,t}}$ is the product over all topics of the probability of topic t being drawn from $\theta^{(d)}$ (the document-specific distribution over topics) raised to the power of the number of tokens assigned to topic t in document d. This is equivalent to $\prod_{j=1}^{N_d}\theta_{z_j}^{(d)}$ in equation 11.

Now that we have products of the same variables over the same number of dimensions, we can again combine exponents as follows:

$$= \prod_{t=1}^{T}\frac{\Gamma\left(\sum_{v=1}^{V}\beta n_v\right)}{\prod_{v=1}^{V}\Gamma(\beta n_v)}\int_{\phi_t}\prod_{v=1}^{V}\left(\phi_{v,t}\right)^{(\beta n_v)+N^*_{v,t}-1}d\phi_t \times \prod_{d=1}^{D}\frac{\Gamma\left(\sum_{t=1}^{T}\alpha m_t\right)}{\prod_{t=1}^{T}\Gamma(\alpha m_t)}\int_{\theta^{(d)}}\prod_{t=1}^{T}\left(\theta_t^{(d)}\right)^{(\alpha m_t)+N^{(d)}_{*,t}-1}d\theta^{(d)} \tag{15}$$

We can see that we have part of a Dirichlet PDF, so now we just need to complete it by using a "multiply by one" trick, making sure to select the right thing to multiply by.

$$= \prod_{t=1}^{T}\frac{\Gamma\left(\sum_{v=1}^{V}\beta n_v\right)}{\prod_{v=1}^{V}\Gamma(\beta n_v)}\frac{\prod_{v=1}^{V}\Gamma(\beta n_v + N^*_{v,t})}{\Gamma\left(\sum_{v=1}^{V}\beta n_v + N^*_{v,t}\right)}\frac{\Gamma\left(\sum_{v=1}^{V}\beta n_v + N^*_{v,t}\right)}{\prod_{v=1}^{V}\Gamma(\beta n_v + N^*_{v,t})}\int_{\phi_t}\prod_{v=1}^{V}\left(\phi_{v,t}\right)^{(\beta n_v)+N^*_{v,t}-1}d\phi_t \times$$
$$\prod_{d=1}^{D}\frac{\Gamma\left(\sum_{t=1}^{T}\alpha m_t\right)}{\prod_{t=1}^{T}\Gamma(\alpha m_t)}\frac{\prod_{t=1}^{T}\Gamma(\alpha m_t + N^{(d)}_{*,t})}{\Gamma\left(\sum_{t=1}^{T}\alpha m_t + N^{(d)}_{*,t}\right)}\frac{\Gamma\left(\sum_{t=1}^{T}\alpha m_t + N^{(d)}_{*,t}\right)}{\prod_{t=1}^{T}\Gamma(\alpha m_t + N^{(d)}_{*,t})}\int_{\theta^{(d)}}\prod_{t=1}^{T}\left(\theta_t^{(d)}\right)^{(\alpha m_t)+N^{(d)}_{*,t}-1}d\theta^{(d)} \tag{16}$$

Now we move the nice half of each of these "multiply by one's" inside the integrals to complete them so that they integrate to one by the definition of the Dirichlet distribution.

$$= \prod_{t=1}^{T}\frac{\Gamma\left(\sum_{v=1}^{V}\beta n_v\right)}{\prod_{v=1}^{V}\Gamma(\beta n_v)}\frac{\prod_{v=1}^{V}\Gamma(\beta n_v + N^*_{v,t})}{\Gamma\left(\sum_{v=1}^{V}\beta n_v + N^*_{v,t}\right)}\int_{\phi_t}\frac{\Gamma\left(\sum_{v=1}^{V}\beta n_v + N^*_{v,t}\right)}{\prod_{v=1}^{V}\Gamma(\beta n_v + N^*_{v,t})}\prod_{v=1}^{V}\left(\phi_{v,t}\right)^{(\beta n_v)+N^*_{v,t}-1}d\phi_t \times$$
$$\prod_{d=1}^{D}\frac{\Gamma\left(\sum_{t=1}^{T}\alpha m_t\right)}{\prod_{t=1}^{T}\Gamma(\alpha m_t)}\frac{\prod_{t=1}^{T}\Gamma(\alpha m_t + N^{(d)}_{*,t})}{\Gamma\left(\sum_{t=1}^{T}\alpha m_t + N^{(d)}_{*,t}\right)}\int_{\theta^{(d)}}\frac{\Gamma\left(\sum_{t=1}^{T}\alpha m_t + N^{(d)}_{*,t}\right)}{\prod_{t=1}^{T}\Gamma(\alpha m_t + N^{(d)}_{*,t})}\prod_{t=1}^{T}\left(\theta_t^{(d)}\right)^{(\alpha m_t)+N^{(d)}_{*,t}-1}d\theta^{(d)} \tag{17}$$
$$= \prod_{t=1}^{T}\frac{\Gamma\left(\sum_{v=1}^{V}\beta n_v\right)}{\prod_{v=1}^{V}\Gamma(\beta n_v)}\frac{\prod_{v=1}^{V}\Gamma(\beta n_v + N^*_{v,t})}{\Gamma\left(\sum_{v=1}^{V}\beta n_v + N^*_{v,t}\right)} \times \prod_{d=1}^{D}\frac{\Gamma\left(\sum_{t=1}^{T}\alpha m_t\right)}{\prod_{t=1}^{T}\Gamma(\alpha m_t)}\frac{\prod_{t=1}^{T}\Gamma(\alpha m_t + N^{(d)}_{*,t})}{\Gamma\left(\sum_{t=1}^{T}\alpha m_t + N^{(d)}_{*,t}\right)} \tag{18}$$

At this point we are very satisfied with ourselves and the Dirichlet distribution. To summarize, what we have shown is that we can integrate out Φ and Θ so that we have:

$$\begin{aligned}&\int_{\Phi}\int_{\Theta} P(\mathcal{W}\,|\,\mathcal{Z},\Phi)P(\mathcal{Y}\,|\,\mathcal{Z},\mathcal{X},\mathcal{L},\mathcal{B},\Gamma,\mathcal{S},\mathcal{A})P(\mathcal{Z}\,|\,\Theta)P(\Phi\,|\,\beta,\mathbf{n})P(\Theta\,|\,\alpha,\mathbf{m})P(\mathcal{L})P(\mathcal{B})P(\Gamma)P(\mathcal{S})\\ &= P(\mathcal{Z}\,|\,\beta,\mathbf{n},\alpha,\mathbf{m})P(\mathcal{W}\,|\,\mathcal{Z})P(\mathcal{Y}\,|\,\mathcal{Z},\mathcal{X},\mathcal{L},\mathcal{B},\Gamma,\mathcal{S},\mathcal{A})P(\mathcal{L})P(\mathcal{B})P(\Gamma)P(\mathcal{S})\end{aligned} \tag{19}$$

which completes the demonstration.

4 Deriving the Collapsed Gibbs Sampler for LDA

4.1 A Standard Approach

What we want to compute to perform collapsed Gibbs sampling in LDA is the probability that a token is assigned to a particular topic conditional on the topic assignments of all other tokens, the word-types of all other tokens, and hyper-parameters.

$$P(z_n^{(d)} = t\,|\,\mathcal{W},\mathcal{Z}_{\setminus d,n},\beta,\mathbf{n},\alpha,\mathbf{m}) \tag{20}$$

We can beging by noting that this is equivalent to:

$$P(z_n^{(d)} = t\,|\,w_n^{(d)},\mathcal{W}_{\setminus d,n},\mathcal{Z}_{\setminus d,n},\beta,\mathbf{n},\alpha,\mathbf{m}) \tag{21}$$

We can beging by applying Bayes rule, which yields

$$P(z_n^{(d)} = t \,|\, w_n^{(d)}, \mathcal{W}_{\setminus d,n}, \mathcal{Z}_{\setminus d,n}, \beta, \mathbf{n}, \alpha, \mathbf{m})$$

$$= \frac{P(w_n^{(d)} = v \,|\, \mathcal{W}_{\setminus d,n}, z_n^{(d)}, \mathcal{Z}_{\setminus d,n}, \beta, \mathbf{n}, \alpha, \mathbf{m}) \times P(z_n^{(d)} = t \,|\, \mathcal{W}_{\setminus d,n}, \mathcal{Z}_{\setminus d,n}, \beta, \mathbf{n}, \alpha, \mathbf{m})}{\sum_{t=1}^{T} P(w_n^{(d)} = v \,|\, \mathcal{W}_{\setminus d,n}, z_n^{(d)} = t, \mathcal{Z}_{\setminus d,n}, \beta, \mathbf{n}, \alpha, \mathbf{m})} \tag{22}$$

$$\propto P(w_n^{(d)} = v \,|\, \mathcal{W}_{\setminus d,n}, \mathcal{Z}, \beta, \mathbf{n}, \alpha, \mathbf{m}) \times P(z_n^{(d)} = t \,|\, \mathcal{W}_{\setminus d,n}, \mathcal{Z}_{\setminus d,n}, \beta, \mathbf{n}, \alpha, \mathbf{m}) \tag{23}$$

$$= P(z_n^{(d)} = t, w_n^{(d)} \,|\, \mathcal{W}_{\setminus d,n}, \mathcal{Z}_{\setminus d,n}, \beta, \mathbf{n}, \alpha, \mathbf{m}) \qquad \text{by the chain rule} \tag{24}$$

Importantly the normalizing constant is tractable in this case, as it is just:

$$\sum_{t=1}^{T} P(z_n^{(d)} = t, w_n^{(d)} \,|\, \mathcal{W}_{\setminus d,n}, \mathcal{Z}_{\setminus d,n}, \beta, \mathbf{n}, \alpha, \mathbf{m}) \tag{25}$$

Next we note that $z_n^{(d)}$ only depends on $\mathcal{W}$ via $w_n^{(d)}$, and since we are no longer conditioning on it in:

$$P(z_n^{(d)} = t \,|\, \mathcal{W}_{\setminus d,n}, \mathcal{Z}_{\setminus d,n}, \beta, \mathbf{n}, \alpha, \mathbf{m}) \tag{26}$$

the conditional probability reduces to:

$$P(z_n^{(d)} = t \,|\, \mathcal{Z}_{\setminus d,n}, \alpha, \mathbf{m}) \tag{27}$$

Thus we have:

$$P(z_n^{(d)} = t \,|\, \mathcal{W}, \mathcal{Z}_{\setminus d,n}, \beta, \mathbf{n}, \alpha, \mathbf{m}) \propto P(w_n^{(d)} = v \,|\, \mathcal{W}_{\setminus d,n}, \mathcal{Z}, \beta, \mathbf{n}, \alpha, \mathbf{m}) \times P(z_n^{(d)} = t \,|\, \mathcal{Z}_{\setminus d,n}, \alpha, \mathbf{m}) \tag{28}$$

Now we just need to compute the conditional probabilities on the right hand side. To do this, we will use Dirichlet-multinomial conjugacy and the Dirichlet/multinomial predictive likelihood to arrive at an expression for these conditional probabilities.

4.1.1 Dirichlet/multinomial predictive likelihood

Each of $z_n^{(d)}$, $w_n^{(d)}$ are discrete, categorical random variables. In the general case, let x_i be a discrete random variable that can take on one of K values, which follows a multinomial distribution $\pi = \{\pi_1, ..., \pi_K\}$. Furthermore, as in our model, we give this multinomial distribution a Dirichlet prior, $\pi \sim \text{Dirichlet}(\alpha, \mathbf{m})$. After we have observed N data points $\{x_i\}_{i=1}^N$ then via Dirichlet-multinomial conjugacy, the posterior of π is also distributed Dirichlet. Let:

$$c_k = \sum_{i=1}^{N} \mathbb{1}(x_i = k) \tag{29}$$

then the posterior takes the form:

$$\pi \,|\, \alpha, \{x_i\}_{i=1}^N \sim \text{Dirichlet}(\alpha m_1 + c_1, ...\alpha m_K + c_K) \tag{30}$$

Thus the distribution of a new observation x_{N+1} given $\{x_i\}_{i=1}^N$ is then:

$$p(x_{N+1} = k \,|\, \alpha, \{x_i\}_{i=1}^N) = E(\pi_k \,|\, \alpha, \sum_{i=1}^{O} x_i) = \frac{\alpha m_k + c_k}{\sum_{j=1}^{K} \alpha m_j c_j} \tag{31}$$

4.1.2 Finishing the Derivation

The key observation that will let us use the Dirichlet/multinomial predictive likelihood to get the expressions for our conditional probabilities is to note that we can treat each $z_n^{(d)}$, $w_n^{(d)}$ as the new observations given all the rest of their values. So then the conditional probability $P(z_n^{(d)} = t \,|\, \mathcal{Z}_{\setminus d,n}, \alpha, \mathbf{m})$, we have:

$$P(z_n^{(d)} = t \,|\, \mathcal{Z}_{\setminus d,n}, \alpha, \mathbf{m}) = \frac{N_{\setminus d,n}^{(t|d)} + \alpha m_t}{\sum_{t'=1}^{T} N_{\setminus d,n}^{(t'|d)} + \alpha m_t'} = \frac{N_{\setminus d,n}^{(t|d)} + \alpha m_t}{N^{(d)} - 1 + \alpha} \tag{32}$$

where $N_{\setminus d,n}^{(t|d)}$ is equivalent to c_k, and is the number of tokens assigned to topic t in the current document, excluding the current position, and $N^{(d)}$ is the number of tokens in the document. Now for the conditional probability $P(w_n^{(d)} =$

$v \,|\, \mathcal{W}_{\setminus d,n}, \mathcal{Z}, \beta, \mathbf{n}, \alpha, \mathbf{m})$, we begin by noting that conditional on $z_n^{(d)} = t$, $w_n^{(d)}$ only depends on the tokens assigned to t, so then we have:

$$P(w_n^{(d)} = v \,|\, \mathcal{W}_{\setminus d,n}, \mathcal{Z}, \beta, \mathbf{n}, \alpha, \mathbf{m}) = \frac{N_{\setminus d,n}^{(w_n^{(d)}|t)} + \beta n_v}{\sum_{v'=1}^{V} N_{\setminus d,n}^{(w_n^{(d)}|t)} + \beta n_v} = \frac{N_{\setminus d,n}^{(w_n^{(d)}|t)} + \beta n_v}{N_{\setminus d,n}^{(t)} + \beta} \tag{33}$$

Where $N_{\setminus d,n}^{(w_n^{(d)}|t)}$ is the number of tokens of type $w_n^{(d)}$ assigned to topic t in the corpus, excluding the current position, and $N_{\setminus d,n}^{(t)}$ is the total number tokens assigned to topic t in the corpus. Putting everything together, we have:

$$P(z_n^{(d)} = t, w_n^{(d)} \,|\, \mathcal{W}_{\setminus d,n}, \mathcal{Z}_{\setminus d,n}, \beta, \mathbf{n}, \alpha, \mathbf{m}) = \frac{N_{\setminus d,n}^{(t|d)} + \alpha m_t}{N^{(d)} - 1 + \alpha} \cdot \frac{N_{\setminus d,n}^{(w_n^{(d)}|t)} + \beta n_v}{N_{\setminus d,n}^{(t)} + \beta} \tag{34}$$

$$\propto \left(N_{\setminus d,n}^{(t|d)} + \alpha m_t\right) \cdot \frac{N_{\setminus d,n}^{(w_n^{(d)}|t)} + \beta n_v}{N_{\setminus d,n}^{(t)} + \beta} \tag{35}$$

which is the standard result for collapsed Gibbs sampling for LDA Griffiths (2002).

4.2 A More Complicated Approach

While the approach to deriving the Gibbs sampling equation for LDA described in the previous section is elegant and intuitive, there is some value in demonstrating that the same result can be arrived at using a different approach. In the previous section, we showed above that we could integrate out Φ and Θ, we were left with the building blocks of the collapsed Gibbs sampler, which we can carry through to the final derivation. Note that I will continue to make use of the same notation as was used in that section. Starting with:

$$= \prod_{t=1}^{T} \frac{\Gamma\left(\sum_{v=1}^{V} \beta n_v\right)}{\prod_{v=1}^{V} \Gamma(\beta n_v)} \frac{\prod_{v=1}^{V} \Gamma(\beta n_v + N_{v,t}^{*})}{\Gamma\left(\sum_{v=1}^{V} \beta n_v + N_{v,t}^{*}\right)} \times \prod_{d=1}^{D} \frac{\Gamma\left(\sum_{t=1}^{T} \alpha m_t\right)}{\prod_{t=1}^{T} \Gamma(\alpha m_t)} \frac{\prod_{t=1}^{T} \Gamma(\alpha m_t + N_{*,t}^{(d)})}{\Gamma\left(\sum_{t=1}^{T} \alpha m_t + N_{*,t}^{(d)}\right)} \tag{36}$$

we can see that the leading fractions are only a function of hyper-parameters (which are constant), so we can drop them:

$$\propto \prod_{t=1}^{T} \frac{\prod_{v=1}^{V} \Gamma(\beta n_v + N_{v,t}^{*})}{\Gamma\left(\sum_{v=1}^{V} \beta n_v + N_{v,t}^{*}\right)} \times \prod_{d=1}^{D} \frac{\prod_{t=1}^{T} \Gamma(\alpha m_t + N_{*,t}^{(d)})}{\Gamma\left(\sum_{t=1}^{T} \alpha m_t + N_{*,t}^{(d)}\right)} \tag{37}$$

The way the collapsed Gibbs sampler works is that we sequentially resample values for each of our latent variables of interest from their conditional posterior (conditioned on the current values of all other variables) leaving out information about the current position. In LDA, we want to resample $z_n^{(d)}$. to do this, we need to calculate the following conditional probability:

$$P(z_n^{(d)} = t, w_n^{(d)} \,|\, \mathcal{W}_{\setminus d,n} \mathcal{Z}_{\setminus d,n}, \beta, \mathbf{n}, \alpha, \mathbf{m}) \tag{38}$$

for each possible value of t, and use these to formulate a discrete distribution form which to sample our updated token-topic assignment. Right now, the expression in in equation 37 is proportion to:

$$P(\mathcal{Z}, \mathcal{W} \,|\, \beta, \mathbf{n}, \alpha, \mathbf{m}) \tag{39}$$

so we simply have to separate out the terms that involve the current word in the current document to get something that is proportional to equation 38. We start by separating out terms in equation 37 to a set the depend on the current position (d,j), and those that do not, as follows:

$$\prod_{t=1}^{T} \frac{\prod_{v=1}^{V} \Gamma(\beta n_v + N_{v,t}^{*})}{\Gamma\left(\sum_{v=1}^{V} \beta n_v + N_{v,t}^{*}\right)} \times \prod_{d=1}^{D} \frac{\prod_{t=1}^{T} \Gamma(\alpha m_t + N_{*,t}^{(d)})}{\Gamma\left(\sum_{t=1}^{T} \alpha m_t + N_{*,t}^{(d)}\right)} \tag{40}$$

$$= \prod_{t=1}^{T} \frac{\prod_{v' \neq w_j} \Gamma(\beta n_v + N_{v',t}^{*}) \times \Gamma(\beta n_{w_j} + N_{w_j,t}^{d})}{\Gamma\left(\sum_{v=1}^{V} \beta n_v + N_{v,t}^{*}\right)} \times \prod_{d' \neq d} \frac{\prod_{t=1}^{T} \Gamma(\alpha m_t + N_{*,t}^{(d')})}{\Gamma\left(\sum_{t=1}^{T} \alpha m_t + N_{*,t}^{(d')}\right)} \times \frac{\prod_{t=1}^{T} \Gamma(\alpha m_t + N_{*,t}^{(d)})}{\Gamma\left(\sum_{t=1}^{T} \alpha m_t + N_{*,t}^{(d)}\right)} \tag{41}$$

As we can see, some of these terms do not depend on the current position (d,j), so we can drop them:

$$\propto \prod_{t=1}^{T} \frac{\Gamma(\beta n_{w_j} + N^d_{w_j,t})}{\Gamma\left(\sum_{v=1}^{V} \beta n_v + N^*_{v,t}\right)} \times \frac{\prod_{t=1}^{T} \Gamma(\alpha m_t + N^{(d)}_{*,t})}{\Gamma\left(\sum_{t=1}^{T} \alpha m_t + N^{(d)}_{*,t}\right)} \tag{42}$$

Next we note that $\sum_{t=1}^{T} \alpha m_t + N^{(d)}_{*,t}$ is just the number of tokens in the corpus, which is constant, so it drops out as well:

$$\propto \prod_{t=1}^{T} \frac{\Gamma(\beta n_{w_j} + N^d_{w_j,t})}{\Gamma\left(\sum_{v=1}^{V} \beta n_v + N^*_{v,t}\right)} \times \prod_{t=1}^{T} \Gamma(\alpha m_t + N^{(d)}_{*,t}) \tag{43}$$

To make further progress, we will need to make use of the following helpful property of the Gamma function:

$$\Gamma(y+1) = y \times \Gamma(y) \tag{44}$$

Ultimately, we want to get rid of all of the gamma functions in our expression so that we are left only with counts (which are easy to compute and keep track of). To do this, we will first expand out equation 43 so that we can get a Gamma term that has a + 1 inside of it. To do this, we can note that when $t = z^{(d)}_n$, the count:

$$N^{(d)}_{*,t} = N^{(d)}_{*,t,\backslash_{n,d}} + 1 \tag{45}$$

or the count of the number of times a word of any type co-occurred with the current topic in the current document, excluding the current position (which we know would add 1 since $t = z^{(d)}_n$), would equal that reduced count plus 1. Furthermore, when $t \neq z^{(d)}_n$, the count:

$$N^{(d)}_{*,t} = N^{(d)}_{*,t,\backslash_{n,d}} \tag{46}$$

because the count does not involve the n'th position. Similarly, when $t = z^{(d)}_n$, the count:

$$N^*_{v,t} = N^*_{v,t,\backslash_{n,d}} + 1 \tag{47}$$

or the count of the number of times tokens with word type v topic t co-occur across all tokens in all documents, excluding the current position (which we know would add 1 since $t = z^{(d)}_n$), would equal that reduced count plus 1.Furthermore, when $t \neq z^{(d)}_n$, the count:

$$N^*_{v,t} = N^*_{v,t,\backslash_{n,d}} \tag{48}$$

because the count does not involve the n'th position. So now we can rewrite everything so we get:

$$= \prod_{t' \neq z^{(d)}_n} \frac{\Gamma(\beta n_{w_j} + N^{(d)}_{w_j,t',\backslash_{n,d}})}{\Gamma\left(\sum_{v=1}^{V} \beta n_v + N^*_{v,t',\backslash_{n,d}}\right)} \times \frac{\Gamma(\beta n_{w_j} + N^{(d)}_{w_j,z^{(d)}_n,\backslash_{n,d}} + 1)}{\Gamma\left(\sum_{v=1}^{V} \beta n_v + N^*_{v,z^{(d)}_n,\backslash_{n,d}} + 1\right)} \times$$
$$\prod_{t' \neq z^{(d)}_n} \Gamma(\alpha m'_t + N^{(d)}_{*,t',\backslash_{n,d}}) \times \Gamma(\alpha m_{z^{(d)}_n} + N^{(d)}_{*,z^{(d)}_n,\backslash_{n,d}} + 1) \tag{49}$$

Now things are set up so we can use equation 44, pull a term outside of the Gamma functions, then collapse and sweep away the Gamma function part because it will no longer depend on the current position:

$$= \prod_{t' \neq z^{(d)}_n} \frac{\Gamma(\beta n_{w_j} + N^{(d)}_{w_j,t',\backslash_{n,d}})}{\Gamma\left(\sum_{v=1}^{V} \beta n_v + N^*_{v,t',\backslash_{n,d}}\right)} \times \frac{\Gamma(\beta n_{w_j} + N^{(d)}_{w_j,z^{(d)}_n,\backslash_{n,d}})}{\Gamma\left(\sum_{v=1}^{V} \beta n_v + N^*_{v,z^{(d)}_n,\backslash_{n,d}}\right)} \times \frac{\beta n_{w_j} + N^{(d)}_{w_j,z^{(d)}_n,\backslash_{n,d}}}{\sum_{v=1}^{V} \beta n_v + N^*_{v,z^{(d)}_n,\backslash_{n,d}}} \times$$
$$\prod_{t' \neq z^{(d)}_n} \Gamma(\alpha m'_t + N^{(d)}_{*,t',\backslash_{n,d}}) \times \Gamma(\alpha m_{z^{(d)}_n} + N^{(d)}_{*,z^{(d)}_n,\backslash_{n,d}}) \times \alpha m_{z^{(d)}_n} + N^{(d)}_{*,z^{(d)}_n,\backslash_{n,d}} \tag{50}$$

$$= \prod_{t=1}^{T} \frac{\Gamma(\beta n_{w_j} + N^{(d)}_{w_j,t,\backslash_{n,d}})}{\Gamma\left(\sum_{v=1}^{V} \beta n_v + N^*_{v,t,\backslash_{n,d}}\right)} \times \frac{\Gamma(\beta n_{w_j} + N^{(d)}_{w_j,z^{(d)}_n,\backslash_{n,d}})}{\Gamma\left(\sum_{v=1}^{V} \beta n_v + N^*_{v,z^{(d)}_n,\backslash_{n,d}}\right)} \times \prod_{t=1}^{T} \Gamma(\alpha m_t + N^{(d)}_{*,t,\backslash_{n,d}}) \times \alpha m_{z^{(d)}_n} + N^{(d)}_{*,z^{(d)}_n,\backslash_{n,d}} \tag{51}$$

$$= \frac{\beta n_{w_j} + N^{(d)}_{w_j,z^{(d)}_n,\backslash_{n,d}}}{\sum_{v=1}^{V} \beta n_v + N^*_{v,z^{(d)}_n,\backslash_{n,d}}} \times \alpha m_{z^{(d)}_n} + N^{(d)}_{*,z^{(d)}_n,\backslash_{n,d}} \tag{52}$$

which is the Gibbs sampling equation for LDA.

5 Inference

We actually observe $\mathcal{W}, \mathcal{Y}, \mathcal{X}$, and $\mathcal{A}$ in our email data. What we want to do is infer the values of all of the latent variables $(\mathcal{Z}, \mathcal{L}, \mathcal{B}, \Gamma, \mathcal{S})$ that most likely would have generated the data we observe, if we believe the generative process. To do this, we will need to invert the generative process using Bayes rule.

$$
\begin{aligned}
&P(\text{Latent Vars.} \mid \text{Observed Vars., Metadata, Hyper-Parameters}) \\
&= \frac{P(\text{Latent Vars.} \mid \text{Hyper-Parameters}) P(\text{Observed Vars.} \mid \text{Latent Vars., Metadata, Hyper-Parameters})}{P(\text{Observed Vars.} \mid \text{Metadata, Hyper-Parameters})} && (53) \\
&\propto P(\text{Latent Vars.} \mid \text{Hyper-Parameters}) P(\text{Observed Vars.} \mid \text{Latent Vars., Metadata, Hyper-Parameters}) && (54) \\
&= P(\text{Latent Vars., Observed Vars.} \mid \text{Metadata , Hyper-Parameters}) && (55)
\end{aligned}
$$

Calculating the joint probability of our observed variables (the evidence) is intractable, so we can only know the the probability of the latent variables given the observed variables up to a proportionality.

$$
P(\underbrace{\mathcal{Z}, \mathcal{L}, \mathcal{B}, \Gamma, \mathcal{S}}_{\text{Latent Vars.}} \mid \underbrace{\mathcal{W}, \mathcal{Y}}_{\text{Observed Vars.}} \underbrace{\mathcal{X}, \mathcal{A}}_{\text{Metadata}} \underbrace{\beta, \mathbf{n}, \alpha, \mathbf{m}}_{\text{Hyper-parameters}}) \propto P(\underbrace{\mathcal{Z}, \mathcal{L}, \mathcal{B}, \Gamma, \mathcal{S}, \mathcal{W}, \mathcal{Y}}_{\text{Latent + Observed Vars.}} \mid \underbrace{\mathcal{X}, \mathcal{A}}_{\text{Metadata}} \underbrace{\beta, \mathbf{n}, \alpha, \mathbf{m}}_{\text{Hyper-parameters}}) \tag{56}
$$

To actually perform inference, we therefore want to use Gibbs sampling, where we will sequentially resample the values of each of our latent variables from their posterior distribution, conditional on all of our other variables.

6 Resampling $\mathcal{Z}$

First we will want to resample values for all of the token topic assignments $z_n^{(d)}$. We do this one token at a time, conditional on everything else. Practically, this means constructing a multinomial distribution over topics, and then drawing a new value from this distribution. We start with:

$$
P(z_n^{(d)} = t \mid \mathcal{W}, \mathcal{Z}_{\setminus d,n} \mathcal{L}, \mathcal{B}, \Gamma, \mathcal{S}, \mathcal{Y}, \mathcal{X}, \mathcal{A}) \tag{57}
$$

which via the application of Bayes rule is proportional to:

$$
P(z_n^{(d)} = t, w_n^{(d)}, \mathbf{y}^{(d)} \mid \mathcal{W}_{\setminus d,n} \mathcal{Z}_{\setminus d,n} \mathcal{L}, \mathcal{B}, \Gamma, \mathcal{S}, \mathcal{Y}_{\setminus d}, \mathcal{X}, \mathcal{A}) \tag{58}
$$

Now, our generative process implies a particular factorization of this distribution into a product of two conditional probabilities:

$$
= \underbrace{P(z_n^{(d)} = t, w_n^{(d)} \mid \mathcal{W}_{\setminus d,n}, \mathcal{Z}_{\setminus d,n}, \beta, \mathbf{n}, \alpha, \mathbf{m})}_{\text{LDA Contribution}} \underbrace{P(\mathbf{y}^{(d)} \mid z_n^{(d)} = t, \mathcal{Z}_{\setminus d,n} \mathcal{L}, \mathcal{B}, \Gamma, \mathcal{S}, \mathcal{Y}_{\setminus d}, \mathcal{X}, \mathcal{A})}_{\text{LSM Contribution}} \tag{59}
$$

The first of these is equivalent to the conditional probability of interest in vanilla LDA, and we have already derived the resampling equation in section 4. The second term is related to the LSM portion of the model, and will be addressed below. To make the model applicable to communication data where the messages contain no tokens we will need to adopt a $\bar{N}$ notation, where $\bar{N} = \max(1, N)$ so that we can cleanly handle messages without tokens. The reason for adopting this approach is that emails which contain no text may still provide valuable relational information. We also expect to encounter these types of messages with some regularity, as people often forward messages without further comment. Therefore, when resampling token topic assignments, we will have to deal with the case where $N^{(d)} > 0$ and the case where $N^{(d)} = 0$ separately.

6.1 The case where the email contains words ($N^{(d)} > 0$)

In the case where there is at least one token associated with the current email, we can proceed along the same lines as one would when resampling $z_n^{(d)}$ in vanilla LDA, since $\bar{N} = N$ when $N \geqslant 1$, but with an addition probability contribution from the interaction pattern part of the model. The resampling equation for the LDA term in equation 59 can therefore just be derived in the same manner as we did for LDA in section 4, now adopting the $\bar{N}$ notation.

$$
P(z_n^{(d)} = t, w_n^{(d)} \mid \mathcal{W}_{\setminus d,n}, \mathcal{Z}_{\setminus d,n}, \beta, \mathbf{n}, \alpha, \mathbf{m}) = \frac{\bar{N}_{\setminus d,n}^{(t|d)} + \alpha m_t}{\bar{N}^{(d)} - 1 + \alpha} \cdot \frac{\bar{N}_{\setminus d,n}^{(w_n^{(d)}|t)} + \beta n_v}{\bar{N}_{\setminus d,n}^{(t)} + \beta} \tag{60}
$$

$$
\propto \left(\bar{N}_{\setminus d,n}^{(t|d)} + \alpha m_t \right) \cdot \frac{\bar{N}_{\setminus d,n}^{(w_n^{(d)}|t)} + \beta n_v}{\bar{N}_{\setminus d,n}^{(t)} + \beta} \tag{61}
$$

Now we have to deal with the interaction pattern contribution in equation 59.

$$\underbrace{P(\mathbf{y}^{(d)} \mid z_n^{(d)} = t, \mathcal{Z}_{\setminus d,n}\mathcal{L}, \mathcal{B}, \Gamma, \mathcal{S}, \mathcal{Y}_{\setminus d}, \mathcal{X}, \mathcal{A})}_{\text{LSM Contribution}} \tag{62}$$

If we go back to the generative process, we first see that the emails are independent of each other, so we can get rid of $\mathcal{Y}_{\setminus d}$. Now we can start from the probability that an edge is present between a message sender a and a potential recipient r, for a given interaction pattern c, following the derivation of the LSM Hoff et al. (2002):

$$p_{ar}^{(c)} = \sigma(b^{(c)} + \boldsymbol{\gamma}^{(c)\top}\boldsymbol{x}^{(ar)} - ||\boldsymbol{s}_a^{(c)} - \boldsymbol{s}_r^{(c)}||) \tag{63}$$

This will be our building block for calculating $P(\mathbf{y}^{(d)} \mid ...)$. We also know from the formulation of the LSM that in the case where all tokens are assigned to a single topic (and thus associated with a single interaction pattern):

$$P(\mathbf{y}^{(d)}) = \prod_{r=1}^{A} \left(p_{a^{(d)}r}^{(c)}\right)^{y_r^{(d)}} \left(1 - p_{a^{(d)}r}^{(c)}\right)^{1-y_r^{(d)}} \tag{64}$$

where $a^{(d)}$ is the the author of the current email, $y_r^{(d)}$ is an indicator of whether potential recipient r was actually included as a recipient of the email, and the exponents therefore toggle whether we are multiplying in the probability of a present or absent edge for each actor. However, our model allows for multiple different interaction patterns, each of which is associated with a topic. Therefore we need to take into account the way our generative process aggregates over the probability of a present (absent) edge existing between two actors in each latent space c – $p_{ar}^{(c)}$ to draw a value for $y_r^{(d)}$ from the generative process:

$$y_r^{(d)} \sim \text{Bern}(\sum_{t=1}^{T} \frac{\bar{N}^{(t|d)}}{\bar{N}^{(d)}} p_{a^{(d)}r}^{(l_t)}) \tag{65}$$

We can see that the Bernoulli probability is an average of the probabilities of a present edge in each latent space, weighted by the token-topic counts for topics associated with that interaction pattern c. So we take all of this information together and then we form the conditional probability of observing the recipients we actually did, for this particular email, given the current token topic assignments (excluding the current position) and current interaction pattern parameters.

$$\begin{aligned}
&P(\mathbf{y}^{(d)} \mid z_n^{(d)} = t, \mathcal{Z}_{\setminus d,n}\mathcal{L}, \mathcal{B}, \Gamma, \mathcal{S}, \mathcal{Y}_{\setminus d}, \mathcal{X}, \mathcal{A}) \\
&= \prod_{r=1}^{A} \left(\sum_{t'=1}^{T} \left[\frac{\bar{N}_{\setminus d,n}^{(t'|d)}}{\bar{N}^{(d)}} + \frac{\delta(t'=t)}{\bar{N}^{(d)}}\right] p_{a^{(d)}r}^{(l'_t)}\right)^{y_r^{(d)}} \left(1 - \sum_{t'=1}^{T} \left[\frac{\bar{N}_{\setminus d,n}^{(t'|d)}}{\bar{N}^{(d)}} + \frac{\delta(t'=t)}{\bar{N}^{(d)}}\right] p_{a^{(d)}r}^{(l'_t)}\right)^{1-y_r^{(d)}}
\end{aligned} \tag{66}$$

We can now combine these two terms to arrive at the full conditional probability, which is:

$$\begin{aligned}
&P(z_n^{(d)} = t \mid \mathcal{W}, \mathcal{Z}_{\setminus d,n}\mathcal{L}, \mathcal{B}, \Gamma, \mathcal{S}, \mathcal{Y}, \mathcal{X}, \mathcal{A}) \\
&\propto \left(\bar{N}_{\setminus d,n}^{(t|d)} + \alpha m_t\right) \cdot \frac{\bar{N}_{\setminus d,n}^{(w_n^{(d)}|t)} + \beta n_v}{\bar{N}_{\setminus d,n}^{(t)} + \beta} \times \prod_{r=1}^{A} \left(\sum_{t'=1}^{T} \left[\frac{\bar{N}_{\setminus d,n}^{(t'|d)}}{\bar{N}^{(d)}} + \frac{\delta(t'=t)}{\bar{N}^{(d)}}\right] p_{a^{(d)}r}^{(l'_t)}\right)^{y_r^{(d)}} \left(1 - \sum_{t'=1}^{T} \left[\frac{\bar{N}_{\setminus d,n}^{(t'|d)}}{\bar{N}^{(d)}} + \frac{\delta(t'=t)}{\bar{N}^{(d)}}\right] p_{a^{(d)}r}^{(l'_t)}\right)^{1-y_r^{(d)}}
\end{aligned} \tag{67}$$

where l'_t is the current cluster assignment of the topic t'.

6.1.1 Simplifying equation 67

Now lets see if we can simplify this equation. We begin by noting that the the exponents involving $y_r^{(d)}$ are designed to function as a multiplicative delta function, ensuring that when $y_r^{(d)} = 1$, only the left term inside the product over A matters, and when $y_r^{(d)} = 0$, only the right term matters. We can rewrite this equivalently using multiplication/addition as follows:

$$\begin{aligned}
&\prod_{r=1}^{A} \left(\sum_{t'=1}^{T} \left[\frac{\bar{N}_{\setminus d,n}^{(t'|d)}}{\bar{N}^{(d)}} + \frac{\delta(t'=t)}{\bar{N}^{(d)}}\right] p_{a^{(d)}r}^{(l'_t)}\right)^{y_r^{(d)}} \left(1 - \sum_{t'=1}^{T} \left[\frac{\bar{N}_{\setminus d,n}^{(t'|d)}}{\bar{N}^{(d)}} + \frac{\delta(t'=t)}{\bar{N}^{(d)}}\right] p_{a^{(d)}r}^{(l'_t)}\right)^{1-y_r^{(d)}} \\
&= \prod_{r=1}^{A} \left\{ y_r^{(d)} \times \left(\sum_{t'=1}^{T} \left[\frac{\bar{N}_{\setminus d,n}^{(t'|d)}}{\bar{N}^{(d)}} + \frac{\delta(t'=t)}{\bar{N}^{(d)}}\right] p_{a^{(d)}r}^{(l'_t)}\right) + (1 - y_r^{(d)}) \times \left(1 - \sum_{t'=1}^{T} \left[\frac{\bar{N}_{\setminus d,n}^{(t'|d)}}{\bar{N}^{(d)}} + \frac{\delta(t'=t)}{\bar{N}^{(d)}}\right] p_{a^{(d)}r}^{(l'_t)}\right) \right\}
\end{aligned} \tag{68}$$

We can do this because multiplying by one is equivalent to adding zero. The only reason this works is because our edge values are binary, so we always get zero or one. Now we can further separate out terms on the inside:

$$= \prod_{r=1}^{A} y_r^{(d)} \times \sum_{t'=1}^{T} \frac{\bar{N}_{\backslash d,n}^{(t'|d)}}{\bar{N}^{(d)}} p_{a^{(d)}r}^{(l'_t)} + y_r^{(d)} \times \sum_{t'=1}^{T} \frac{\delta(t'=t)}{\bar{N}^{(d)}} p_{a^{(d)}r}^{(l'_t)} + (1-y_r^{(d)}) \times \left(1 - \sum_{t'=1}^{T} \frac{\bar{N}_{\backslash d,n}^{(t'|d)}}{\bar{N}^{(d)}} p_{a^{(d)}r}^{(l'_t)}\right)$$
$$+ (1-y_r^{(d)}) \times \left(1 - \sum_{t'=1}^{T} \frac{\delta(t'=t)}{\bar{N}^{(d)}} p_{a^{(d)}r}^{(l'_t)}\right) \tag{69}$$

Now since the term $\frac{\delta(t'=t)}{\bar{N}^{(d)}}$ will only have a non-zero value when $t'=t$, the above simplifies to:

$$= \prod_{r=1}^{A} y_r^{(d)} \times \sum_{t'=1}^{T} \frac{\bar{N}_{\backslash d,n}^{(t'|d)}}{\bar{N}^{(d)}} p_{a^{(d)}r}^{(l'_t)} + y_r^{(d)} \times \frac{1}{\bar{N}^{(d)}} p_{a^{(d)}r}^{(l_t)} + (1-y_r^{(d)}) \times \left(1 - \sum_{t'=1}^{T} \frac{\bar{N}_{\backslash d,n}^{(t'|d)}}{\bar{N}^{(d)}} p_{a^{(d)}r}^{(l'_t)}\right)$$
$$+ (1-y_r^{(d)}) \times \left(1 - \frac{1}{\bar{N}^{(d)}} p_{a^{(d)}r}^{(l_t)}\right) \tag{70}$$

This expression cannot be simplified further, but clearly illustrates the potential for caching to speed up computation time, as each $\bar{N}_{\backslash d,n}^{(t'|d)}$ and $p_{a^{(d)}r}^{(l'_t)}$ only needs to be calculated once, and could actually be kept in a global table. Thus we can rewrite equation 67 as:

$$P(z_n^{(d)} = t \,|\, \mathcal{W}, \mathcal{Z}_{\backslash d,n}\mathcal{L}, \mathcal{B}, \Gamma, \mathcal{S}, \mathcal{Y}, \mathcal{X}, \mathcal{A})$$
$$\propto \bar{N}_{\backslash d,n}^{(t|d)} + \alpha m_t \cdot \frac{\bar{N}_{\backslash d,n}^{(w_n^{(d)}|t)} + \beta n_v}{\bar{N}_{\backslash d,n}^{(t)} + \beta} \times \prod_{r=1}^{A} \left\{ y_r^{(d)} \times \sum_{t'=1}^{T} \frac{\bar{N}_{\backslash d,n}^{(t'|d)}}{\bar{N}^{(d)}} p_{a^{(d)}r}^{(l'_t)} + y_r^{(d)} \times \frac{1}{\bar{N}^{(d)}} p_{a^{(d)}r}^{(l_t)} + \right.$$
$$\left. (1-y_r^{(d)}) \times \left(1 - \sum_{t'=1}^{T} \frac{\bar{N}_{\backslash d,n}^{(t'|d)}}{\bar{N}^{(d)}} p_{a^{(d)}r}^{(l'_t)}\right) + (1-y_r^{(d)}) \times \left(1 - \frac{1}{\bar{N}^{(d)}} p_{a^{(d)}r}^{(l_t)}\right) \right\} \tag{71}$$

6.1.2 Taking Logs

To help address potential problems with numerical underflow in the large product in Equation 71, we can follow the common practice of working in log space. To do this, we need to take the log of Equation 71:

$$\log \left[\bar{N}_{\backslash d,n}^{(t|d)} + \alpha m_t \cdot \frac{\bar{N}_{\backslash d,n}^{(w_n^{(d)}|t)} + \beta n_v}{\bar{N}_{\backslash d,n}^{(t)} + \beta} \times \prod_{r=1}^{A} \left\{ y_r^{(d)} \times \sum_{t'=1}^{T} \frac{\bar{N}_{\backslash d,n}^{(t'|d)}}{\bar{N}^{(d)}} p_{a^{(d)}r}^{(l'_t)} + y_r^{(d)} \times \frac{1}{\bar{N}^{(d)}} p_{a^{(d)}r}^{(l_t)} + \right.\right.$$
$$\left.\left. (1-y_r^{(d)}) \times \left(1 - \sum_{t'=1}^{T} \frac{\bar{N}_{\backslash d,n}^{(t'|d)}}{\bar{N}^{(d)}} p_{a^{(d)}r}^{(l'_t)}\right) + (1-y_r^{(d)}) \times \left(1 - \frac{1}{\bar{N}^{(d)}} p_{a^{(d)}r}^{(l_t)}\right) \right\} \right] \tag{72}$$

$$= \log \left[\bar{N}_{\backslash d,n}^{(t|d)} + \alpha m_t \cdot \frac{\bar{N}_{\backslash d,n}^{(w_n^{(d)}|t)} + \beta n_v}{\bar{N}_{\backslash d,n}^{(t)} + \beta} \right] + \sum_{r=1}^{A} \log \left[y_r^{(d)} \times \sum_{t'=1}^{T} \frac{\bar{N}_{\backslash d,n}^{(t'|d)}}{\bar{N}^{(d)}} p_{a^{(d)}r}^{(l'_t)} + y_r^{(d)} \times \frac{1}{\bar{N}^{(d)}} p_{a^{(d)}r}^{(l_t)} + \right.$$
$$\left. (1-y_r^{(d)}) \times \left(1 - \sum_{t'=1}^{T} \frac{\bar{N}_{\backslash d,n}^{(t'|d)}}{\bar{N}^{(d)}} p_{a^{(d)}r}^{(l'_t)}\right) + (1-y_r^{(d)}) \times \left(1 - \frac{1}{\bar{N}^{(d)}} p_{a^{(d)}r}^{(l_t)}\right) \right] \tag{73}$$

It is important to note that we will only ever evaluate two the four terms in the large summation in practice, as the other two will automatically drop out since they are dependent on whether we observe a present or absent edge.

6.2 The case where the email does not contain words ($N^{(d)} = 0$)

In the case where $N^{(d)} = 0$, we assign a dummy token that can take a topic assignment as a way to associate the document to an interaction pattern. If we look at 67 we can see that when we have no tokens in the current document, the LDA contribution

becomes constant in t. Thus we can ignore it and are left with:

$$P(z_n^{(d)} = t \,|\, \mathcal{W}, \mathcal{Z}_{\backslash d,n}\mathcal{L}, \mathcal{B}, \Gamma, \mathcal{S}, \mathcal{Y}, \mathcal{X}, \mathcal{A})$$
$$\propto \prod_{r=1}^{A} \left(\sum_{t'=1}^{T} \left[\frac{\bar{N}_{\backslash d,n}^{(t'|d)}}{\bar{N}^{(d)}} + \frac{\delta(t'=t)}{\bar{N}^{(d)}} \right] p_{a^{(d)}r}^{(l'_t)} \right)^{y_r^{(d)}} \left(1 - \sum_{t'=1}^{T} \left[\frac{\bar{N}_{\backslash d,n}^{(t'|d)}}{\bar{N}^{(d)}} + \frac{\delta(t'=t)}{\bar{N}^{(d)}} \right] p_{a^{(d)}r}^{(l'_t)} \right)^{1-y_r^{(d)}} \tag{74}$$

but here again, the term $\bar{N}_{\backslash d,n}^{(t'|d)}$ is just going to equal zero, so the only contribution from the sum over t will come from the case where $t' = t$. Therefore, this expression reduces down to equation 64, where l_t is the interaction pattern associated with topic t:

$$P(z_n^{(d)} = t \,|\, \mathcal{W}, \mathcal{Z}_{\backslash d,n}\mathcal{L}, \mathcal{B}, \Gamma, \mathcal{S}, \mathcal{Y}, \mathcal{X}, \mathcal{A}) \propto \prod_{r=1}^{A} \left(p_{a^{(d)}r}^{(l_t)} \right)^{y_r^{(d)}} \left(1 - p_{a^{(d)}r}^{(l_t)} \right)^{1-y_r^{(d)}} \tag{75}$$

6.2.1 Taking Logs

As we are going to be working in log space, we will need to take logs of Equation 75 to perform inference. Thus we end up with:

$$\log \left[\prod_{r=1}^{A} \left(p_{a^{(d)}r}^{(l_t)} \right)^{y_r^{(d)}} \left(1 - p_{a^{(d)}r}^{(l_t)} \right)^{1-y_r^{(d)}} \right] \tag{76}$$
$$= \sum_{r=1}^{A} \log \left[\left(p_{a^{(d)}r}^{(l_t)} \right)^{y_r^{(d)}} \left(1 - p_{a^{(d)}r}^{(l_t)} \right)^{1-y_r^{(d)}} \right] \tag{77}$$
$$= \sum_{r=1}^{A} y_r^{(d)} \log \left(p_{a^{(d)}r}^{(l_t)} \right) + (1 - y_r^{(d)}) \log \left(1 - p_{a^{(d)}r}^{(l_t)} \right) \tag{78}$$

7 Resampling $\mathcal{L}$

The next set of variables we need to resample are the topic-interaction pattern assignments. Thus the conditional posterior probability we want to calculate is:

$$P(l_t = c \,|\, \mathcal{Z}, \mathcal{W}, \mathcal{L}_{\backslash t}, \mathcal{B}, \Gamma, \mathcal{S}, \mathcal{Y}, \mathcal{X}, \mathcal{A}, \beta, \mathbf{n}, \alpha, \mathbf{m}) \tag{79}$$

In words, what we want is the conditional posterior probability that topic t is associated interaction pattern c, given everything else. But if we apply Bayes rule, we can also see that this is proportional to:

$$P(l_t = c, \mathcal{Y} \,|\, \mathcal{Z}, \mathcal{W}, \mathcal{L}_{\backslash t}, \mathcal{B}, \Gamma, \mathcal{S}, \mathcal{X}, \mathcal{A}, \beta, \mathbf{n}, \alpha, \mathbf{m}) \tag{80}$$

which via the chain rule is equal to:

$$P(l_t = c) \cdot P(\mathcal{Y} \,|\, \mathcal{Z}, \mathcal{W}, \mathcal{L}_{\backslash t}, \mathcal{B}, \Gamma, \mathcal{S}, \mathcal{X}, \mathcal{A}, \beta, \mathbf{n}, \alpha, \mathbf{m}) \tag{81}$$

This is a useful formulation because the term $P(l_t = c)$ is just the prior probability that a topic is assigned to any interaction pattern, which is constant and uniform and therefore can be dropped so that:

$$P(l_t = c \,|\, \mathcal{Z}, \mathcal{W}, \mathcal{L}_{\backslash t}, \mathcal{B}, \Gamma, \mathcal{S}, \mathcal{Y}, \mathcal{X}, \mathcal{A}, \beta, \mathbf{n}, \alpha, \mathbf{m}) \propto P(\mathcal{Y} \,|\, \mathcal{Z}, \mathcal{W}, \mathcal{L}_{\backslash t}, \mathcal{B}, \Gamma, \mathcal{S}, \mathcal{X}, \mathcal{A}, \beta, \mathbf{n}, \alpha, \mathbf{m}) \tag{82}$$

This formulation is very nice since we have already derived most of it in the previous section, and is just equal to the product over all documents and all potential recipients of the probability that we observed the edge values we actually do, if $l_t = c$ for the current topic.

$$= \prod_{d=1}^{D} \prod_{r=1}^{A} \left(\sum_{t'=1}^{T} \frac{\bar{N}^{(t'|d)}}{\bar{N}^{(d)}} p_{a^{(d)}r}^{(l'_t)} \right)^{y_r^{(d)}} \left(1 - \sum_{t'=1}^{T} \frac{\bar{N}^{(t'|d)}}{\bar{N}^{(d)}} p_{a^{(d)}r}^{(l'_t)} \right)^{1-y_r^{(d)}} \tag{83}$$

However, looking at this further we see that there are some terms which will be constant, and can be ignored. For example, since the interaction pattern assignment of t is the only thing we are varying, those emails that do not contain any tokens assigned to topic t will be a constant multiplicative factor, and can thus be ignored:

$$\propto \prod_{d:\bar{N}^{(t|d)}>0} \prod_{r=1}^{A} \left(\sum_{t'=1}^{T} \frac{\bar{N}^{(t'|d)}}{\bar{N}^{(d)}} p_{a^{(d)}r}^{(l'_t)} \right)^{y_r^{(d)}} \left(1 - \sum_{t'=1}^{T} \frac{\bar{N}^{(t'|d)}}{\bar{N}^{(d)}} p_{a^{(d)}r}^{(l'_t)} \right)^{1-y_r^{(d)}} \tag{84}$$

Now we apply the same trick we did in equation 68 to rewrite the above as:

$$= \prod_{d:\bar{N}^{(t|d)}>0} \prod_{r=1}^{A} y_r^{(d)} \times \left(\sum_{t'=1}^{T} \frac{\bar{N}^{(t'|d)}}{\bar{N}^{(d)}} p^{(l'_t)}_{a^{(d)}r} \right) + (1 - y_r^{(d)}) \times \left(1 - \sum_{t'=1}^{T} \frac{\bar{N}^{(t'|d)}}{\bar{N}^{(d)}} p^{(l'_t)}_{a^{(d)}r} \right) \tag{85}$$

From here, we can split up the sum over t' into counts involving t and those not involving t:

$$\begin{aligned} &= \prod_{d:\bar{N}^{(t|d)}>0} \prod_{r=1}^{A} y_r^{(d)} \times \left(\sum_{t' \neq t} \frac{\bar{N}^{(t'|d)}}{\bar{N}^{(d)}} p^{(l'_t)}_{a^{(d)}r} \right) + y_r^{(d)} \times \left(\frac{\bar{N}^{(t|d)}}{\bar{N}^{(d)}} p^{(l_t)}_{a^{(d)}r} \right) + (1 - y_r^{(d)}) \times \left(1 - \sum_{t' \neq t} \frac{\bar{N}^{(t'|d)}}{\bar{N}^{(d)}} p^{(l'_t)}_{a^{(d)}r} \right) \\ &+ (1 - y_r^{(d)}) \times \left(1 - \frac{\bar{N}^{(t|d)}}{\bar{N}^{(d)}} p^{(l_t)}_{a^{(d)}r} \right) \end{aligned} \tag{86}$$

Similar to the case with resampling z, this expression cannot be simplified further, but clearly illustrates the potential for caching to speed up computation time, as each $\bar{N}^{(t'|d)}_{\setminus d,n}$ and $p^{(l'_t)}_{a^{(d)}r}$ only needs to be calculated once.

7.0.1 Taking Logs

As we are going to be working in log space, we will need to take logs of Equation 86 to perform inference. Thus we end up with:

$$\begin{aligned} &\log \left[\prod_{d:\bar{N}^{(t|d)}>0} \prod_{r=1}^{A} y_r^{(d)} \times \left(\sum_{t' \neq t} \frac{\bar{N}^{(t'|d)}}{\bar{N}^{(d)}} p^{(l'_t)}_{a^{(d)}r} \right) + y_r^{(d)} \times \left(\frac{\bar{N}^{(t|d)}}{\bar{N}^{(d)}} p^{(l_t)}_{a^{(d)}r} \right) + (1 - y_r^{(d)}) \times \left(1 - \sum_{t' \neq t} \frac{\bar{N}^{(t'|d)}}{\bar{N}^{(d)}} p^{(l'_t)}_{a^{(d)}r} \right) \right. \\ &\left. + (1 - y_r^{(d)}) \times \left(1 - \frac{\bar{N}^{(t|d)}}{\bar{N}^{(d)}} p^{(l_t)}_{a^{(d)}r} \right) \right] \end{aligned} \tag{87}$$

$$\begin{aligned} &= \sum_{d:\bar{N}^{(t|d)}>0} \sum_{r=1}^{A} \log \left[y_r^{(d)} \times \left(\sum_{t' \neq t} \frac{\bar{N}^{(t'|d)}}{\bar{N}^{(d)}} p^{(l'_t)}_{a^{(d)}r} \right) + y_r^{(d)} \times \left(\frac{\bar{N}^{(t|d)}}{\bar{N}^{(d)}} p^{(l_t)}_{a^{(d)}r} \right) + (1 - y_r^{(d)}) \times \left(1 - \sum_{t' \neq t} \frac{\bar{N}^{(t'|d)}}{\bar{N}^{(d)}} p^{(l'_t)}_{a^{(d)}r} \right) \right. \\ &\left. + (1 - y_r^{(d)}) \times \left(1 - \frac{\bar{N}^{(t|d)}}{\bar{N}^{(d)}} p^{(l_t)}_{a^{(d)}r} \right) \right] \end{aligned} \tag{88}$$

It is important to note again that we will only ever evaluate two the four terms in the large summation in practice, as the other two will automatically drop out since they are dependent on whether we observe a present or absent edge.

8 Resampling $\mathcal{B}, \Gamma, \mathcal{S}$

The last quantities we wish to update are the interaction pattern parameters $\mathcal{B}, \Gamma, \mathcal{S}$. We can do this using the Metropolis-Hastings algorithm, where we propose updated values for the parameters from a proposal distribution centered at the current values, and then accept or reject the proposed values based on our probability model. The general idea behind the Metropolis Hasting algorithm is that by repeating this process many times, we will eventually converge to the posterior distribution over the parameters, which we can then use to approximate their true values. Since we have placed multivariate Gaussian priors on all of our interaction pattern parameters, sampling new values for these parameteres is relatively straightforward. The two main challenges we face are in working out the probability of accepting a proposed set of parameter values, and in tuning the proposal distribution variances so that we accept a reasonable proportion of proposed parameter value updates. Below, I will derive an expression for calculating the probability of accepting a new proposal, and then discuss how we can find an optimal proposal variance using a simple adaptive Metropolis algorithm.

8.1 Proposal Acceptance Probability

The Metropolis Hastings algorithm specifies the following general equation for the probability of accepting a proposal:

$$\text{Acceptance Probability } = \frac{\text{Q(Current} \mid \text{Proposed)}}{Q(\text{Proposed} \mid \text{Current})} \frac{\text{P(Proposed} \mid \text{Data)}}{Q(\text{Current} \mid \text{Data})} \tag{89}$$

Our goal is to derive expressions for the component probabilities. We begin with the general form of our model. We propose new values $\langle \mathcal{B}', \Gamma', \mathcal{S}' \rangle$ from a multivariate normal distribution with mean vector $\langle \mathcal{B}, \Gamma, \mathcal{S} \rangle$ and diagonal covariance matrix

$\sigma^2 I$. We then accept this new proposal probability equal to:

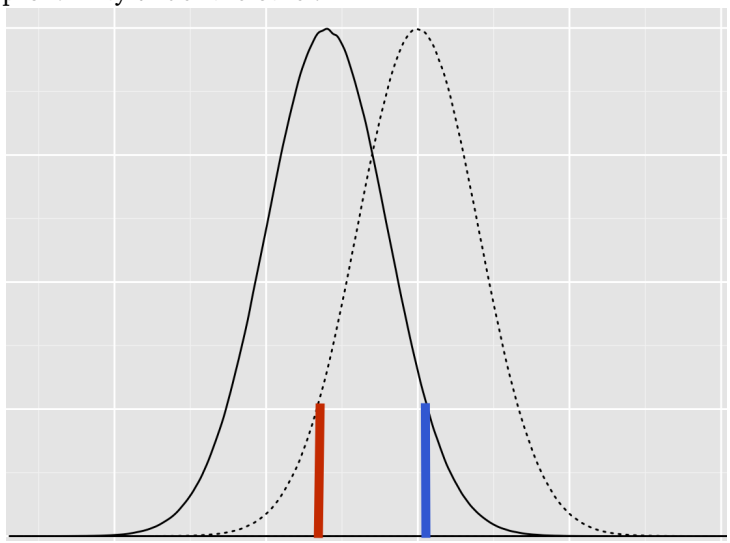
$$\text{Acceptance Probability} = \begin{cases} \frac{Q(\mathcal{B},\Gamma,\mathcal{S}\,|\,\mathcal{B}',\Gamma',\mathcal{S}')}{Q(\mathcal{B}',\Gamma',\mathcal{S}'\,|\,\mathcal{B},\Gamma,\mathcal{S})} \cdot \frac{P(\mathcal{B}',\Gamma',\mathcal{S}'\,|\,\mathcal{Z},\mathcal{W},\mathcal{L},\mathcal{Y},\mathcal{X},\mathcal{A})}{P(\mathcal{B},\Gamma,\mathcal{S}\,|\,\mathcal{Z},\mathcal{W},\mathcal{L},\mathcal{Y},\mathcal{X},\mathcal{A})} & \text{if } < 1 \\ 1 & \text{else} \end{cases} \tag{90}$$

All we have done here is to plug in our interaction pattern parameters and data into equation 89. The reason we have a conditional statement here is that the ratio can be greater than one, in which case our probability of accepting the proposal will simply be equal to one. Fortunately, because we are using a symmetric proposal distribution (the Multivariate Gaussian distribution), we know:

$$Q(\mathcal{B},\Gamma,\mathcal{S}\,|\,\mathcal{B}',\Gamma',\mathcal{S}') = Q(\mathcal{B}',\Gamma',\mathcal{S}'\,|\,\mathcal{B},\Gamma,\mathcal{S}) \tag{91}$$

so the Q-ratio cancels out. A graphical illustration is provided in 1.

Figure 1: Example of two Gaussian distributions with identical variances but different means. We can see that the mean of each distribution has identical probability under the other.



Therefore we just have to calculate the P ratio:

$$\frac{P(\mathcal{B}',\Gamma',\mathcal{S}'\,|\,\mathcal{Z},\mathcal{W},\mathcal{L},\mathcal{Y},\mathcal{X},\mathcal{A})}{P(\mathcal{B},\Gamma,\mathcal{S}\,|\,\mathcal{Z},\mathcal{W},\mathcal{L},\mathcal{Y},\mathcal{X},\mathcal{A})} \tag{92}$$

in order to determine whether we accept or reject our new proposal for the values of $\langle \mathcal{B}',\Gamma',\mathcal{S}' \rangle$. To do this, we start by noting that:

$$\frac{P(\mathcal{B}',\Gamma',\mathcal{S}'\,|\,\mathcal{Z},\mathcal{W},\mathcal{L},\mathcal{Y},\mathcal{X},\mathcal{A})}{P(\mathcal{B},\Gamma,\mathcal{S}\,|\,\mathcal{Z},\mathcal{W},\mathcal{L},\mathcal{Y},\mathcal{X},\mathcal{A})} = \frac{P(\mathcal{B}',\Gamma',\mathcal{S}',\mathcal{Z},\mathcal{W},\mathcal{L},\mathcal{Y}\,|\,\mathcal{X},\mathcal{A})}{P(\mathcal{B},\Gamma,\mathcal{S},\mathcal{Z},\mathcal{W},\mathcal{L},\mathcal{Y}\,|\,\mathcal{X},\mathcal{A})} \tag{93}$$

because the normalizing constants involved in going between these two representations involve marginalizing over $\mathcal{Z},\mathcal{W},\mathcal{L},\mathcal{Y}$ which are fixed in $\mathcal{B},\Gamma,\mathcal{S}$, and are therefore equal for the numerator and denominator so they cancel out. Now that we are working with the joint distribution, we can factor the probabilities in the numerator and denominator of equation 92 according to the factorization implied by our generative process.

$$\begin{aligned} &\frac{P(\mathcal{B}',\Gamma',\mathcal{S}',\mathcal{Z},\mathcal{W},\mathcal{L},\mathcal{Y}\,|\,\mathcal{X},\mathcal{A})}{P(\mathcal{B},\Gamma,\mathcal{S},\mathcal{Z},\mathcal{W},\mathcal{L},\mathcal{Y}\,|\,\mathcal{X},\mathcal{A})} \\ &= \frac{P(\mathcal{Y}\,|\,\mathcal{B}',\Gamma',\mathcal{S}',\mathcal{Z},\mathcal{L}\,|\,\mathcal{X},\mathcal{A})P(\mathcal{W}\,|\,\mathcal{Z})P(\mathcal{Z})P(\mathcal{L})P(\mathcal{B}')P(\Gamma')P(\mathcal{S}')}{P(\mathcal{Y}\,|\,\mathcal{B},\Gamma,\mathcal{S},\mathcal{Z},\mathcal{L}\mathcal{X},\mathcal{A})P(\mathcal{W}\,|\,\mathcal{Z})P(\mathcal{Z})P(\mathcal{L})P(\mathcal{B})P(\Gamma)P(\mathcal{S})} \qquad (94) \\ &= \frac{P(\mathcal{Y}\,|\,\mathcal{B}',\Gamma',\mathcal{S}',\mathcal{Z},\mathcal{L}\,|\,\mathcal{X},\mathcal{A})P(\mathcal{B}')P(\Gamma')P(\mathcal{S}')}{P(\mathcal{Y}\,|\,\mathcal{B},\Gamma,\mathcal{S},\mathcal{Z},\mathcal{L}\mathcal{X},\mathcal{A})P(\mathcal{B})P(\Gamma)P(\mathcal{S})} \qquad (95) \end{aligned}$$

So now we have two sets of terms: $P(\mathcal{B})P(\Gamma)P(\mathcal{S})$ and $P(\mathcal{B}')P(\Gamma')P(\mathcal{S}')$ which are the prior probabilities of the current and proposed interaction pattern parameters; and $P(\mathcal{Y}\,|\,\mathcal{B}',\Gamma',\mathcal{S}',\mathcal{Z},\mathcal{L}\,|\,\mathcal{X},\mathcal{A})$ and $P(\mathcal{Y}\,|\,\mathcal{B}',\Gamma',\mathcal{S}',\mathcal{Z},\mathcal{L}\,|\,\mathcal{X},\mathcal{A})$ which are just the probabilities of observing the edges we do under the current and proposed interaction pattern parameters. The prior

probabilities can be calculated simply using the parameter values and an appropriate multivariate normal probability density function for the prior distribution on the parameters. In general, we select a multivariate Gaussian prior with mean vector zero and a moderate variance (on the order of 5) for the interaction pattern parameters. It is important to select an informative prior as the latent space model prefers to place nodes who are not connected infinitely far away from each other.

We have already done most of the work of deriving the functional form for the second set of terms in Section 7. In particular, equation 83 provides us with the function form of both the numerator and denominator of the P-ratio, excluding the priors. Stylistically, we can rewrite the product of the two terms raised to the edge value as a sum of products:

$$\begin{aligned}
&\prod_{d=1}^{D}\prod_{r=1}^{A}\left(\sum_{t'=1}^{T}\frac{\bar{N}^{(t'|d)}}{\bar{N}^{(d)}}p^{(l'_t)}_{a^{(d)}r}\right)^{y_r^{(d)}}\left(1-\sum_{t'=1}^{T}\frac{\bar{N}^{(t'|d)}}{\bar{N}^{(d)}}p^{(l'_t)}_{a^{(d)}r}\right)^{1-y_r^{(d)}}\\
&=\prod_{d=1}^{D}\prod_{r=1}^{A}\left[y_r^{(d)}\times\left(\sum_{t=1}^{T}\frac{\bar{N}^{(t|d)}}{\bar{N}^{(d)}}p^{(l_t)}_{a^{(d)}r}\right)+(1-y_r^{(d)})\times\left(1-\sum_{t=1}^{T}\frac{\bar{N}^{(t|d)}}{\bar{N}^{(d)}}p^{(l_t)}_{a^{(d)}r}\right)\right]
\end{aligned}\tag{96}$$

This works out because $y_r^{(d)}$ can only ever take the values of 0 or 1, so these two representations are mathematically equivalent. Now we have enough information to write down the full functional form of the P-ratio:

$$\begin{aligned}
&\frac{P(\mathcal{Y}\,|\,\mathcal{B}',\Gamma',\mathcal{S}',\mathcal{Z},\mathcal{L}\,|\,\mathcal{X},\mathcal{A})P(\mathcal{B}')P(\Gamma')P(\mathcal{S}')}{P(\mathcal{Y}\,|\,\mathcal{B},\Gamma,\mathcal{S},\mathcal{Z},\mathcal{L}\mathcal{X},\mathcal{A})P(\mathcal{B})P(\Gamma)P(\mathcal{S})}\\
&=\frac{\prod_{d=1}^{D}\prod_{r=1}^{A}\left[y_r^{(d)}\times\left(\sum_{t=1}^{T}\frac{\bar{N}^{(t|d)}}{\bar{N}^{(d)}}p'^{(l_t)}_{a^{(d)}r}\right)+(1-y_r^{(d)})\times\left(1-\sum_{t=1}^{T}\frac{\bar{N}^{(t|d)}}{\bar{N}^{(d)}}p'^{(l_t)}_{a^{(d)}r}\right)\right]P(\mathcal{B}')P(\Gamma')P(\mathcal{S}')}{\prod_{d=1}^{D}\prod_{r=1}^{A}\left[y_r^{(d)}\times\left(\sum_{t=1}^{T}\frac{\bar{N}^{(t|d)}}{\bar{N}^{(d)}}p^{(l_t)}_{a^{(d)}r}\right)+(1-y_r^{(d)})\times\left(1-\sum_{t=1}^{T}\frac{\bar{N}^{(t|d)}}{\bar{N}^{(d)}}p^{(l_t)}_{a^{(d)}r}\right)\right]P(\mathcal{B})P(\Gamma)P(\mathcal{S})}
\end{aligned}\tag{97}$$

8.1.1 Taking Logs

As we are going to be working in log space, we will again need to take logs of Equation 97 to perform inference.

$$\log\left[\frac{\prod_{d=1}^{D}\prod_{r=1}^{A}\left[y_r^{(d)}\times\left(\sum_{t=1}^{T}\frac{\bar{N}^{(t|d)}}{\bar{N}^{(d)}}p'^{(l_t)}_{a^{(d)}r}\right)+(1-y_r^{(d)})\times\left(1-\sum_{t=1}^{T}\frac{\bar{N}^{(t|d)}}{\bar{N}^{(d)}}p'^{(l_t)}_{a^{(d)}r}\right)\right]P(\mathcal{B}')P(\Gamma')P(\mathcal{S}')}{\prod_{d=1}^{D}\prod_{r=1}^{A}\left[y_r^{(d)}\times\left(\sum_{t=1}^{T}\frac{\bar{N}^{(t|d)}}{\bar{N}^{(d)}}p^{(l_t)}_{a^{(d)}r}\right)+(1-y_r^{(d)})\times\left(1-\sum_{t=1}^{T}\frac{\bar{N}^{(t|d)}}{\bar{N}^{(d)}}p^{(l_t)}_{a^{(d)}r}\right)\right]P(\mathcal{B})P(\Gamma)P(\mathcal{S})}\right]\tag{98}$$

We can then push the log through and reorganize terms to end up with:

$$\begin{aligned}
&=\log\left[P(\mathcal{B}')\right]+\log\left[P(\Gamma')\right]+\log\left[P(\mathcal{S}')\right]-\log\left[P(\mathcal{B})\right]-\log\left[P(\Gamma)\right]-\log\left[P(\mathcal{S})\right]+\\
&\sum_{d=1}^{D}\sum_{r=1}^{A}\log\left[y_r^{(d)}\times\left(\sum_{t=1}^{T}\frac{\bar{N}^{(t|d)}}{\bar{N}^{(d)}}p'^{(l_t)}_{a^{(d)}r}\right)+(1-y_r^{(d)})\times\left(1-\sum_{t=1}^{T}\frac{\bar{N}^{(t|d)}}{\bar{N}^{(d)}}p'^{(l_t)}_{a^{(d)}r}\right)\right]-\\
&\sum_{d=1}^{D}\sum_{r=1}^{A}\log\left[y_r^{(d)}\times\left(\sum_{t=1}^{T}\frac{\bar{N}^{(t|d)}}{\bar{N}^{(d)}}p^{(l_t)}_{a^{(d)}r}\right)+(1-y_r^{(d)})\times\left(1-\sum_{t=1}^{T}\frac{\bar{N}^{(t|d)}}{\bar{N}^{(d)}}p^{(l_t)}_{a^{(d)}r}\right)\right]
\end{aligned}\tag{99}$$

Equation 99 now gives us a way to calculate the log of the P-ratio, and so we have:

$$\log(\text{Acceptance Probability})=\min(\text{Log P-Ratio},1)\tag{100}$$

To determine whether we accept the proposed update to the interaction parameter values, we simply compare this value to the log of a draw from a uniform(0,1) distribution.

9 Getting It Right

The model is currently implemented in the CCAS R package, available on Github here: [github.com/matthewjdenny/CCAS]. To verify that this implementation is correct, we have adopted two complementary validation procedures:

1. First we reimplemented the functions described in Section 10 in R in order to check the C++ code, and wrote unit tests (which automatically run in R's testing framework) for all of them. The code for these tests is here: [CCAS/tests/testthat].

2. Second, we implemented Geweke's "Getting it Right" test (Geweke, 2004). Below, we outline our implementation of Getting it Right.

Geweke's "Getting it Right" (GiR) test can often tell the user whether something is wrong (with their code or their math), but not what the error is. Furthermore, just because one's code passes GiR, this does not guarantee that there are not errors in the code or math, only that it is pretty likely that one's generative process and inference code match up. At the highest level, GiR involves performing two sets of simulations and comparing the output of these simulatons. The first set of simulatons involves drawing N (independent) samples of data and latent variables from the model generative process and storing them. We call these samples **Forward Samples**.

Algorithm 6: `Getting_It_Right()`

Data: `model hyperparameters, number of samples`

```
# generate forward samples
for i ∈ 1:number of samples do
    # draw one sample from the generative process – document term matrix (optional), token topic assignments, topic interaction pattern assignments, LSM parameters, document edge matrix
    Take_Draw_From_Generative_Process() [Algorithms 1, 2, 3, 4]
    # calculate a bunch of statistics on the data and laten variables – means, variances, sum of LS distances, number of tokens in each topic, number of topics/tokens in each interaction pattern
    Calculate_Descriptive_Statistics()
    # save statistics
end

# generate backward samples
# initialize data
Take_Draw_From_Generative_Process() [Algorithms 1, 2, 3, 4]
for i ∈ 1:number of samples do
    # run one (or a small number) iteration of inference to update latent variables
    Inference()
    # re-draw one sample from the generative process using the latent variables we just inferred to generate new data.
    Take_Draw_From_Backward_Generative_Process() [Algorithm 5]
    # calculate a bunch of statistics on the data and laten variables
    Calculate_Descriptive_Statistics()
    # save statistics
end
# Return descriptive statistics
return (forward and backward sample statistics)
```

The second set of simulations involves iteratively generating new data from the generative process, then running the model inference code for a small number of iterations (or just one) to estimate the latent variables from the data generated by the generative process. We then use the latent variables we estimated to generate more data from the generative process. We repeat this cycle N times, saving our simulated data and latent variables at each iteration. We call these samples **Backward Samples**. This process is illustrated (at a high level) in Algorithm 6.

Finally, we calculate lots of quantities of interest on the forward and backward samples, and compare them using statistical tests and PP plots. The distributions of the statistics we calculate on these two samples should be essentially indistinguishable. While this sounds reasonably straightforward in principle, for a complex model like CCAS, we could "fail" the test simply because we have not collected enough samples, or because we have selected a parameterization of our model which is too complex (too many topics, actors, tokens, etc.) for it to sufficiently mix. We provide details for our implementation of GiR for CCAS below.

9.1 CCAS GiR Implementation Details

We implemented GiR as one of the test functions which runs automatically every time the package is rebuilt. The code to run this test is included in the `/tests/testthat/getting_it_right.R` subdirectory of the package which can be viewed **[here]**. While we tried a number of different parameter combinations in the course of testing, we outline our standard setup below.

1. We selected the following parameter values: **5 documents, 4 tokens per document, 4 topics, 5 unique word types, 4**

actors, 2 interaction patterns, 2 LSM latent dimensions, 1 edge covariate with mean 0 and standard deviation 2, uniform base measures for $\mathbf{m}$ and $\mathbf{n}$, α, $\beta = 2$, document authors assigned uniformly at random, 1 iteration of Gibbs sampling and 50 iterations of Metropolis Hastings for each backwards sampling iteration, LSM intercept, coefficient and latent position prior means of 0, with standard deviation 1 for all parameters, and a proposal variance for all LSM parameters of 0.5 (resulting in a 20% MH acceptance rate).

2. We then collected 1,000,000 forward and backward samples. This took under 5 minutes, and was sufficient to produce highly similar sets of sample statistics. Our results are not sensitive to increasing the number of iterations.

3. One additional critical component of our implementation is that it increments the seed for the inference code in backwards sampling by 100 every iteration of backwards sampling. This is necessary to ensure that inference does not produce biased results.

All of the code that implements GiR and saves the statistics for return to the user is implemented in C++ and relies on functions that have been previously unit tested. The output of our implementation of GiR is two identical R data.frame objects storing statistics calculated on each iteration of forward and backward samples. We detail the selection of statistics we save below:

1. We save the value of the LSM intercept parameter, LSM covariate coefficient, and the mean of the LSM latent positions for each interaction pattern at each iteration.

2. We also calculate the sum of distances between actors (based on their latent positions) for each interaction pattern at each iteration and save it.

3. Next we calculate the number of tokens in topics assigned to each interaction pattern and save those counts.

4. For each topic, we count the number of tokens assigned to it at each iteration and save that count as well.

5. For each unique word type, we calculate the number of tokens assigned to it at each iteration and save that count.

6. We save the mean edge value (network density) at each iteration.

7. We save the mean topic interaction pattern assignment (for interaction patterns indexed by 0 and 1) at each iteration.

Having saved a set of samples, we compared them. To do so, we generated QQ plots for each of the 21 statistics we saved. We calculated **1,000 quantiles** for each of the LSM related statistics, and **50 quantiles** for the rest of the statistics. The reason we calculate a greater number of quantiles for the LSM statistics is because they take on continuous values, while the other statistics can only take on a relatively small number of distinct values. We also calculated a t-test p-value for the equivalence of statistic means between forward and backward samples, and a Mann-Whitney test p-value for the equivalence of statistic distributions between forward and backward samples. Before we calculated these statistics, we first thinned our sample of statistics by taking every 90th sample starting at the 100,000th sample for a resulting sample size of 10,000. Our reason for doing so was to reduce the autocorrection in the Markov chain, which could deflate the p-values. More precisely, low p-values with un-thinned chains are attributable to the levels of autocorrelation in the chains. The assumption common to these tests is that observations are independent. The autocorrelation in the un-thinned chains violates this assumption, and the tests underestimate the variance in the sample, leading to Type I errors. In each case, if we observe a large p-value, this gives us evidence that the statistics have the same mean and distribution respectively. We included a diagonal line in each plot that we expect these QQ dots to line up on if we are passing GiR. The QQ-plots are depicted in Figure 2.

10 Implementation

To implement the inference procedure outlined in the previous sections, we will need to provide an overall blueprint for performing the analysis, determine a proper set of data structures, a way to break up the inference code into sub functions, and outline pseudocode for these functions.

10.1 Blueprint

The basic blueprint for the R package that implements inference in our model is that the user will read their data in to R, and then call the main estimation function (specifying the appropriate parameters such as number of Gibbs sampling iterations and LDA hyper parameters – these are discussed in greater detail below) – as illustrated in algorithm 7. Once the estimation procedure is complete, the user will then call a function that generates useable output from the model object – as illustrated in algorithm 8.
The overall structure of the core inference algorithm is illustrated in algorithm 9. This algorithm is expanded in Algorithm 22 to clarify the inference procedure.

Algorithm 7: `CCAS()`

```
Transform_Data()
Initialize_Latent_Variables()
Inference()
Update_Interaction_Patterns_To_Convergence()
Calculate_Convergence_Diagnostics()
```

return (Model Object)

Algorithm 8: `Generate_Output()`

```
Trace_Plots()
Network_Plots()
Topic_Plots()
Generate_Useful_Output()
```

return (Plots, Useful Output)

Algorithm 9: `Inference()`

for $i \in$ **1:Iterations do**

```
    Update_Token_Topic_Assignments()
    Update_Topic_Interaction_Pattern_Assignments()
    Update_LDA_Hyperparameters()
    Adaptive_Metropolis()
    Update_Interaction_Pattern_Parameters()
```

end

return (Samples)

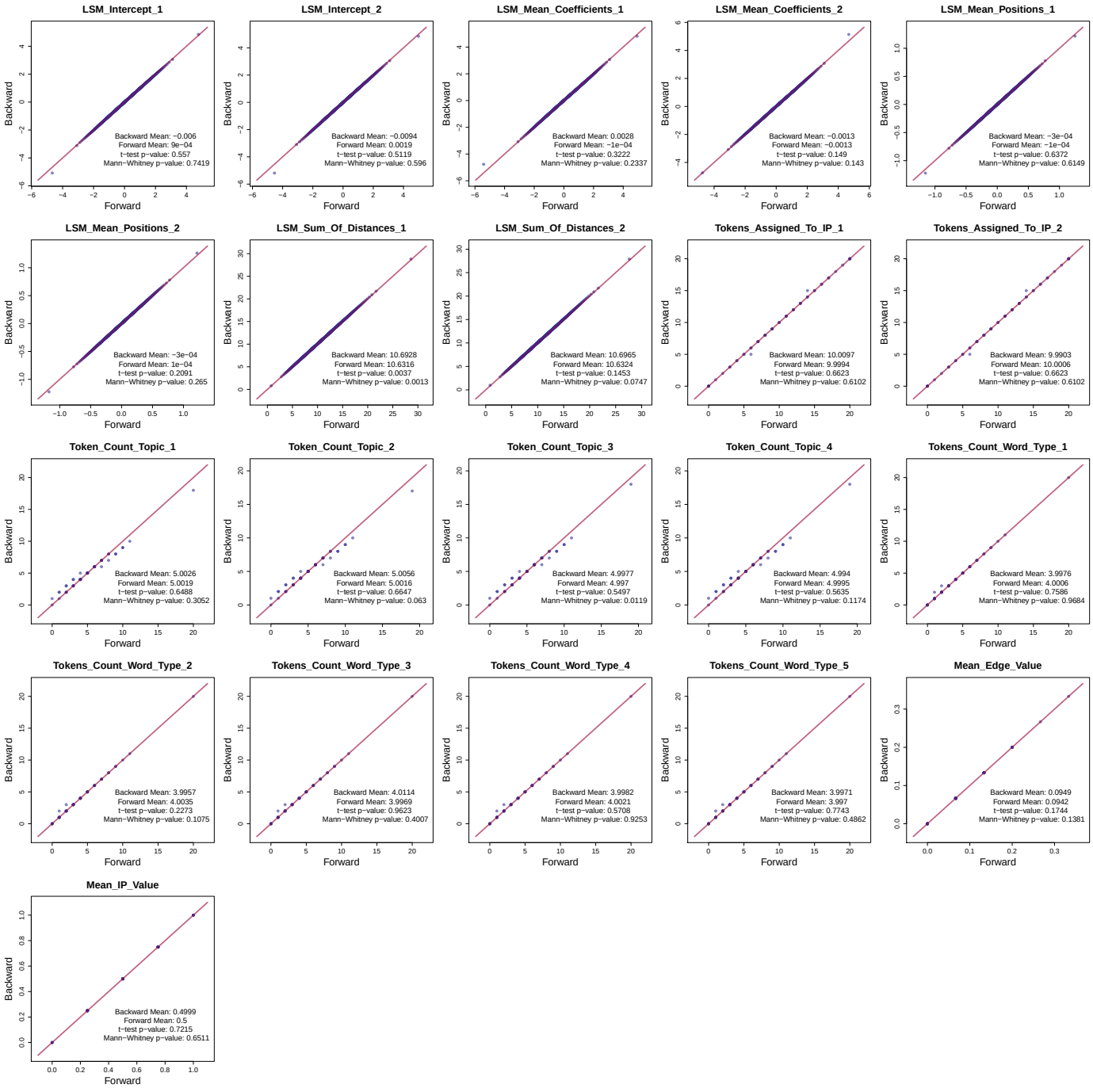


Figure 2: QQ-plot with 1,000 quantile points displayed for each of the LSM related statistics, and 50 quantile points for the rest of the statistics, calculated on 1,000,000 samples of forward and backward samples. Note that many of these points will overlap.

10.2 Data Structures

When we are working in R, I plan to store all relevant data in a object of class CCAS (or whatever our acronym is) that will have a bunch of appropriate fields for all of the sub data structures and will do a bunch of automatic checks to ensure the data is in the right form. These are just the major data structures, there will be many more intermediate ones which I will define in the appropriate functions.

10.2.1 Observed Data Structures

1. `author_indexes` – $D \times 1$ vector of document author indexes.
2. `document_edge_matrix` – $D \times A$ document-recipient indicator matrix.
3. `token_word_types` A D-dimensional list of $N^{(d)}$ dimensional vectors of token word-type index vectors. Can be provided by the user as a document term matrix.
4. `covariates` $A \times A \times |\gamma|$ array of sender-recipient dyad specific fixed covariates.

10.2.2 Latent Variable Data Structures

1. `topic_interaction_patterns` – $T \times 1$ vector of topic-interaction pattern assignments.
2. `document_topic_counts` – $D \times T$ matrix of document-topic counts.
3. `word_type_topic_counts` – $V \times T$ matrix of word-type topic counts.
4. `topic_token_counts` – $T \times 1$ vector of topic-token counts.
5. `token_topic_assignments` – D-dimensional list of $N^{(d)}$ dimensional vectors of token-topic assignment vectors.
6. `token_word_types` – D-dimensional list of $N^{(d)}$ dimensional vectors of token-word-type vectors.
7. `intercepts` – $C \times 1$ vector containing interaction pattern-specific intercept parameters.
8. `coefficients` – $C \times |\gamma|$ matrix containing interaction pattern-specific coefficients.
9. `latent_positions` – $C \times A \times |s|$ array containing interaction pattern-specific latent positions for actors.

10.3 Functions

10.3.1 User-Facing Functions

1. `CCAS()` – Main function that performs inference in our model.
2. `Run_Latent_Space_Model()` – Run a vanilla latent space model (we get this basically for free and it will be useful for comparison, especially since we can make it easier to specify the prior that in the latentnet implementation)
3. `Generate_Output()` – Generate a whole bunch of diagnostic plots (everything we need to put in our paper) and output in a useful format.
4. `Validate_Topics()` – Calculates topic coherence, exclusivity, etc. and outputs top emails in each topic for hand coding.
5. `Descriptive_Statistics()` – gives the user some descriptive statistics about their dataset.
6. `Prepare_Data()` – Could eventually incorporate functionality to parse emails and clean text before analysis. This will be a lower priority.
7. `TPME()` – Runs the TPME model as a special case of ours. Lower priority.

10.3.2 Internal R Functions

1. `Transform_Data()` – Transform data from input format to the appropriate format for inference.
2. `Initialize_Latent_Variables()` – Initialize all latent variable values in R (it is much easier to sample the values in R than C++).
3. `Calculate_Convergence_Diagnostics()` – Calculates Geweke convergence diagnostics on all latent variables as well as un-normalized topic model log likelihood and reports/stores them.
4. `Trace_Plots()` – Creates trace plots of un-normalized topic model log likelihood, LSM log likelihoods as well as mixing parameters and intercepts (and perhaps some latent positions).

5. `Network_Plots()` – Generates network plots similar to the ones we already have.
6. `Topic_Plots()` – Plots and tables of top words like those in our current paper.
7. `Generate_Useful_Output()` – Generate output from model in a form that will make it easy to do downstream analysis. We already do this in the current implementation.

10.3.3 Inference Functions (Written in C++)

1. `Inference()` – Main C++ function to perform inference in our model.
2. `Update_Interaction_Patterns_To_Convergence()` – Final run of Metropolis Hastings until interaction pattern parameter estimates converge.
3. `Update_Token_Topic_Assignments()` – Perfroms a single iteration of token topic assignment updates. See algorithm 18.
4. `Update_Interaction_Pattern_Parameters()` – Performs some number of iterations of updates to interaction pattern parameters via Metropolis Hastings. See algorithm 19
5. `Sample_New_Interaction_Pattern_Parameters()` – Samples new parameters centered at the current values. See algorithm 14
6. `Update_Topic_Interaction_Pattern_Assignments()` – Updates interaction pattern assignments for each topic. See algorithm 20
7. `Update_LDA_Hyperparameters()` – can be turned on to update α and β, but also serves to calculate the un-normalized topic model log likelihood which can be used to asses model convergence.
8. `Adaptive_Metropolis()` – Update proposal variances based on Metropolis Hastings acceptance rate from previous round.
9. `Update_Single_Token_Topic_Assignment()` – Resamples topic assignment for a single token. See algorithm 17.
10. `LDA_Contribution()` – calculates the LDA contribution to equation 67 (the first two terms). See algorithm 16.
11. `LSM_Contribution()` – calculates the LSM contribution to equation 67 (the product over A terms). See algorithm 15.
12. `Edge_Probability()` – calculates $p^{(c)}_{a^{(d)}r}$ for a given interaction pattern c. See algorithm 11.
13. `Prior_Probability_Interaction_Pattern_Parameters()` – calculates $P(\mathcal{B})P(\Gamma)P(\mathcal{S})$ for given interaction pattern parameters (current or proposed) and prior means and variances. See algorithm 13.
14. `Sum_Over_T_Edge_Probability()` – Calculates:

$$\sum_{t=1}^{T} \frac{*}{\bar{N}^{(d)}} p^{(l_t)}_{a^{(d)}r} \tag{101}$$

where $*$ can be $\bar{N}^{(t|d)}$ or $\bar{N}^{(t|d)}_{\backslash d,n}$ and thes quantities are passed in as a vector along with a vector of $p^{(c)}_{a^{(d)}r}$ for all c (in order to speed up computation). See algorithm 12.

15. `Log_Space_Multinomial_Sampler()` – makes resampling topics in log space easy. See algorithm 10.

10.4 Pseudocode

Here I am going to focus primarily on the inference code as opposed to the supporting R functions, as these are much more straightforward.

Algorithm 10: `Log_Space_Multinomial_Sampler()`

Data: `log_probabilities`

```
# Do the log-sum-exp trick for the whole distribution
m = max(log_probabilities)
log_sum_exp_dist = m + log(sum(log_probabilities - m))
# set the current total that we will increment in order to sample
total = -Inf  # set our target value, which we will stop when we reach
uniform_draw = uniform_distribution_0_1(RNG)
target = log(uniform_draw) + log_sum_exp_dist

for i ∈ 1:Length of Vector do
    cur_max = max(total,log_probabilities[i])
    total = cur_max + log(sum(exp(log_probabilities[i] - cur_max), exp(total - cur_max)))
    if total > target then
        draw = i
        return (draw)
    end
end
```

Algorithm 11: `Edge_Probability()`

Data: `intercepts`,
`coefficients`,
`latent_positions`,
`sender`,
`recipient`,
`current_covariates = covariates[sender, recipient,]`,
`interaction_pattern`,
`using_coefficients`

```
# Linear predictor = intercept + coefficients - latent distances
linear_predictor = intercepts[interaction_pattern]

# Only include coefficients if they were specified.
if using_coefficients then
    for i ∈ 1:Number of Coefficients do
        linear_predictor += coefficients[interaction_pattern,i] × current_covariates[i]
    end
end

for i ∈ 1:Number of Latent Dimensions do
    linear_predictor -= |latent_positions[cluster,sender,i] - latent_positions[cluster,recipient,i]|
end

# Run the linear predictor through the logistic sigmoid function
edge_prob = 1/(1+ exp(-linear_predictor))
return (edge_prob)
```

Algorithm 12: `Sum_Over_T_Edge_Probability()` – Since the current edge probabilities (across different interaction patterns) stay the same in this step, we can pass them in as an array.

```
Data: edge_probabilities, # An array of pre-calculated edge probabilities
      tokens_in_document,  # A fixed count
      current_token_topic_assignment, # an integer index
      current_document_topic_counts = document_topic_counts[document,],
      leave_out_current_token,
      topic_interaction_patterns,
      document_sender,
      document_recipient,
      leave_out_topic

# Initialize sum to zero.
sum = 0

# decrement doc-topic counts if we are leaving the current token out.
if leave_out_current_token then
    current_document_topic_counts[current_token_topic_assignment] -= 1
end

# leave out topic can default to -1 in which case we do not leave out any topics. This outer conditional makes things run faster for most cases.
if leave_out_topic > -1 then
    for t ∈ 1:Number of Topics do
        if leave_out_topic != t then
            sum += (current_document_topic_counts[t]/tokens_in_document) ×
            edge_probabilities[document_sender, document_recipient, topic_interaction_patterns[t]]
        end
    end
end
# Otherwise we do not have to check the conditional
else
    for t ∈ 1:Number of Topics do
        sum += (current_document_topic_counts[t]/tokens_in_document) × edge_probabilities[document_sender,
        document_recipient, topic_interaction_patterns[t]]
    end
end
return (sum)
```

Algorithm 13: `Prior_Probability_Interaction_Pattern_Parameters()` – We want to work all in log space, so just return the log prior probability. We will use the normal pdf from Rmath.c which is very fast an accurate to 18 decimal places.

Data: `intercepts`,
`coefficients`,
`latent_positions`,
`intercept_prior_mean`,
`intercept_prior_variance`,
`coefficient_prior_mean`,
`coefficient_prior_variance`,
`latent_position_prior_mean`,
`latent_position_prior_variance`,
`using_coefficients`

log_prior_probability = 0

for $i \in$ **1:Number of Interaction Patterns do**
  log_prior_probability += log(normal_pdf(`intercepts[i]; intercept_prior_mean, intercept_prior_variance`))
end

if `using_coefficients` **then**
  for $i \in$ **1:Number of Interaction Patterns do**
    for $j \in$ **1:Number of Coefficients do**
      log_prior_probability += log(normal_pdf(`coefficients[i,j]; coefficient_prior_mean, coefficient_prior_variance`))
    end
  end
end

for $i \in$ **1:Number of Interaction Patterns do**
  for $j \in$ **1:Number of Actors do**
    for $k \in$ **1:Number of Latent Dimensions do**
      log_prior_probability += log(normal_pdf(`latent_positions[i,j,k]; latent_position_prior_mean, latent_position_prior_variance`))
    end
  end
end
return (log_prior_probability)

Algorithm 14: `Sample_New_Interaction_Pattern_Parameters()` – sample proposed positions from independent normals centered at their current values and returns proposed interaction pattern parameters in an Rcpp List object.

```
Data: intercepts,
      coefficients,
      latent_positions,
      intercept_proposal_variances,  # allows for different values for each interaction pattern
      coefficient_proposal_variances,  # allows for different values for each interaction pattern
      latent_position_proposal_variances,  # allows for different values for each interaction pattern
      using_coefficients

# A vector of zeros
proposed_intercepts = zeros(number_of_interaction_patterns)
# A matrix of zeros
proposed_coefficients = zeros(number_of_interaction_patterns, number_of_coefficients)
# An array of zeros
proposed_latent_positions = zeros(number_of_interaction_patterns, number_of_actors,
number_of_latent_dimensions)

for i ∈ 1:Number of Interaction Patterns do
  proposed_intercepts[i] = normal_distribution(intercepts[i], intercept_proposal_variances[i])
end

if using_coefficients then
  for i ∈ 1:Number of Interaction Patterns do
    for j ∈ 1:Number of Coefficients do
      proposed_coefficients[i,j] = normal_distribution(coefficients[i,j],
      coefficient_proposal_variances[i])
    end
  end
end

for i ∈ 1:Number of Interaction Patterns do
  for j ∈ 1:Number of Actors do
    for k ∈ 1:Number of Latent Dimensions do
      proposed_latent_positions[i,j,k] = normal_distribution(latent_positions[i,j,k],
      latent_position_proposal_variances[i])
    end
  end
end
# Return an Rcpp list object containing the new values
return (proposed_intercepts, proposed_coefficients, proposed_latent_positions)
```

Algorithm 15: `LSM_Contribution()` – we are going to work in log space. This calculates the stuff in the sum in equation 73.

Data: `edge_probabilities`,
`tokens_in_document`,
`topic`,
`current_token_topic_assignment`,
`current_document_topic_counts = document_topic_counts[document,]`,
`document_edge_values = document_edge_matrix[document,]`,
`topic_interaction_patterns`,
`document_sender`,
`current_topic`

value = 0

for $a \in$ **1:Number of Actors do**
  if `a` $\neq$ `document_sender` **then**
    # If there is a present edge in the current document
    if `document_edge_values[a] == 1` **then**
      value +=
      log[`Sum_Over_T_Edge_Probability(leave_out_current_token = TRUE, leave_out_topic = -1)` +
      `tokens_in_document`$^{-1}$ $\times$ `edge_probabilities[document_sender, a, topic_interaction_patterns[topic]]`
    end
    # If there is an absent edge in the current document
    else
      value += log[(1- `Sum_Over_T_Edge_Probability(leave_out_current_token = TRUE, leave_out_topic = -1)`) + (1- `tokens_in_document`$^{-1}$ $\times$ `edge_probabilities[document_sender, a, topic_interaction_patterns[topic])`]
    end
  end
end
return (value)

Algorithm 16: `LDA_Contribution()` – we are going to work in log space to prevent underflow

Data: `tokens_in_document`,
`current_token_topic_assignment`,
`current_document_topic_counts = document_topic_counts[document,]`,
`word_type_topic_counts`,
`topic_token_counts`,
`topic`,
`current_word_type`,
$\alpha\mathbf{m}$,
$\beta\mathbf{n}$,
β,
`current_dtc = document_topic_counts[document,topic]`

wttc = `word_type_topic_counts[current_word_type,topic]`
tc = `topic_token_counts[topic]`
if `current_token_topic_assignment == topic` **then**
  `current_dtc` -= 1
  wttc -= 1
  tc -= 1
end
value = log(`current_dtc` + $\alpha\mathbf{m}$[`topic`]) + log(wttc + $\beta\mathbf{n}$[`current_word_type`]) - log(tc + β)

return (value)

Algorithm 17: `Update_Single_Token_Topic_Assignment()`

Data: `edge_probabilities`,
`tokens_in_document`,
`current_token_topic_assignment`,
`current_document_topic_counts = document_topic_counts[document,]`,
`word_type_topic_counts`,
`topic_token_counts`,
`current_word_type`,
$\alpha\mathbf{m}$,
$\beta\mathbf{n}$,
`document_edge_values = document_edge_matrix[document,]`,
`topic_interaction_patterns`,
`document_sender`,
`current_token_topic_assignment`,
`random_number`

# calculate beta, the sum over beta_n
beta = sum(beta_n)
# get the number of topics
number_of_topics = alpha_m.n_elem
# Create a blank distribution.
unnormalized_log_probabilities = zeros(Number of Topics)

# loop over topics and populate with unnormalized log probabilities.
for $t \in$ **1:Number of Topics do**
  unnormalized_log_probabilities[t] = `LDA_Contribution()` + `LSM_Contribution()`
end
# sample a new token-topic assignment from this distribution.
new_assignment = `Log_Space_Multinomial_Sampler(unnormalized_log_probabilities, random_number)`
return (new_assignment)

Algorithm 18: `Update_Token_Topic_Assignments()`

Data: `author_indexes`,
`document_edge_matrix`,
`topic_interaction_patterns`,
`document_topic_counts`,
`word_type_topic_counts`,
`topic_token_counts`,
`token_topic_assignments`,
`token_word_types`,
`intercepts`,
`coefficients`,
`latent_positions`,
`covariates`,
$\alpha\mathbf{m}$,
$\beta\mathbf{n}$,
`random_numbers`,
`using_coefficients`

```
# pre-calculate all edge probabilities in all interaction patterns. Eventually we will want to pass in the values from previous iteration so (they will come from updating interaction pattern parameters), so we will only ever have to do this once. Lets start the slower way for now.
edge_probabilities = zeros(number_of_actors,number_of_actors,number_of_interaction_patterns)
for a ∈ 1:number_of_actors do
    for r ∈ 1:number_of_actors do
        if a != r then
            for i ∈ 1:number_of_interaction_patterns do
                edge_probabilities[a,r,i] = Edge_Probability()
            end
        end
    end
end

# Now update token topic assignments
for d ∈ 1:Number of Documents do
    for i ∈ 1:Number of Tokens in Current Document do
        current_topic_assignment = token_topic_assignments[[d]][i]
        new_topic_assignment = Update_Single_Token_Topic_Assignment()
        # if we did end up changing the token-topic assignment, then we need to update all of the data structures that keep track of counts to reflect this
        if current_topic_assignment != new_topic_assignment then
            document_topic_counts[current_topic_assignment] -= 1
            document_topic_counts[new_topic_assignment] += 1
            token_topic_assignments[[d]][i] = new_topic_assignment
            topic_token_counts[current_topic_assignment] -= 1
            topic_token_counts[new_topic_assignment] += 1
            word_type_topic_counts[token_word_types[[d]][i], current_topic_assignment] -= 1
            word_type_topic_counts[token_word_types[[d]][i], new_topic_assignment] += 1
        end
    end
end

# We return edge probabilities so we can use them to save computation time in updating interaction pattern parameters
return (document_topic_counts, word_type_topic_counts, topic_token_counts, token_topic_assignments, edge_probabilities)
```

Algorithm 19: `Update_Interaction_Pattern_Parameters()`

Data: `author_indexes`,
`edge_probabilities`,
`document_edge_matrix`,
`document_topic_counts`,
`intercepts`,
`coefficients`,
`latent_positions`,
`intercept_prior_mean`,
`intercept_prior_variance`,
`intercept_proposal_variances`, # allows for different values for each interaction pattern
`coefficient_prior_mean`,
`coefficient_prior_variance`,
`coefficient_proposal_variances`, # allows for different values for each interaction pattern
`latent_position_prior_mean`,
`latent_position_prior_variance`,
`latent_position_proposal_variances`, # allows for different values for each interaction pattern
`random_numbers`,
`edge_probabilities`,
`using_coefficients`

proposed params = `Sample_New_Interaction_Pattern_Parameters()`
log_current_prior = `Prior_Probability_Interaction_Pattern_Parameters(current params)`
log_proposed_prior = `Prior_Probability_Interaction_Pattern_Parameters(proposed params)`
log_current_prob = log_proposed_prob = 0
# pre-calculate proposed edge probabilities in all interaction patterns.
`proposed_edge_probabilities` = zeros(number_of_actors,number_of_actors,number_of_interaction_patterns)
for $a \in$ **1:number_of_actors do**
  for $r \in$ **1:number_of_actors do**
    if `a != r` **then**
      for $i \in$ **1:number_of_interaction_patterns do**
        `proposed_edge_probabilities[a,r,i]` = `Edge_Probability()`
      end
    end
  end
end
for $d \in$ **1:Number of Documents do**
  for $a \in$ **1:Number of Actors do**
    if `a` $\neq$ `document_sender` **then**
      if `document_edge_matrix[d,a] == 1` **then**
        log_current_prob += log[`Sum_Over_T_Edge_Probability(current params)`]
        log_proposed_prob += log[`Sum_Over_T_Edge_Probability(proposed params)`]
      end
      else
        log_current_prob += log[1- `Sum_Over_T_Edge_Probability(current params)`]
        log_proposed_prob += log[1- `Sum_Over_T_Edge_Probability(proposed params)`]
      end
    end
  end
end
accept_log_prob = log_proposed_prior + log_proposed_prob - log_current_prior - log_current_prob
if `accept_log_prob > log(uniform_0_1_draw())` **then**
  return (`proposed_intercepts`, `proposed_coefficients`, `proposed_latent_positions`, `proposed_edge_probabilities`, accepted_proposal = TRUE)
end
else
  return (`intercepts`, `coefficients`, `latent_positions`, `edge_probabilities`, accepted_proposal = FALSE)
end

Algorithm 20: `Update_Topic_Interaction_Pattern_Assignments()`

```
Data: author_indexes,
      document_edge_matrix,
      document_topic_counts,
      intercepts,
      coefficients,
      latent_positions,
      covariates,
      topic_interaction_patterns,
      using_coefficients,
      edge_probabilities,
      random_numbers

for t ∈ 1:Number of Topics do
    interaction_pattern_assignment_prob = zeros(Number of interaction patterns)
    # pre calculate sum over t' terms holding out t and store in a matrix
    held_out_sum_over_t_terms = zeros(number of documents, number of actors)
    for d ∈ 1:Number of Documents do
        for a ∈ 1:Number of Actors do
            if a ≠ document_sender then
                if document_edge_matrix[d,a] == 1 then
                    held_out_sum_over_t_terms[d,a] =
                    Sum_Over_T_Edge_Probability(leave_out_current_token = FALSE, leave_out_topic = t)
                end
                else
                    held_out_sum_over_t_terms[d,a] = 1-
                    Sum_Over_T_Edge_Probability(leave_out_current_token = FALSE, leave_out_topic = t)
                end
            end
        end
    end
    for c ∈ 1:Interaction Patterns do
        log_prob = 0
        for d ∈ 1:Number of Documents do
            for a ∈ 1:Number of Actors do
                if a ≠ document_sender then
                    if document_edge_matrix[d,a] == 1 then
                        log_prob += log[held_out_sum_over_t_terms[d,a] +
                        (document_topic_counts[document,t]/tokens_in_document) ×
                        edge_probabilities[author_indexes[d], a, c ]
                    end
                    else
                        log_prob += log[held_out_sum_over_t_terms[d,a] + 1-
                        (document_topic_counts[document,t]/tokens_in_document) ×
                        edge_probabilities[author_indexes[d], a, c ]
                    end
                end
            end
        end
        interaction_pattern_assignment_log_prob[c] = log_prob
    end
    topic_interaction_patterns[t] =
    Log_Space_Multinomial_Sampler(interaction_pattern_assignment_log_prob)
end
return (topic_interaction_patterns)
```

Algorithm 21: `Adaptive_Metropolis()`

```
Data: intercept_proposal_variances, # allows for different values for each interaction pattern
      coefficient_proposal_variances, # allows for different values for each interaction pattern
      latent_position_proposal_variances, # allows for different values for each interaction pattern
      accept_rates,
      target_accept_rate,
      tollerance,
      update_size

# In our first iteration of this code, we are going to have a common proposal variance across all LSM parameters.
Eventually we can make all of them different if we so choose. For now, we will update them together.
for i ∈ 1:Number of Interaction Patterns do
    if accept_rates[i] < target_accept_rate - tollerance then
        if proposal_variances > update_size then
            intercept_proposal_variances[i] -= update_size
            coefficient_proposal_variances[i] -= update_size
            latent_position_proposal_variances[i] -= update_size
        end
    end
    if accept_rates[i] > target_accept_rate + tollerance then
        intercept_proposal_variances[i] += update_size
        coefficient_proposal_variances[i] += update_size
        latent_position_proposal_variances[i] += update_size
    end
end
return (proposal_variances)
```

References

Geweke, John. Getting It Right: Joint Distribution Tests of Posterior Simulators. *Journal of the American Statistical Association*, 99(467):799–804, September 2004. http://www.tandfonline.com/doi/abs/10.1198/016214504000001132.

Griffiths, Tom. Gibbs samplng in the generative model of Latent Dirichlet Allocation. 2002.

Hoff, Peter D, Adrian E Raftery, and Mark S Handcock. Latent Space Approaches to Social Network Analysis. *Journal of the American Statistical Association*, 97(460):1090–1098, December 2002. http://www.tandfonline.com/doi/abs/10.1198/016214502388618906.

Algorithm 22: `Inference()` – This algorithm expands and clarifies Algorithm 9 for actual implementation.

Data: `author_indexes`,
`document_edge_matrix`,
`document_topic_counts`,
`topic_interaction_patterns`,
`word_type_topic_counts`,
`topic_token_counts`,
`token_topic_assignments`,
`token_word_types`,
`intercepts`,
`coefficients`,
`latent_positions`,
`covariates`,
$\alpha\mathbf{m}$,
$\beta\mathbf{n}$,
`using_coefficients`,
`intercept_prior_mean`,
`intercept_prior_variance`,
`intercept_proposal_variances`,
`coefficient_prior_mean`,
`coefficient_prior_variance`,
`coefficient_proposal_variances`,
`latent_position_prior_mean`,
`latent_position_prior_variance`,
`latent_position_proposal_variances`,
`target_accept_rate`,
`tollerance`,
`update_size`
`seed`,
`iterations`,
`metropolis_iterations`,
`total_number_of_tokens`,
`iterations_before_t_i_p_updates`,
`update_t_i_p_every_x_iterations`,
`perform_adaptive_metropolis`

```
# Initialize the Random Number Generator
# Initialize a bunch of global variables to be used by sub-functions
# Initialized data structures to store samples in
```

Algorithm 23: Continuation of `Inference()` algorithm.

```
for i ∈ 1:Iterations do
    Update_Token_Topic_Assignments()
    Update_Topic_Interaction_Pattern_Assignments()
    Update_LDA_Hyperparameters()
    Adaptive_Metropolis()
    for j ∈ 1:Metropolis Iterations do
        Update_Interaction_Pattern_Parameters()
    end
end
return (Samples)
```