Gwangbin Bae gb585@cam.ac.uk

4F13: Probabilistic Machine Learning
Coursework #3: Latent Dirichlet Allocation

1. Question A

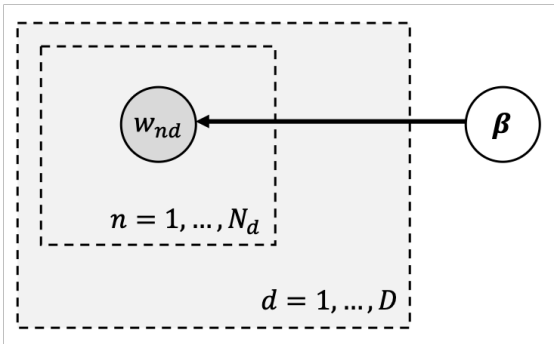

Figure 1 *Graphical representation of a simple multinomial model*

In multinomial model, the likelihood of a sequence can be represented as following.

$$p(\mathbf{w}|\boldsymbol{\beta}) = \prod_{m=1}^{M} \beta_m^{c_m} \quad \rightarrow \quad \log p(\mathbf{w}|\boldsymbol{\beta}) = \sum_{m=1}^{M} c_m \log \beta_m \tag{1}$$

In such model, the maximum likelihood estimate for $\boldsymbol{\beta}$ is given as following.

$$\hat{\beta}_j = \frac{c_j}{N} = \frac{c_j}{\sum_{m=1}^{M} c_m} \tag{2}$$

Command 1 *Maximum likelihood multinomial over the words in training data A*

```
beta(I) = count_i/total_count; % repeat for I = 1:W
[B,I] = sort(beta,'descend');
B = B(1:20); I = I(1:20); names = V(I);
```

Command 1 computes the maximum likelihood estimate for $\boldsymbol{\beta}$. The histogram in **Figure 2** shows the 20 words with the largest probabilities.

The maximum likelihood estimate can be used to compute the likelihood of a test document. Whenever a word j is observed, $\hat{\beta}_j$ is multiplied. If a new word is observed, corresponding $\hat{\beta}_j$ will be zero since c_j was zero for training data. Hence, the test set probability becomes zero. (The log probability becomes negative infinity.)

A possible way to solve such problem is to inflate the counts with pseudo-counts and renormalize. By doing so, the words that were not observed in the training data can earn some probability.

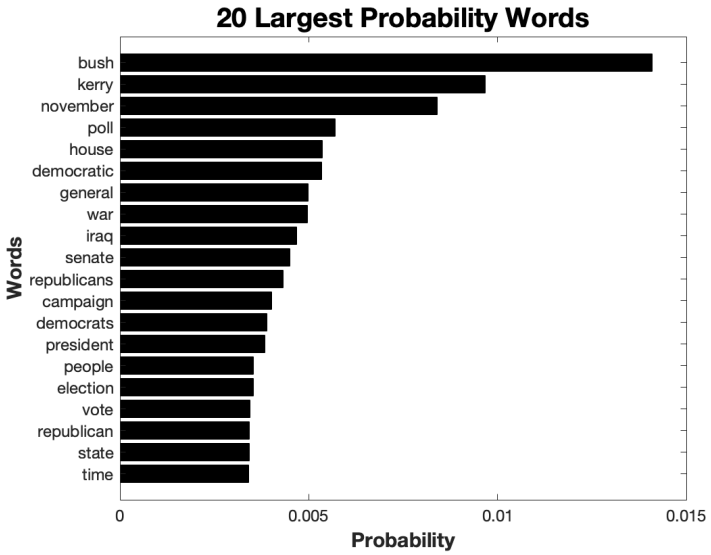

Figure 2 *20 largest probability words, based on simple multinomial model*

2. Question B

The Bayesian inference method incorporates the prior knowledge on the parameter and computes its posterior distribution. A possible choice of prior can be a symmetric Dirichlet prior with $\alpha = 0.1$:

$$p(\boldsymbol{\beta}) = \mathrm{Dir}(\boldsymbol{\beta}|\boldsymbol{\alpha}) = \frac{1}{B(\boldsymbol{\alpha})}\prod_{m=1}^{M}\beta_m^{\alpha-1} \tag{3}$$

Then, the posterior distribution can be obtained by multiplying the prior to the likelihood.

$$\begin{aligned} p(\boldsymbol{\beta}|w_A) &\propto p(w_A|\boldsymbol{\beta})p(\boldsymbol{\beta}) \\ &\propto \prod_{m=1}^{M}\beta_m^{c_m}\prod_{m=1}^{M}\beta_m^{\alpha-1} = \prod_{m=1}^{M}\beta_m^{c_m+\alpha-1} \end{aligned} \tag{4}$$

$$\therefore\ p(\boldsymbol{\beta}|w_A) = \mathrm{Dir}(\boldsymbol{\beta}|c_1+\alpha,\dots,c_M+\alpha)$$

It can be seen that the posterior distribution is another Dirichlet distribution. (i.e. Dirichlet distribution is conjugate prior for multinomial likelihood.) Accordingly, the posterior predictive distribution can be given as following.

$$\begin{aligned} p(X=j|w_A) &= \int p(X=j|\boldsymbol{\beta})p(\boldsymbol{\beta}|w_A)d\boldsymbol{\beta} \\ &= \frac{\alpha+c_j}{\sum_{m=1}^{M}(\alpha+c_m)} = \frac{\alpha+c_j}{M\alpha+N} \end{aligned} \tag{5}$$

This shows that incorporating a symmetric Dirichlet prior is equivalent to adding a pseudo-count α to every word in the dictionary. The maximum likelihood estimate $\beta_j = c_j/N$ is recovered for $\alpha = 0$.

Adding a pseudo-count increases the probabilities of rare words, and reduces those of commons words. Such give-and-take must occur in order to normalize β_j. More specifically, the word earns probability if $c_j < N/M$, and loses if $c_j > N/M$.

3. Question C

The probability of a sequence of independent words can be computed by multiplying the probability of each word.

$$p(\mathbf{w}|\boldsymbol{\beta}) = \prod_{m=1}^{M} \beta_m^{c_m} \tag{6}$$

In order to obtain the probability of word counts, a combinatorial factor should be multiplied.

$$p(c_1, \dots, c_m|\boldsymbol{\beta}) = \frac{M!}{\prod_{m=1}^{M} c_m!} \prod_{m=1}^{M} \beta_m^{c_m} \tag{7}$$

The combinatorial factor represents the number of sequences that are consistent with the designated word counts. In document modelling, such factor is unnecessary because the documents are already arranged in a specific order. Although the given dataset does not provide information on the ordering, the word counts are sufficient statistics for computing the sequence probability.

Then, the log probability and perplexity of the document 2001 can be expressed as following. ($c_m^{(A)}$: word counts in data A, $c_m^{(w_{2001})}$: word counts in document 2001)

$$\log p(\boldsymbol{w}_{2001}|\boldsymbol{w}_A, \alpha) = \sum_{m=1}^{M} c_m^{(w_{2001})} \log\left(\frac{\alpha + c_m^{(A)}}{M\alpha + N}\right) \tag{8}$$

$$p_{2001} = \exp\left(-\frac{\log p(\boldsymbol{w}_{2001}|\boldsymbol{w}_A, \alpha)}{\sum_{m=1}^{M} c_m^{(w_{2001})}}\right) \tag{9}$$

Command 2 *Calculating the log probability and perplexity of document 2001*

```
for m = 1:M
    count_m = sum(doc_2001(doc_2001(:,2)==m,3));    % Count for doc 2001
    total_count = total_count + count_m;            % Update total count
    log_prob = log_prob + count_m * log(beta(m));   % Update log probability
end
perplexity = exp(-log_prob/total_count);            % Calculate perplexity
```

Table 1 *Log probability and per-word perplexity, calculated for document 2001*

Document ID: 2001	***Log Probability***	***Number of Words***	***Per-word Perplexity***
Values	−3691	440	4399

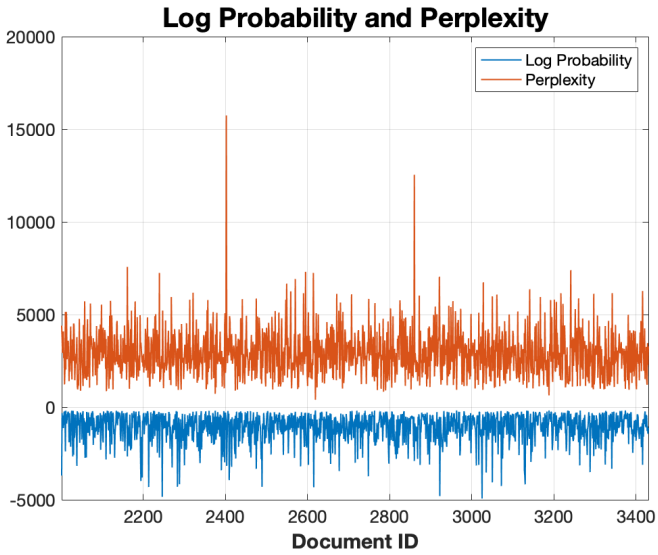

Figure 3 *Log probability and per-word perplexity calculated for every document in dataset B*

The perplexity can be interpreted as the average uncertainty associated with each observation. By rewriting the definition, it can be shown that the perplexity calculates the average of the inverse probability in log domain.

$$p = \exp\left(\frac{1}{N}\left(\sum_{n=1}^{N} \log \frac{1}{p(w_n|\mathcal{D})}\right)\right) \tag{10}$$

Therefore, if a document consists of common words, the perplexity would be small. Similarly, the perplexity increases for rare words, and becomes infinite if there is an impossible word. This explains why the perplexity of different documents are different, as seen in **Figure 3**.

For uniform multinomial, every word will share the probability of $1/M$. Hence, the perplexity will be $\exp(\log M) = M = 6906$. Such model is analogous to rolling a dice with 6906 sides.

4. Question D

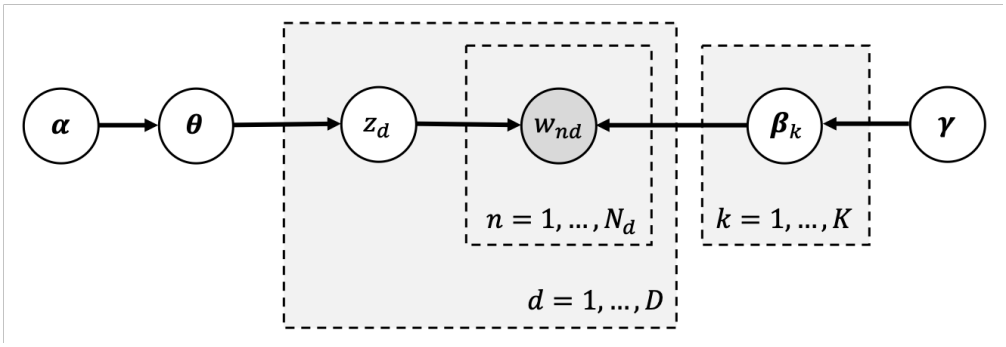

Figure 4 *Graphical representation of Bayesian mixture model*

In the Bayesian mixture model, the latent topic assignments can be sampled using a collapsed Gibbs sampler. The latent distribution is dependent on the likelihood given topic, and the word counts of other documents. (Such dependency results from the collapsing of $\boldsymbol{\theta}$)

$$p(z_n = k|\boldsymbol{w}_d, \boldsymbol{z}_{-n}, \boldsymbol{\beta}, \alpha) \propto p(\boldsymbol{w}_d|\beta_k)\, p(z_n = k|\boldsymbol{z}_{-n}, \alpha)$$
$$= p(\boldsymbol{w}_d|\beta_k) \frac{c_{-d,k} + \alpha}{\sum_{j=1}^{K}(c_{-d,j} + \alpha)} \tag{11}$$

At each iteration, the mixing proportion can be computed simply from the empirical counts, inflated with pseudo-counts.

$$\theta_k = \frac{c_k + \alpha}{\sum_{j=1}^{K}(c_j + \alpha)} \tag{12}$$

Command 3 *Calculating mixture proportions in Bayesian mixture model*

```
theta(:,iter) = (sk_docs(:) + alpha) / sum(sk_docs(:) + alpha);
```

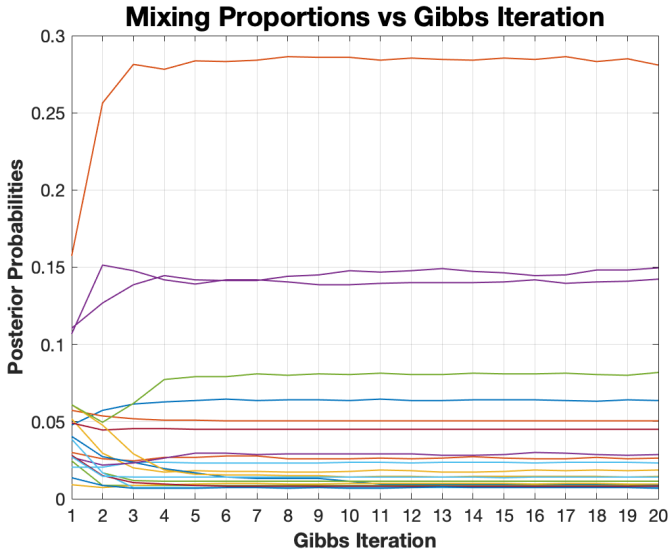

Figure 5 *Evolution of mixing proportions as a function of Gibbs iteration*

Figure 5 shows that it requires few iterations for the mixing proportions to reach certain values. Then, the values fluctuate, suggesting that the sampler is moving inside the posterior. However, it is unclear if the sampler is exploring the entire posterior.

The convergence of a Gibbs sampler is achieved when the samples become representative of the underlying posterior distribution. Since the underlying distribution is independent of the initial condition, the samples, after 'burn-in' and 'thinning', should behave similarly regardless of the random seed.

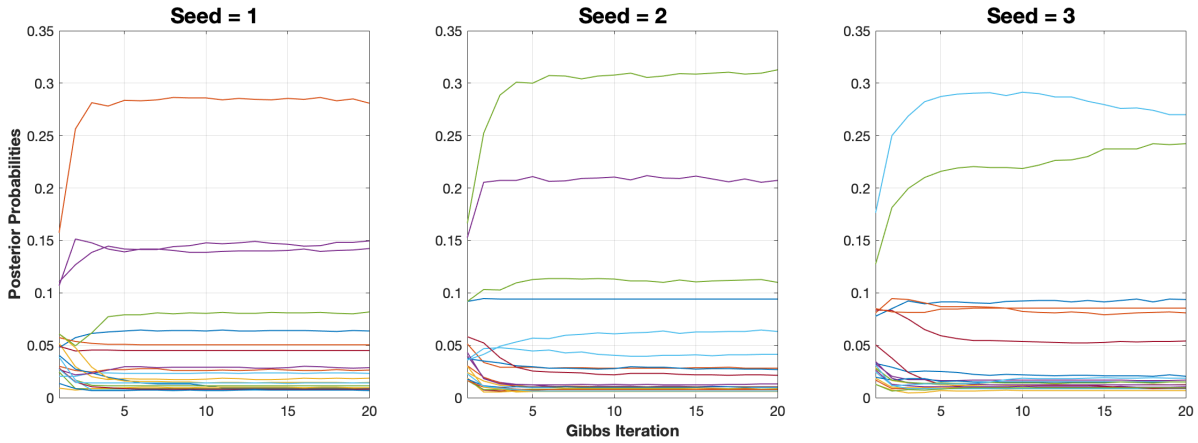

Figure 6 *Evolution of the mixing proportions obtained using different random seeds*

However, **Figure 6** shows that differently initialized Gibbs sampler moves around different local areas of the latent posterior. Therefore, the Gibbs sampler does not converge in this model. Instead of exploring the entire posterior, the Gibbs sampler is stuck in a local optimum.

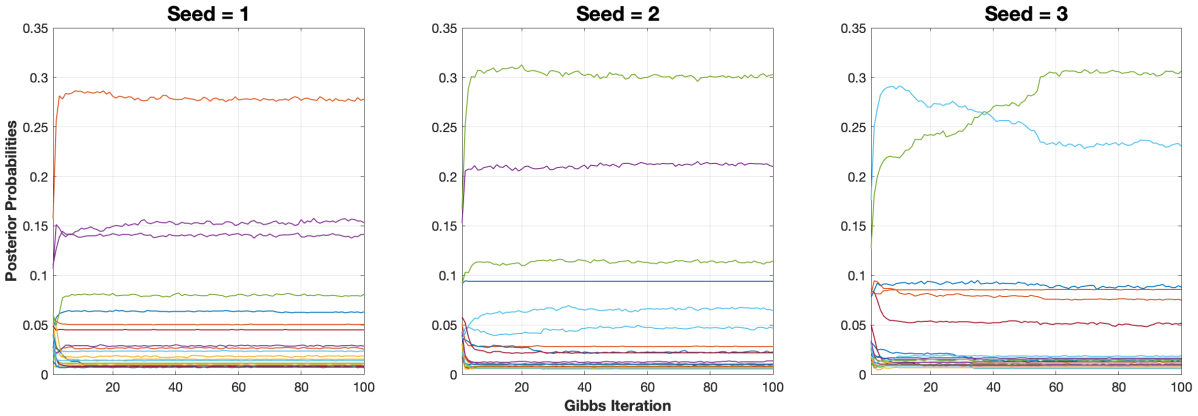

Figure 7 *Evolution of the mixing proportions obtained using different random seeds (100 iterations)*

Figure 7, plotted for 100 Gibbs iterations, shows that it is highly difficult for the Gibbs sampler to escape the local optimum. The failure in convergence is mainly because of the extremely high dimensionality of the posterior distribution. However, the Gibbs sampler stuck in a local optimum is still good at explaining the given test data (see **Table 2**).

Table 2 *Perplexity of the test data B, evaluated from different models*

Model	***ML Estimate***	***Bayesian Inference***	***Collapsed Gibbs Sampler***				
			seed = 1	***seed = 2***	***seed = 3***	***seed = 4***	***seed = 5***
Perplexity	∞	2697	2093	2151	2126	2115	2100

There are 20^{2000} different ways of assigning the topics in data A. Hence, it is computationally impossible to find the best configuration. An alternative is to find local optima, which are set of 'reasonable' configurations.

Table 2 also shows that differently initialized Gibbs samplers show similar perplexity, despite having different configurations. This suggests that all these configurations are consistent with the data.

5. Question E

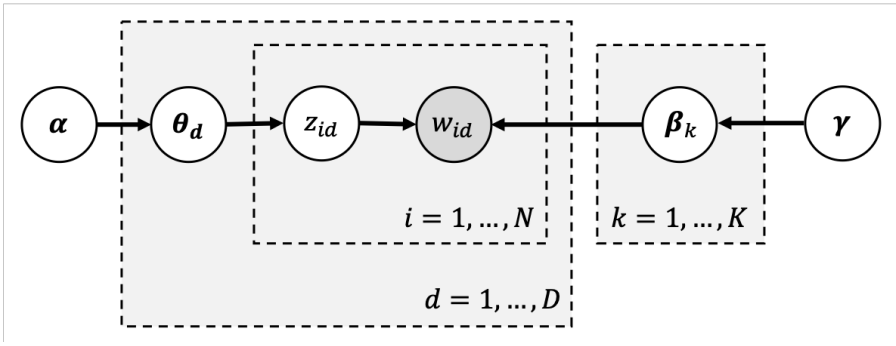

Figure 8 *Graphical representation of the LDA model*

The LDA model allows the words in the same document to be drawn from different topics. The predictive topic distribution of a single word can be expressed simply with the word counts.

$$p(z_{nd} = k|\{z_{-nd}\},\{w\},\gamma,\alpha) \propto \frac{\alpha + c^{k}_{-nd}}{\sum_{j=1}^{K}(\alpha + c^{j}_{-nd})} \times \frac{\gamma + \tilde{c}^{k}_{-nd}}{\sum_{m=1}^{M}(\gamma + \tilde{c}^{k}_{-m})} \tag{13}$$

Command 4 *Calculating the topic posterior after each Gibbs iteration*

```
theta(:,iter)= (skd(:,2)+alpha) / (sum(skd(:,2)+alpha));
```

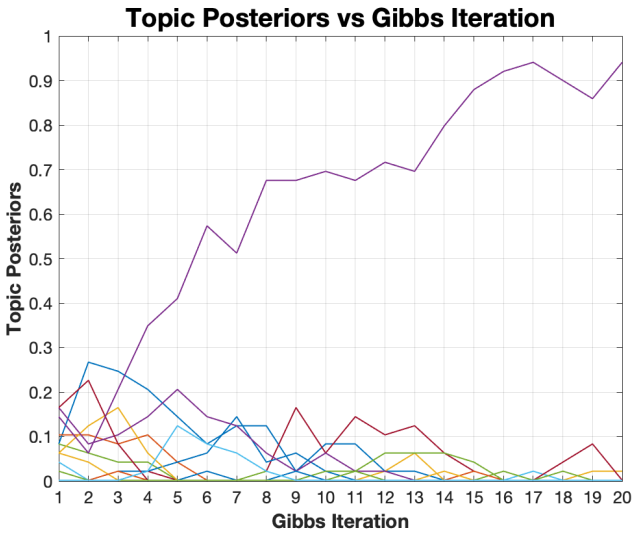

Figure 9 *Topic posteriors as a function of Gibbs iteration*

Figure 9 shows that the topic posterior is dominated by a single topic. This can be the result of the rich-get-richer property of the collapsed Gibbs sampler. This also gives an important insight that most of the words can be explained with a single topic. Such topic contains the most common words in the data (See **Table 3** in page 9).

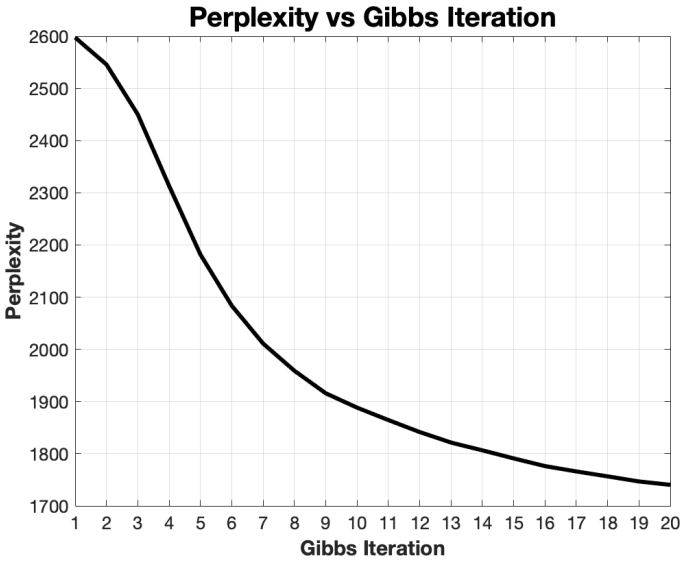

Figure 10 *Perplexity for documents in B as a function of Gibbs iteration*

Figure 10 shows that the perplexity decreases smoothly, despite the irregular fluctuation of the topic posterior. The resulting perplexity is 1772, much lower than the mixture-of-multinomials model. The perplexity would continue to decrease for larger iteration, but it is computationally cost-efficient to stop at 20 iterations.

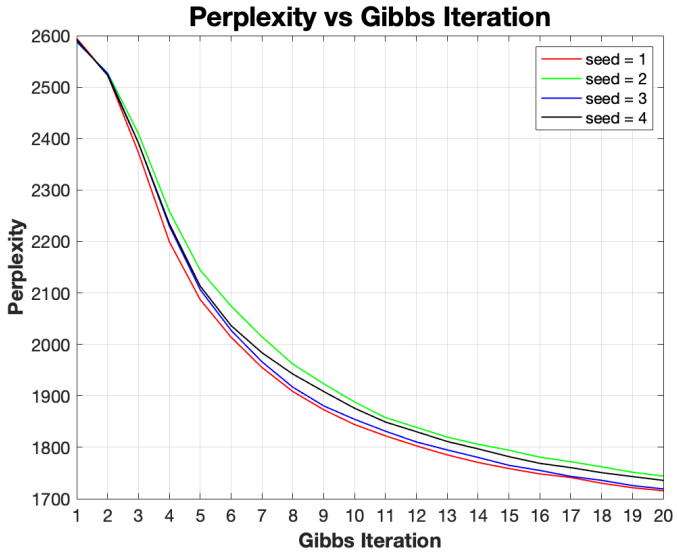

Figure 11 *Perplexity as a function of Gibbs iteration, obtained for different random seeds*

Similar to the mixture-of-multinomials model, differently initialized models show similar performances. Again, this suggests that there are multiple 'reasonable' configurations.

Command 6 *Calculating word entropy after each Gibbs iteration*

```
beta(:,o) = (swk(:,o) + gamma) / (sum(swk(:,o) + gamma)); % repeat K times
entropy = entropy + (-beta(j,i)*log(beta(j,i))); % repeat W times for each K
```

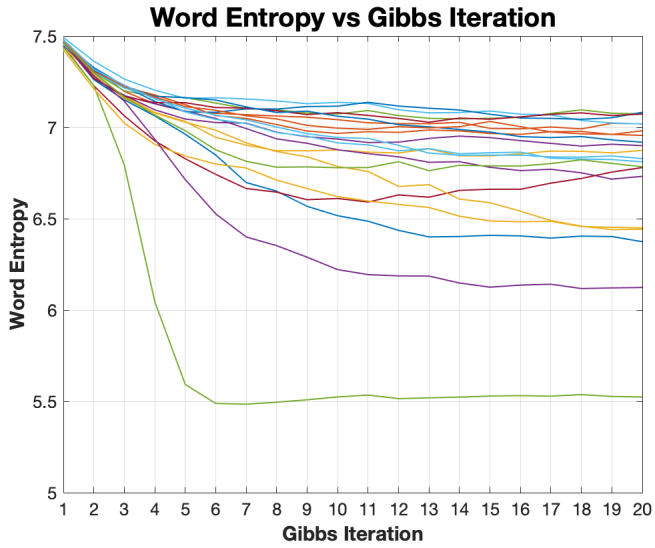

Figure 12 *Word entropy as a function of Gibbs iteration*

Entropy can be interpreted as the measure of uncertainty associated with each observation. Hence, zero entropy is achieved when all the probability mass is concentrated in one word. On the contrary, the entropy is maximized if the distribution is uniform. The entropy associated with $\boldsymbol{\beta}_k$ can be defined as following:

$$\text{Entropy}_k = -\sum_{m=1}^{M} (\boldsymbol{\beta}_k)_m \log_e(\boldsymbol{\beta}_k)_m \tag{14}$$

The unit of entropy is 'nats' since natural logarithm is used. When converted to 'bits' ($1\ bit = \log_e 2\ nats$), the entropy can be interpreted as the average number of bits required for the variable-length word encoding. Hence, small entropy is preferable.

In the beginning, the entropies are very high across all topics. Then, the entropies decrease as $\boldsymbol{\beta}_k$ become specialized. After around 10 iterations, the entropies stop decreasing and start fluctuating. This is because the Gibbs sampler has reached a local optimum.

The topic with the lowest word entropy is Topic 12 in this case. **Table 3** shows that this topic concentrates the probability heavily on one word. Such specialization explains the low entropy.

Table 3 *Top 10 words for topic 12 (lowest word entropy) and topic 18 (highest posterior)*

Topic 12 (Lowest Word Entropy)		***Topic 18 (Highest Posterior)***	
Words	***Probability***	***Word***	***Probability***
november	0.0639	kerry	0.0576
poll	0.0137	dean	0.0404
house	0.0131	poll	0.0244
account	0.0126	edwards	0.0237
governor	0.0125	primary	0.0207
electoral	0.0124	clark	0.0197
republicans	0.0123	democratic	0.0173
senate	0.0118	gephardt	0.0145
polls	0.0117	iowa	0.0136
vote	0.0099	polls	0.0127