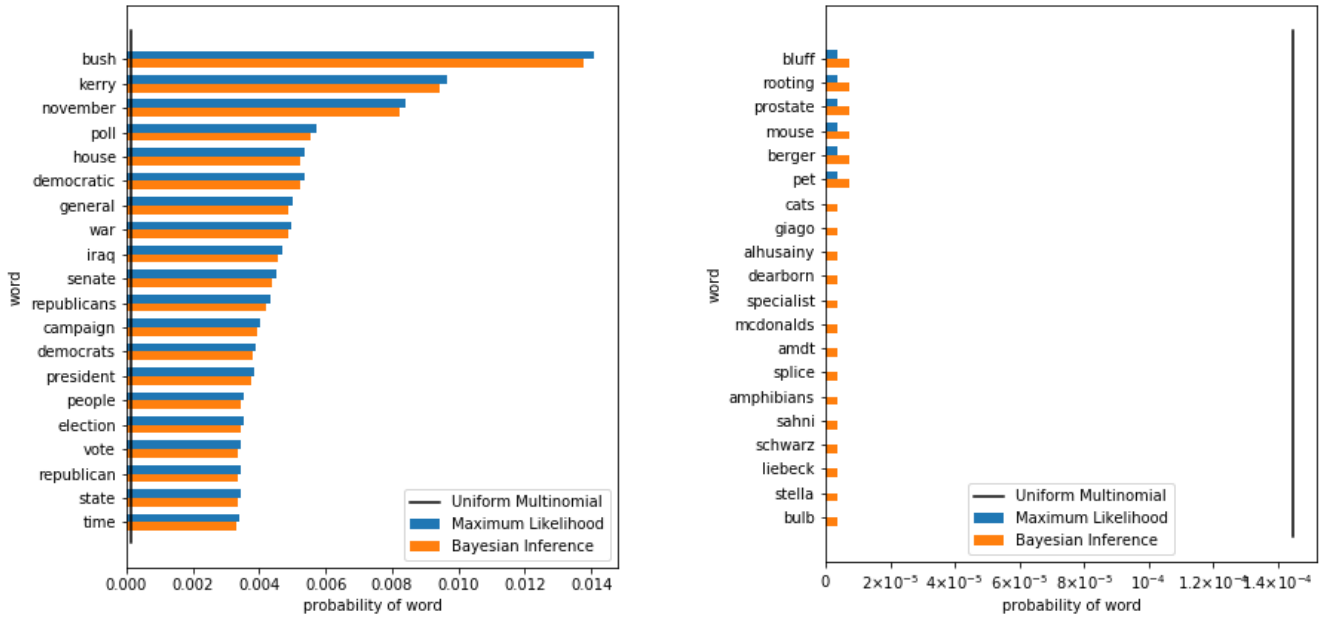
4F13 Coursework #3: Latent Dirichlet Allocation

CCN: **5636B** 5th Dec 2019 word count: 997



(a) maximum β_m's

(b) minimum β_m's

Figure 1: Comparisons between the **global** vocab distributions $\underline{\beta}$ derived from three different Multinomial models - Uniform, Maximum Likelihood (ML) and Bayesian Inference (BI, $\gamma = 1$). (a) and (b) show the 20 vocabs of **highest and lowest probabilities** respectively (in both ML and BI).

Preface

Section a, b & c model the global vocab distribution using a single multinomial, i.e. every word w_{nd} from every document d is drawn from $\underline{\beta} = [\beta_1, ..., \beta_m, ..., \beta_M]^T$, where M is the total number of vocabs.

Then Bayesian Mixture model (BMM) in Section d attempts categorising documents into different topics, by **assigning a single topic** $k \in \{1, ..., K\}$ **to** (the latent variable z_d of) **each document**. Each topic k now has its own topic-specific vocab distribution $\underline{\beta}_k$, from which every word in topic-k documents is drawn.

Finally, Latent Dirichlet Allocation (LDA) model in Section e **allows a document to belong to multiple topics** k, **characterised by its topic distribution** $\underline{\theta}_d = [\theta_{1d}, ..., \theta_{kd}, ..., \theta_{Kd}]^T$. Each word in the document will be assigned a topic drawn from $\underline{\theta}_d$, then the actual value of the word will be drawn from the topic-specific vocab distribution $\underline{\beta}_k$.

a Maximum Likelihood Multinomial model

The contents in Figure 1 span over Section a, b & c. Here only the β_m's from Maximum Likelihood (blue bars) are concerned:

$$\text{ML } \beta_m = \frac{\text{total count of vocab } m \text{ in train set}}{\text{total no of words in train set}} = \frac{c_{m,train}}{N_{train}} \text{ or } \frac{c_{m,train}}{\sum_{m'=1}^{M} c_{m',train}}$$

When attempting to generate all test documents from the ML $\underline{\beta}$, the test set log probability is given by:

$$ln[P(\underline{w}_{test}|\underline{\beta})] = \sum_{m=1}^{M} c_{m,test} ln[P(w_{nd} = m|\underline{\beta})] = \sum_{m=1}^{M} c_{m,test} ln(\beta_m)$$

where $\underline{w}_{test}$ are all the words in test set,and $c_{m,test}$ is the corresponding total count of vocab m.

Hence, for any possible test set, the highest log probability is when all the words are **vocab** m **with the largest** β_m, i.e. "bush" with $\beta_m = 0.0141$ in Figure 1a:

$$\text{largest } ln[P(\underline{w}_{test}|\underline{\beta})] = N_{test} ln(0.0141)$$

In contrast, the lowest log probability is when one or more words from test set did not appear in the train set, since the probability of the unseen word β_m will be zero and so will that of the entire test set.

$$\text{smallest } ln[P(\underline{w}_{test}|\underline{\beta})] = ln(0) = -\infty$$

This implies that ML $\underline{\beta}$ is not capable to deal with any unseen vocabs. This problem can be addressed by Bayesian Inference.

```
vocab_counts_IDorder = np.zeros(V.shape[0])
for vocab_m in train_vocab_IDs: # Start from 1 to 6906, with missing vocabs.
    vocab_m_indices = np.where(A[:,1] == vocab_m)
    vocab_counts_IDorder[vocab_m - 1] = np.sum(A[vocab_m_indices, 2].T) # c_m
beta_IDorder = vocab_counts_IDorder/np.sum(vocab_counts_IDorder)
```

Listing 1: Code for calculating ML $\underline{\beta}$.

b Bayesian Inferred Multinomial model

This second model derives $\underline{\beta} = [\beta_1, ..., \beta_M]^T$ from the predictive mean of posterior $P(\underline{\beta}|D, \gamma)$, where γ is the "pseudo vocab count" of the Dirichlet prior $\underline{\beta} \sim Dir(\gamma)$. Since the posterior is $\underline{\beta}|D, \gamma \sim Dir(\gamma + c_{1,train}, ..., \gamma + c_{M,train})$, the predictive mean is given by:

$$\bar{\beta}_m = \mathbb{E}_{P(\underline{\beta}|D,\gamma)}[\beta_m] = \frac{\gamma + c_{m,train}}{\sum_{m'=1}^{M}(\gamma + c_{m',train})}$$

As observed in Figure 1 (orange bars), the introduction of pseudo counts γ leads to slight decreases in β_m of the highest probabilities (Figure 1a) when compared against ML β_m. These removed probability masses are re-distributed in the β_m of the lowest probabilities which previously were zeros (Figure 1b). Hence the model can now assign non-zero probability to a test vocab unseen in train set.

Different values of pseudo count result in different vocab distributions. For small $\gamma = 0.1$, the actual observed counts are dominant over pseudo counts, hence Bayesian $\underline{\beta}$ resembles ML $\underline{\beta}$ (Figure 2). For large $\gamma = 100$, pseudo counts dominate, which enforces $\underline{\beta}$ to be more uniform across all vocabs (Figure 3).

```
# Use `vocab_counts_IDorder` from Section (a).
counts_IDorder = vocab_counts_IDorder + gamma
beta_post_mean = counts_IDorder / np.sum(counts_IDorder)
```

Listing 2: Code for computing predictive mean $\underline{\beta}$

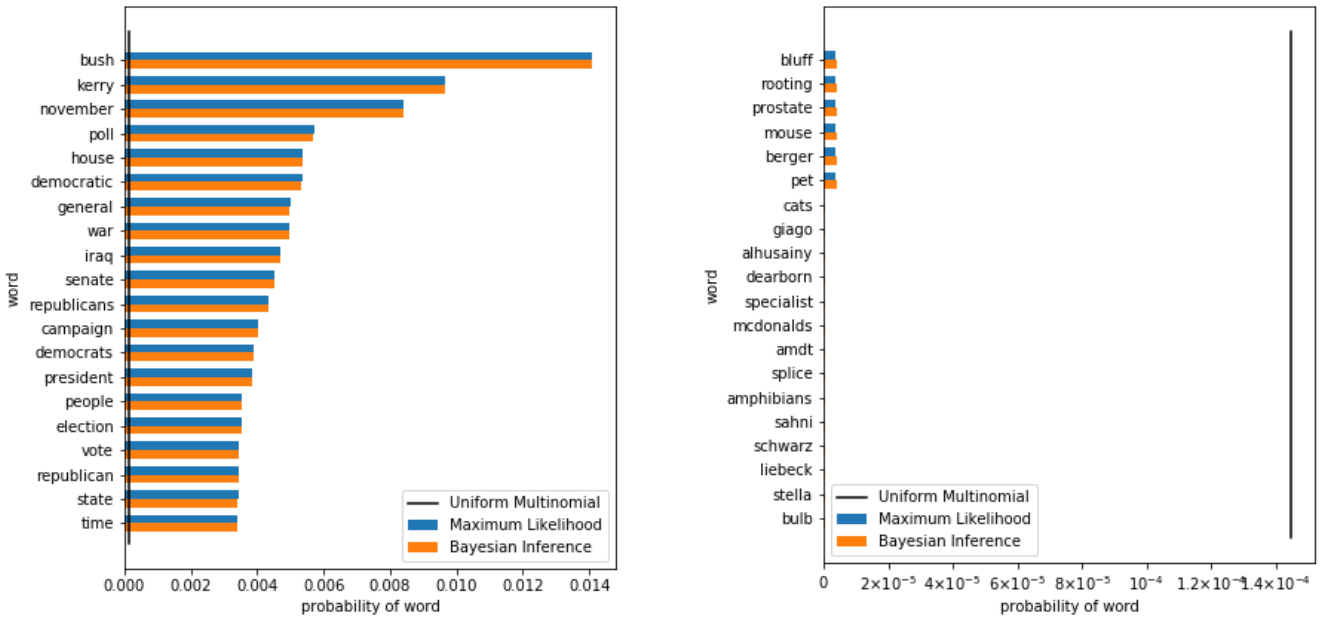
c Perplexities in different Multinomial models

This section focuses on the generative performance of a model on the test set, measured by its per-word perplexity (PWPP) of the entire test set. Consider PWPP of a test document:

$$\text{PWPP of test doc 2001} = exp(-\frac{1}{N_{d=2001}} ln[P(\underline{w}_{d=2001}|\underline{\beta})]) = 4373.1$$

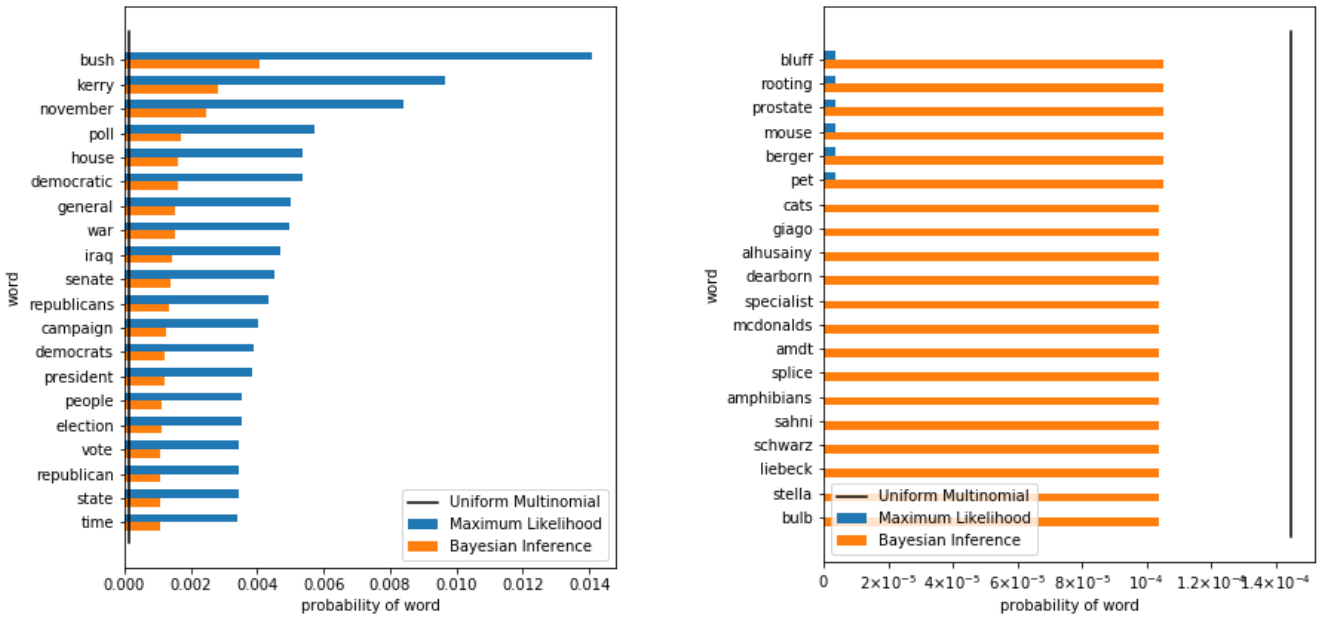
where $ln[P(\underline{w}_{d=2001}|\underline{\beta})]$ is the log probability doc 2001:

$$ln[P(\underline{w}_{d=2001}|\underline{\beta})] = \sum_{m=1}^{M} c_{m,d=2001} ln(\beta_m) = -3688.6$$



(a) maximum β_m's (b) minimum β_m's

Figure 2: Variant of Figure 1, with $\gamma = 0.1$



(a) maximum β_m's (b) minimum β_m's

Figure 3: Variant of Figure 1, with $\gamma = 100$

PWPP of a document basically means the number of "available choices" for choosing a test word from all M vocabs. "Available choices" refers to the average number of choices **weighted by the vocab probabilities** $\underline{\beta}$. Lower PWPP represents lower uncertainty for choosing the test word, hence better generative performance of the model.

To consolidate this idea, first consider $\underline{\beta}$ being uniform multinomial, i.e. $\beta_m = \frac{1}{M}$. To generate a test word from it, There are $M = 6906$ "available choices". The PPWP is M, as shown in the line in Figure 4.

On the other hand, when $\underline{\beta}$ is trained by Bayesian Inference, some vocabs will have higher probabilities than others. This extra information lowers the uncertainty for choosing a test word, hence the number of "available choices" (PWPP) decreases. This is why most datapoints in Figure 4 lie below the uniform

multinomial line. Different documents have different PWPPs because their different sets of words $\underline{w}_d$ yield different joint probabilities $p(\underline{w}_d)$ from $\underline{\beta}$.

Finally, an explanation for **why some test documents have PWPP greater than** $M = 6906$ is given here. Although **non-uniform** vocab multinomial distribution generally lowers PWPP, this is only true when most words in the particular test document are also frequently observed in the training documents. Otherwise the joint probabilities of words in the test document will be smaller than $(\frac{1}{M})^{N_d}$, contributed by the words with tiny β_m in Figure 1b, 2b or 3b. As a result, PWPPs for those documents are greater than M.

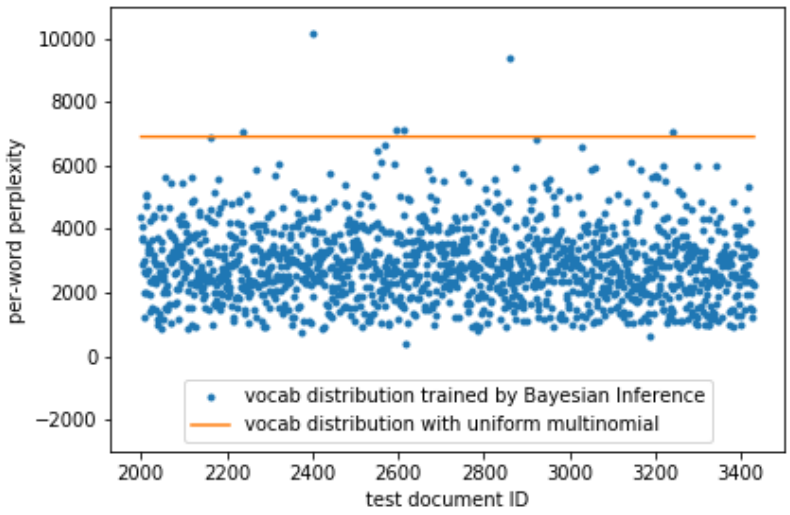


Figure 4: Per-word Perplexity for each document in the test set **B**, using Bayesian Inference Multinomial model and being compared with Uniform Multinomial model.

Averaging over all documents gives the PWPP for the entire test set:

$$\text{PWPP of test set} = exp(-\frac{1}{N_{test}}\sum_d ln[P(\underline{w}_d|\underline{\beta})]) = 2684.0$$

where N_{test} is the total word count in test set.

```
log_prob_doc_d = 0
for i, m in enumerate(w):  # array 'w' contains vocab IDs in doc d
    log_prob_doc_d += c[i] * np.log(BI_beta[m-1])
```

Listing 3: Code for computing the log probability of test document d.

d Bayesian Mixture model (BMM)

In BMM, Gibbs sampling was used to update the topic assignment $z_d \in \{1, ..., K\}$ to each document d. At the end of each iteration, the new set of $\{z_d\}_{d=1:2000}$ can be used to compute the mixing proportions $\underline{\theta} = [\theta_1, ..., \theta_K]^T$:

$$\theta_k = \frac{\alpha + c_k}{\sum_{k'=1}^{K}(\alpha + c'_k)}$$

Each θ_k represents the probability of a test document being assigned to topic k.

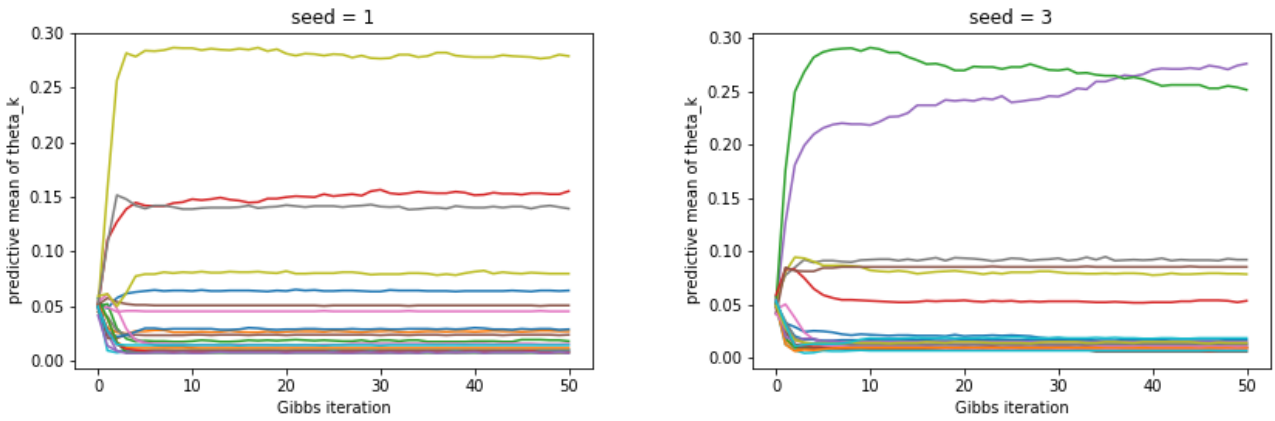
Figure 5 shows their evolution as Gibbs is run. Two different sets of initial topic assignments $\{z_d\}_{d=1:2000}$ were used, and it is very apparent from the colour code of the graphs that each θ_k (with same colour on both graphs) converges to very different probabilities.

The underlying reason behind the very different final $\{\theta_k\}_{k=1:20}$ is that the initial $\{z_d\}_{d=1:2000}$ determines where in the manifold Gibbs sampling begins. This greatly affects which local optimal the Gibbs sampler converges to. As observed in Figure 5, in each case the Gibbs sampler is stuck at a local maximum, not able to converge to the global maximum.

Note that although Gibbs fails to explore the entire stationary distribution of this model (under practical number of iteration), this model gives a significantly lower PWPP of test set than before:

$$\text{PWPP for seed } 1 = 2092.3$$

$$\text{PWPP for seed } 3 = 2123.6$$



(a) (b)

Figure 5: Evolution of mixing proportions in BMM as a function of Gibbs iterations. (a) and (b) are evolution for different initial topic assignments $\{z_d\}_{d=1:2000}$.

```
arr_ck_plus_alpha = arr_of_rows_sk_docs + alpha
arr_theta_k = arr_ck_plus_alpha / sum(arr_ck_plus_alpha[0])
```

Listing 4: Code for computing a 20×51 matrix of $\{\theta_k\}_{k=1:20}$ evolution where each row corresponds to each curve in Figure 5a or 5b.

e Latent Dirichlet Allocation (LDA) model

As mentioned in **Preface**, each document in LDA model has its topic proportions $\underline{\theta}_d = [\theta_{1d}, ..., \theta_{kd}, ..., \theta_{Kd}]^T$. Each θ_{kd} are derived from the count of words c_{kd} (in doc d) assigned to topic k:

$$\theta_{kd} = \frac{\alpha + c_{kd}}{\sum_{k'=1}^{K}(c_{k'd})} = \frac{\alpha + c_{kd}}{K \times \alpha + N_d} \quad \text{, where } N_d \text{ is total no of words in document } d$$

Figure 6 plots the evolution of $\underline{\theta}_{d=1000}$ during Gibbs sampling. The observation that one particular topic has a much larger final proportion than the others can be explained by the "rich-get-richer" property - a "rich" topic with large word count is more likely to be assigned to the word being resampled, hence getting even larger word count ("richer"). On a different note, 50 iterations may not be enough since some θ_{kd} (such as the blue curve) still have an increasing trend.

The PPWP of test set for this model is 1642.9, much lower than any previous PPWPs. Hence this model performs best in generating the test set.

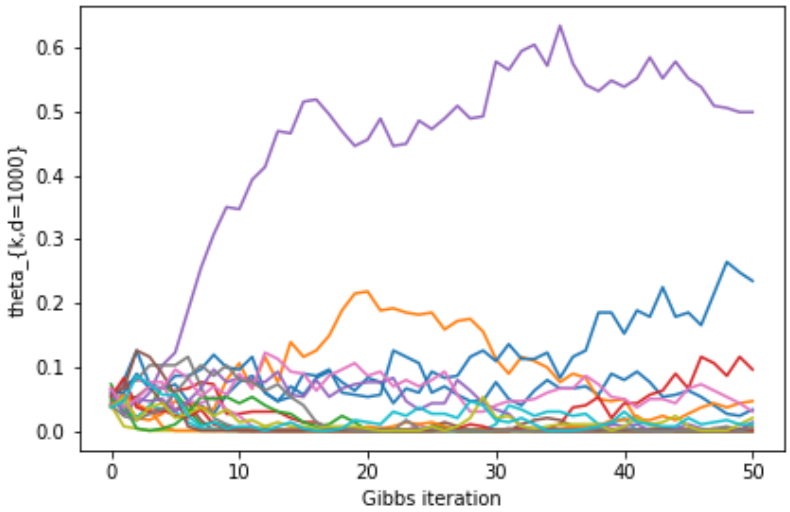


Figure 6: Evolution of topic proportions $\underline{\theta}_d$ for document 1000 ($K = 20$)

Finally, Figure 7 shows that the vocab entropy for each topic k:

$$H_k(vocab) \text{ distributed by } \underline{\beta}_k = -\sum_{m=1}^{M} \beta_{mk} log_2 \frac{1}{\beta_{mk}}$$

decreases as sampling proceeds, indicating each $\underline{\beta}_k$ becomes more non-uniform. Each topic is specialising into certain subset of vocabs.

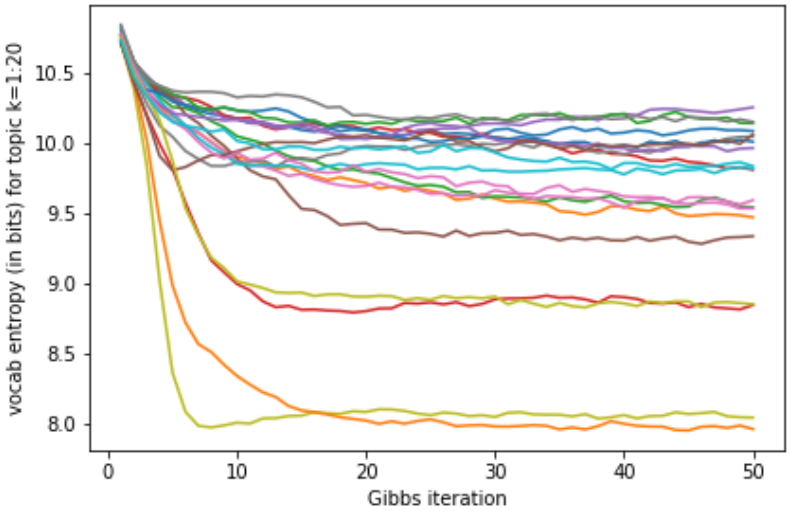


Figure 7: Evolution of vocab entropy for each topic k. Note that calculation used log in base 2, so the units of entropy is bits.

```
for k = 1:K
  entropy = 0;
  for m = 1:W
      entropy = entropy + (-beta(m,k)*log2(beta(m,k)));
  end
  word_entropy(k,iter) = entropy;
end
```

Listing 5: Code for computing the vocab entropy for each topic. (MATLAB was used in this last section.)