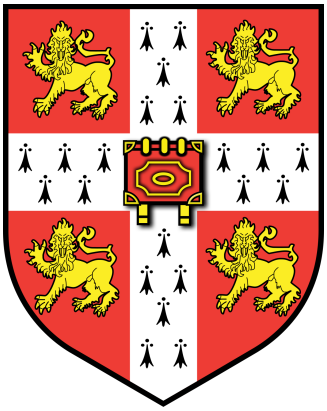

University of Cambridge

MPhil MLMI

4F13 Assignment 3

Candidate Number:
J902G

Date:
December 2, 2021

Word Count: 991

Part A

```
for i in range(min(A[:,1]), max(A[:,1])+1):
    occurences_i = np.where(A[:,1] == i) # return the indices of A where the word ID is i
    count_i = sum(A[occurences_i][:,2])
    betas[i-1] = count_i
betas /= sum(betas)
```

Listing 1: Code to calculate the maximum likelihood estimates for the β values, computed using Equation (2)

For the most simple multinomial model, the entire training-set is treated as a single document, and the likelihood for the parameters β is given by:

$$P(\vec{w}^{(\text{train})}|\vec{\beta}) = \prod_{m=1}^{M} \beta_m^{C_m^{(\text{train})}} \tag{1}$$

for $\tilde{\beta} = (\beta_{\mathbf{1}}, ..., \beta_{\mathbf{M}})^{\mathbf{T}}$. The maximum likelihood estimation for the β values having observed the words in the training documents is then simply given by the frequency of occurrence of each word:

$$\hat{\beta}_{\mathbf{m}} = \frac{\mathbf{C_m^{(train)}}}{\sum_{\mathbf{n}} \mathbf{C_n^{(train)}}} \tag{2}$$

for $C_m^{(train)}$ the count of word m in the training data. The highest log-probability for the test-set is then when every word is the most probable from the training-set, i.e. every word is *'Bush'*, and is given by:

$$\frac{\ln P(\vec{w}^{(\text{test})}|\hat{\beta})}{N_{\text{test}}} = -4.26 \tag{3}$$

for $\vec{w}^{(\text{test})}$) the words in the test-set, containing N_{test} words. Conversely, The lowest probability is if one of the words with probability 0 is chosen (see Figure 1). Then $P(\vec{w}^{(\text{test})}) = 0$ and $\ln P(\vec{w}^{(\text{test})}) = -\infty$.

Figure 1 shows the relative probabilities of top and bottom 20 words in vocabulary, found via maximum likelihood estimation of β values having observed the training-set. It demonstrates a shortcoming of MLE, that if at test-time a word has not been seen in training, the word probability becomes 0, and therefore so does the probability of the entire test-document too.

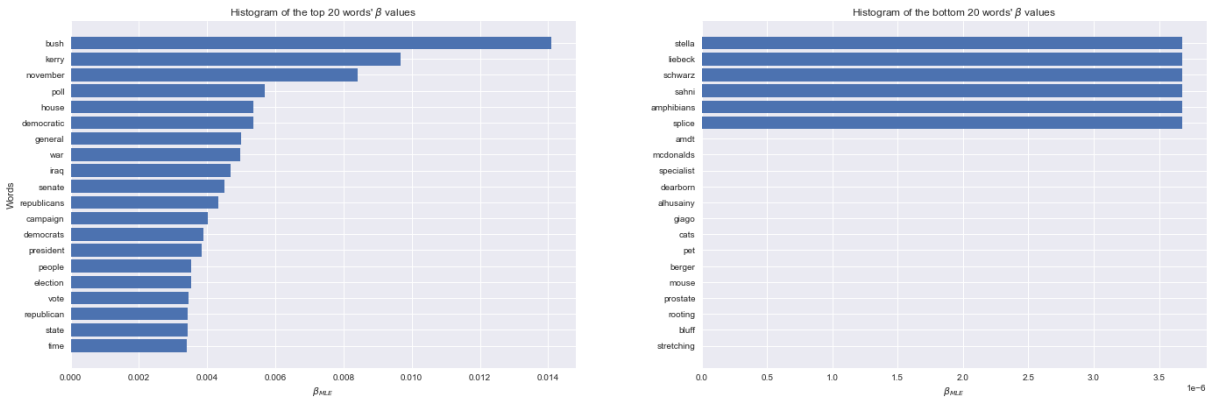

Figure 1: Plot of the probabilities of the top 20 and bottom 20 words in the training data, ordered in descending order of β values.

Part B

```
alpha = 0.1
for i in range(len(betas)):
    betas[i] += alpha
betas = np.divide(betas, sum(betas))
```

Listing 2: Code snippet for altering the values for β to give the optimal values from Bayesian inference.

```
id_sorted = np.argsort(betas)
betas_sorted = betas[id_sorted]

num_show = 20
plot_words = []
for i in np.arange(num_show, 0, step=-1):
    word = V[int(id_sorted[-i])] # words in ascending order of probability
    plot_words.append(word[0][0])
```

Listing 3: Code snippet for ordering the words according to their probability. This shows the top 20, but can be easily adjusted for the bottom 20.

To fix the issue of 0 probability words, Bayesian inference can be used. Now a (symmetric) Dirichlet prior is defined over the values of β, which gives a posterior of:

$$P(\vec{\beta}|\vec{w}^{(\text{train})};\alpha) \propto P(\vec{\beta}|\alpha)P(\vec{w}^{(\text{train})}|\vec{\beta}) \propto \text{Dir}(\vec{C}+\alpha) \tag{4}$$

where C_m is the number of times word m appears in the training-set, and α is the symmetric Dirichlet parameter representing the pseudo-count of each word m prior to observations. The predictive distribution can then be calculated for new words:

$$\begin{aligned} P(w^* = m|w^{(\text{train})}) &= \int P(w^* = m|\vec{\beta})P(\vec{\beta}|\vec{w}^{(\text{train})};\alpha)d\vec{\beta} \\ &= \frac{\alpha + C_m^{(train)}}{\sum_n \left(\alpha + C_n^{(train)}\right)} \end{aligned} \tag{5}$$

In effect, this is adding a smoothing term to the MLE result. Figure 2 shows the effect of this - including a pseudo-count for *all* words (even if unseen in training) means that there are no words with 0 probability, instead these words are assigned a small constant probability.

The extent of this smoothing is governed by the magnitude of α. The probabilities for two α's are plotted in Figure 3. As α increases, rare words are assigned higher probability and common words lower probability. Indeed, in the limit that $\alpha >> \max_m C_m^{(\text{train})}$, the predictive distribution tends to a uniform distribution, as seen from Equation (5).

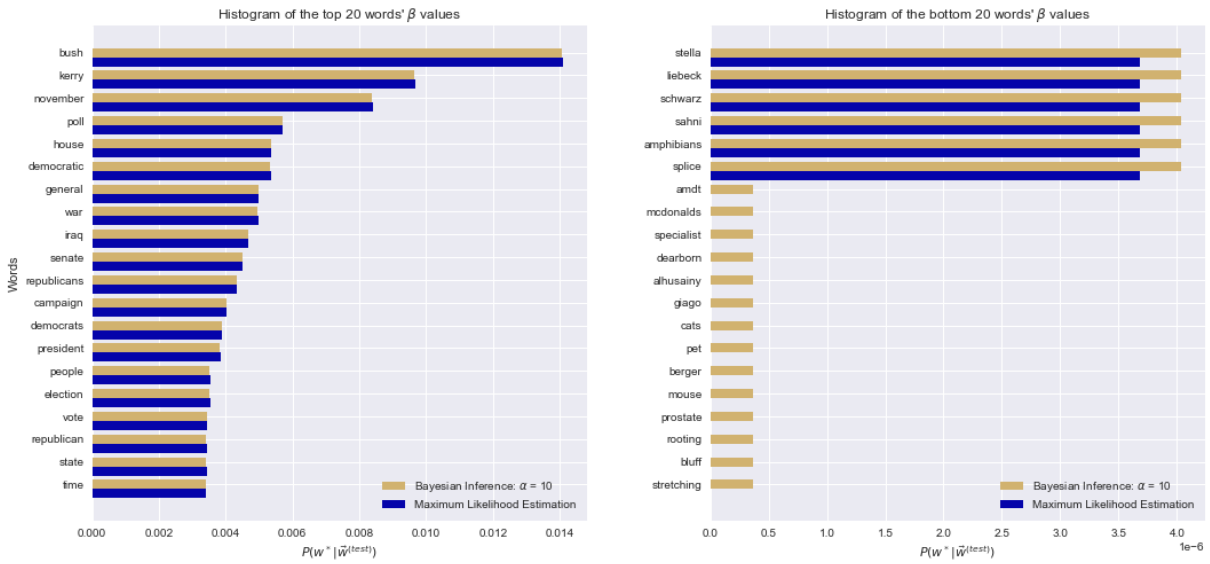

Figure 2: Comparison of using maximum likelihood and Bayesian inference for obtaining the values for β (i.e. using Equation (2) vs Equation (5)). The words are again ordered in descending order of β values (using the Bayesian inference β values here for sorting, but the sorting is the same for both so this is not important).

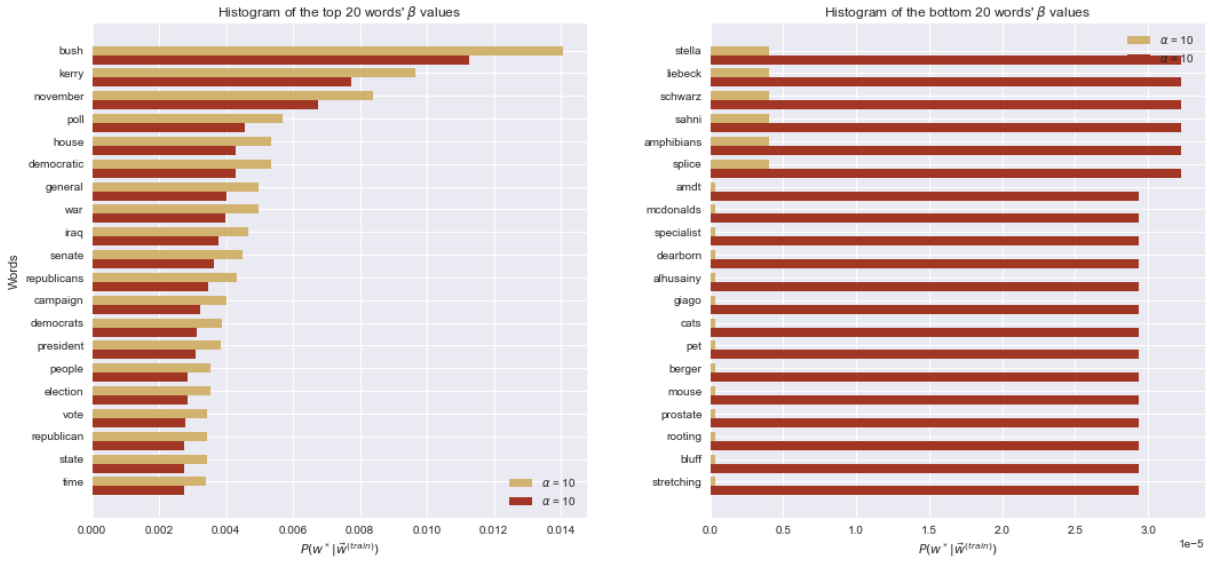

Figure 3: Comparison of β values using high and low α values in the symmetric Dirichlet prior.

Part C

```
doc_id = 2001
data = np.array(list(filter(lambda x: x[0] == doc_id, B)))
betas_for_test_doc = list(itemgetter(*data[:,1]-1)(betas)) # -1 so that 0-start index of betas
    matches with 1-start index of data[:,1]
count_multiplied = list(map(lambda x,y: x*np.log(y), data[:,2], betas_for_test_doc))
cat_log_probability = sum(count_multiplied)
```

Listing 4: Code snippet for calculating the categorical log-probability of the test-document with ID 2001, using the β values found via training.

```
doc_id = 2001
data = np.array(list(filter(lambda x: x[0] == doc_id, B)))
counts = data[:,2]
probs = list(itemgetter(*data[:,1]-1)(betas))
M = sum(counts)
result = log_factorial(M) - sum(log_factorial(counts)) + sum(counts * log(probs))
```

Listing 5: Code snippet for calculating the multinomial log-probability of the test-document with ID 2001, using the β values found via training, calculated using Equation (7). 'log-factorial' here uses scipy's $\ln(x!)$ approximation.

There are two distributions to consider for multiple category observations, the categorical distribution which assumes set word order, and the multinomial distribution which considers only counts without order. They thus differ by a combinatoric factor and are given by:

$$P^{(cat)}(\{W_{d,m}\}_{m=1}^{M}|\vec{\beta}) = \prod_{m=1}^{M} \beta_m^{C_m} \tag{6}$$

$$P^{(multi)}(\{C_m\}_{m=1}^{M}|\vec{\beta}) = \frac{M!}{(C_1!C_2!...C_M!)} \prod_{m=1}^{M} \beta_m^{C_m} \tag{7}$$

For the test-documents, we are only given word counts and not word sequences, and therefore must use the multinomial distribution to calculate document probabilities. However, as we know that each test-document is only a single combination of the word counts (and does not include all word combinations), we then recover the categorical distribution by diving through by the number of combinations. Therefore, from here-on, categorical distributions are used, e.g. for calculating the perplexity, which is calculated as:

$$\text{PWPP}(\vec{w}_d^{(\text{test})}) = \exp\left[-\frac{1}{N_\text{d}} \ln P(\vec{w}_d^{(\text{test})}|\vec{\beta})\right] \tag{8}$$

Log-probability (categorical)	Log-probability (multinomial)	Per-word-perplexity
-3691	-1691	4399

Table 1: The results for the document with ID 2001, using the Bayesian model from part B with $\alpha = 0.1$.

For N_d words in test-document d, the per-word-perplexity across all D test-documents is given by:

$$\text{PWPP}(\{\vec{w}_d^{(\text{test})}\}_{d=1}^{D}) = \exp\left[-\frac{1}{N_{\text{test}}}\sum_{d=1}^{D}\ln P(\vec{w}_d^{(\text{test})}|\vec{\beta})\right] \tag{9}$$

The Bayesian model from part B with $\alpha = 0.1$ gave a test-set perplexity of 2697. Figure 6 shows (categorical) log-probabilities and per-word-perplexities for all test-set documents. The perplexity differs between documents because it depends on the documents' word contents, and therefore changes as the words in a document change. For the case of a uniform multinomial, every word has equal probability: $1/M$. Thus the perplexity is simply equal to the number of words in the vocabulary (6906).

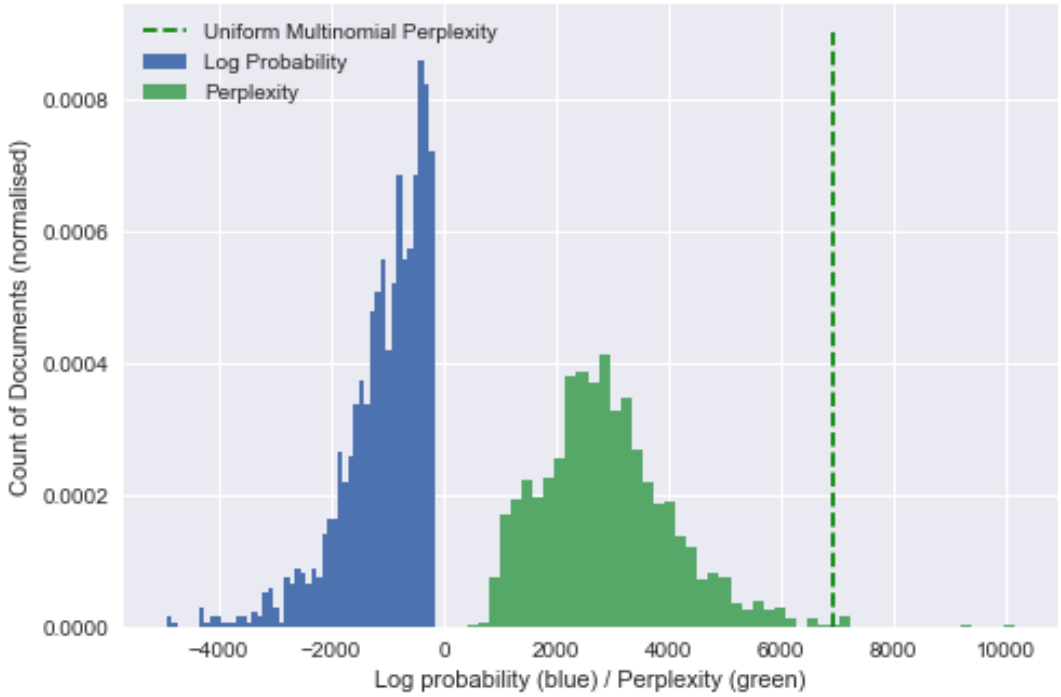

Figure 4: A plot of two histograms: the left (blue) representing the distribution of log-probabilities over all of the test-documents. The right shows the distribution of perplexities for the same test-documents (with the dashed line being the perplexity of a uniform multinomial). Both were calculated using the categorical probability density function (Equation (6)). Notice how there are a few documents with perplexity higher than the perplexity of the uniform multinomial. This occurs if a test-document contains many words that were very rare or did not occur in training, which results in the log-probability of that document being very negative, and thus the PWPP very positive from Equation (8).

Part D

```
theta[iter_] = (sk_docs.T[0] + alpha) / sum(sk_docs.T[0] + alpha)
posterior_alpha_params = sk_docs.T[0] + alpha
```

Listing 6: Code snippet for calculating the mixing proportions, θ, for each Gibbs iteration. The posterior Dirichlet parameter (the observed counts plus the pseudo-counts) can also be calculated for each iteration in order to sample from these posterior Dirichlet distributions.

$$\begin{aligned} \vec{\theta} &\sim \text{Dir}(\alpha) \\ z_d|\vec{\theta} &\sim \text{Cat}(\vec{\theta}) \end{aligned} \tag{10}$$

Using Bayesian inference, the posterior distribution over mixing proportions is given by:

$$\begin{aligned} P(\theta_k|\{z_d\}_{d=1}^D, \alpha) &\propto P(\theta_k|\alpha)P(\{z_d\}|\theta_k) \propto \text{Dir}(C'_k + \alpha) \\ \langle\theta_k\rangle_{P(\theta_k|\{z_d\}_{d=1}^D,\alpha)} &= \frac{C'_k + \alpha}{\sum_i (C'_i + \alpha)} \end{aligned} \tag{11}$$

where C'_k is now the count of *documents* in topic k. Figure 5 shows the mean of the posterior Dirichlet distribution above, plotted as a function of Gibbs iteration number for 3 random seeds. The plot shows that the random initialisation of the collapsed Gibbs sampler still has a noticeable effect on the mixing proportions, even after 50 iterations. Convergence can only be reached if Gibbs sampling explores the entire posterior distribution and finds the global optimum, in which case the converged means would be the same for all initialisations. This is not the case here, as for each seed, the sampler is getting stuck in an often different local optimum and not exploring the whole posterior distribution. Hence whilst it appears that the means converge within each initialisation, this convergence is not global. In order to explore the entire distribution we would therefore need to take many random initialisations using many different seeds and compare their mixing proportions.

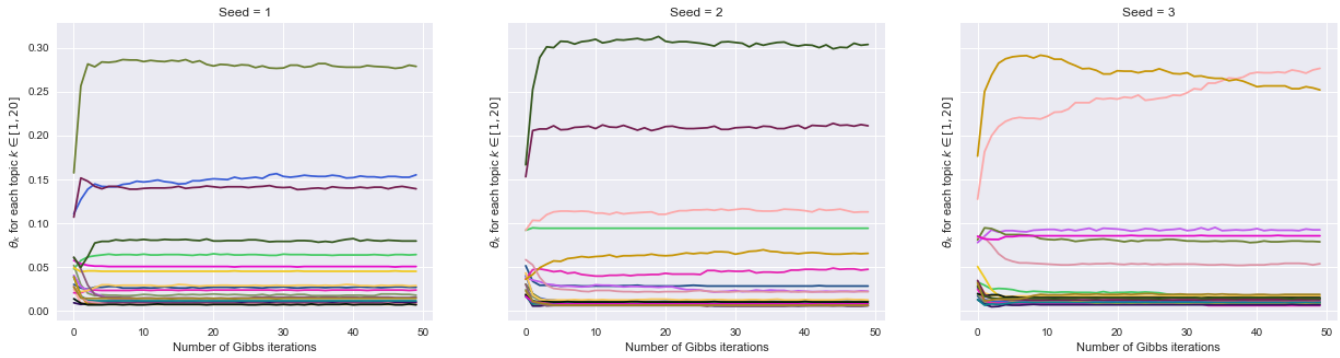

Figure 5: Means of the Dirichlet posteriors (i.e. the mixing proportions) as a function of Gibbs iteration number for three different seeds. The fact that several topics have mixing proportions decreasing towards 0 for each seed suggests that it could be possible to model the test set with fewer than 20 topic categories without greatly affecting the perplexities.

Figure 6 also shows how mixing proportions can be sampled from the posterior for each iteration, by using the fact that the posterior is a $\text{Dir}(\alpha + \vec{C}')$ distribution.

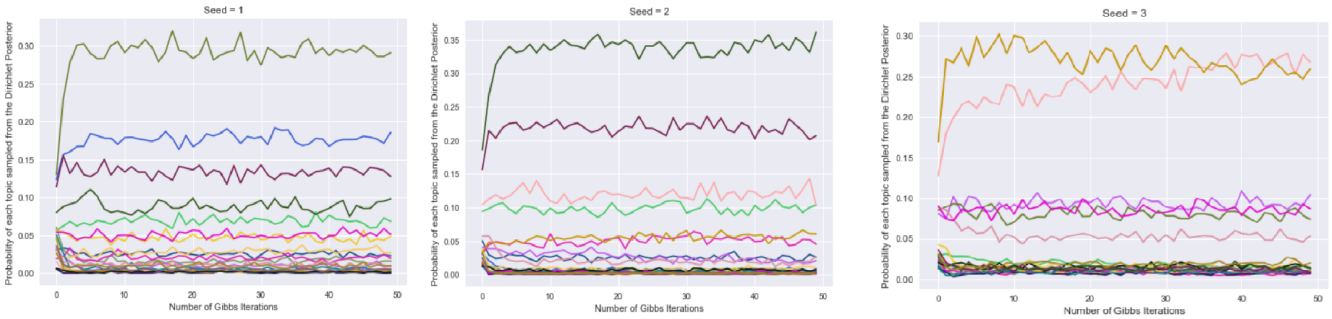

Figure 6: Samples drawn from the posterior Dirichlet distributions. The samples oscillate around the means plotted in Figure 5, and seem to have a variance of oscillations that increases with mixing proportion size.

Despite the fact that Gibbs sampling is very unlikely to converge to a global optimum, the test-set perplexities of the BMM model are still a lot lower for the 3 seeds than for the previous, simpler, models. They are shown in Table 2, and by being lower demonstrate that this model is better able to predict the test-set than the previous methods. The equation for the per-word-perplexities of a single document is shown in Equation (8), and for a set of documents in Equation (9).

	Uniform Multinomial	MLE	Bayesian Inference	Bayesian Mixture Model (seeds 1,2,3)		
PWPPs	6906	∞	2697	2092	2152	2124

Table 2: Per-Word-Perplexities (averaged over all test-set documents) for the three seeds using a Bayesian Mixture Model compared to the simple categorical models. The perplexities are calculated over the entire test-set via Equation (9).

Part E

```
theta[d, iter_] = (skd[:, d] + alpha) / sum((skd[:, d] + alpha))
beta[:,k] = (swk[:, k] + gamma) / sum(swk[:, k] + gamma)
entropy = np.sum(-beta*np.ma.log(beta), axis=0) # entropy per each k
store_entropy_per_iter[iter_] = entropy
```

Listing 7: Code for calculating the entropy of each topic per Gibbs iteration, where the entropy is calculated over all training documents according to Equation (14). Entropy here is in units of *nats*, as a natural logarithm is used.

The LDA model now increases the granularity by defining a distinct topic distribution for each document, $\vec{\theta}_d$, so that the word distributions are able to change within a document. The model can now be described by:

$$\begin{aligned} \vec{\theta}_d &\sim \text{Dir}(\alpha) \\ z_{m,d} &\sim \text{Cat}(\vec{\theta}_d) \\ w_{m,d}|z_{m,d} &\sim \text{Dir}(\vec{\beta}_k) \end{aligned} \tag{12}$$

The means of the Dirichlet topic posteriors for a single document are then calculated for each document as:

$$\langle \theta_{k,d} \rangle = \frac{C_{k,d} + \alpha}{\sum_i (C_{i,d} + \alpha)} \tag{13}$$

where $C_{k,d}$ is the count of words in document d assigned to topic k. Figure 7 shows this mean for two documents (ID 2 and 996). Note the variation of mean values is a lot greater than for in Figure 6 as the plots are specific for documents and not averaged over all documents. These plots show the *'rich-get-richer'* property inherent to Gibbs sampling from Dirichlet distributions: the posteriors become dominated by a single topic as iteration number increases.

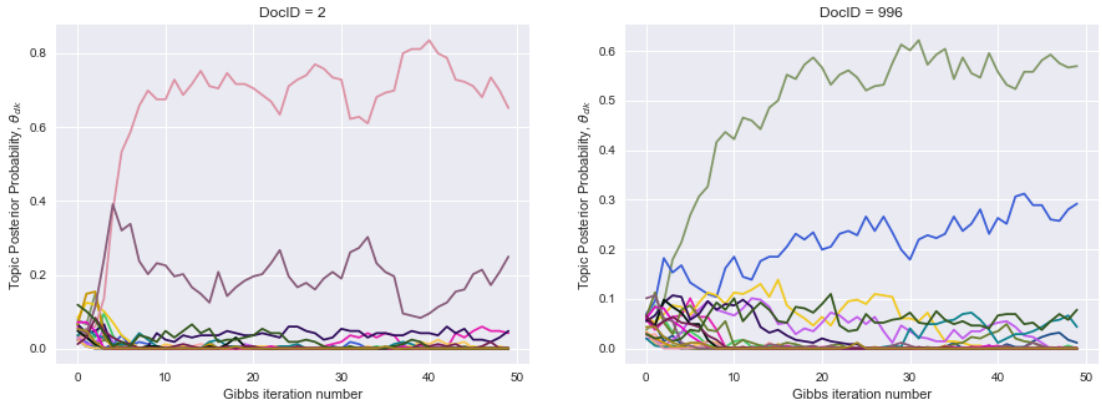

Figure 7: Plots of the means of the *document-specific* posterior Dirichlet distributions for each topic, k, for two arbitrary documents with IDs 2 and 996. In both cases, as the number of iterations increases, one topic grows to dominate the distribution of topics, in the *'rich get richer'* fashion inherent to Gibbs sampling.

After 50 iterations, the perplexities for the documents in B were as shown in Table 3 (and plotted in Figure 9 for seed 1). They are significantly lower than for BMM and the simple models, demonstrating that the additional complexity of LDA is better suited for modelling the test-documents.

	Seed 1	Seed 2	Seed 3
PWPPs	1641	1660	1644

Table 3: Per-Word-Perplexities (averaged over all test-set documents) for the three seeds using LDA. The perplexities are lower than for BMM (which were around 2000) and the simple models. Although different seeds only convergence to local optima, there is consistency between the perplexities for each.

The entropy for the topic-specific word-distributions is calculated over all training documents as:

$$H_k = -\sum_{m=1}^{M} \beta_{k,m} \ln \beta_{k,m} \tag{14}$$

This is plotted in Figure 7 (in units of nats) for the 20 topics as a function of Gibbs iteration. A lower entropy for a topic means that its word distribution is more peaked at certain words and further from uniform. The entropy decreases for all topics with Gibbs iteration because the words distribution are initialised uniformly (due to a symmetric Dirichlet prior) to the same values, and uniform distributions have maximal entropy. As the number of iterations increases, the word distributions for each topic become less uniform and hence the entropy decreases.

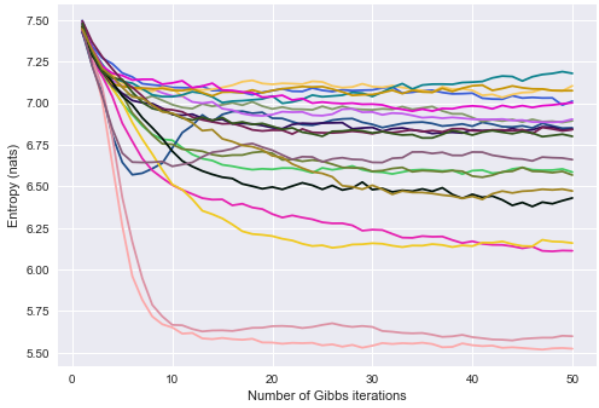

Figure 8: Variation of the entropy of each topic as a function of the Gibbs iteration number. After the 50 iterations it appears that the entropies are converging to local (not global) optima.

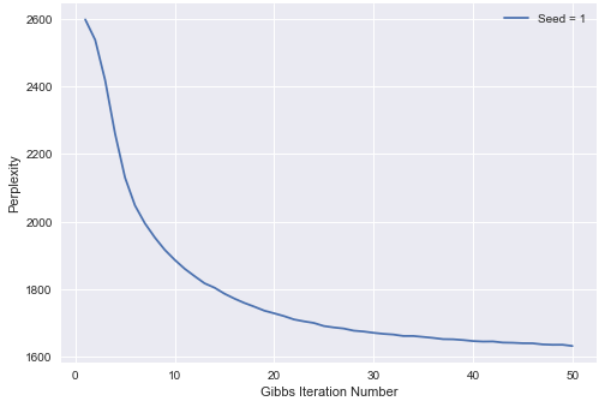

Figure 9: Plot of the decay of perplexity of the entire test-set using LDA with a single seed. After 50 iterations the decay of the perplexity has greatly slowed down, but has not yet converged (even locally), suggesting that more iterations are required even for local convergence.