CUED - Engineering Tripos Part IIB 2021-2022 **Module Coursework**

Module	4F13	Title of report	Coursework 3: Latent Dirichlet Allocation
Date submitted:	03/12/2021	Assessment for this module is ☒ 100% / ☐ 25% coursework of which this assignment forms 33.33 %	

UNDERGRADUATE STUDENTS ONLY		POST GRADUATE STUDENTS ONLY			
Candidate number:	5588C	Name:		College:	

Feedback to the student ☐ **See also comments in the text**		Very good	**Good**	Needs improvmt
CONTENT	**Completeness, quantity of content:** Has the report covered all aspects of the lab? Has the analysis been carried out thoroughly?			
	Correctness, quality of content Is the data correct? Is the analysis of the data correct? Are the conclusions correct?			
	Depth of understanding, quality of discussion Does the report show a good technical understanding? Have all the relevant conclusions been drawn?			
	Comments:			
PRESENTATION	**Attention to detail, typesetting and typographical errors** Is the report free of typographical errors? Are the figures/tables/references presented professionally?			
	Comments:			

Indicative grades are not provided for the FINAL piece of coursework in a module

Assessment (circle one or two grades)		A*	A	B	C	D
Indicative grade guideline		>75%	65-75%	55-65%	40-55%	<40%
Penalty for lateness:		*20% of maximum achievable marks per week or part week that the work is late.*				

Marker: Date:

1 Exercise (a)

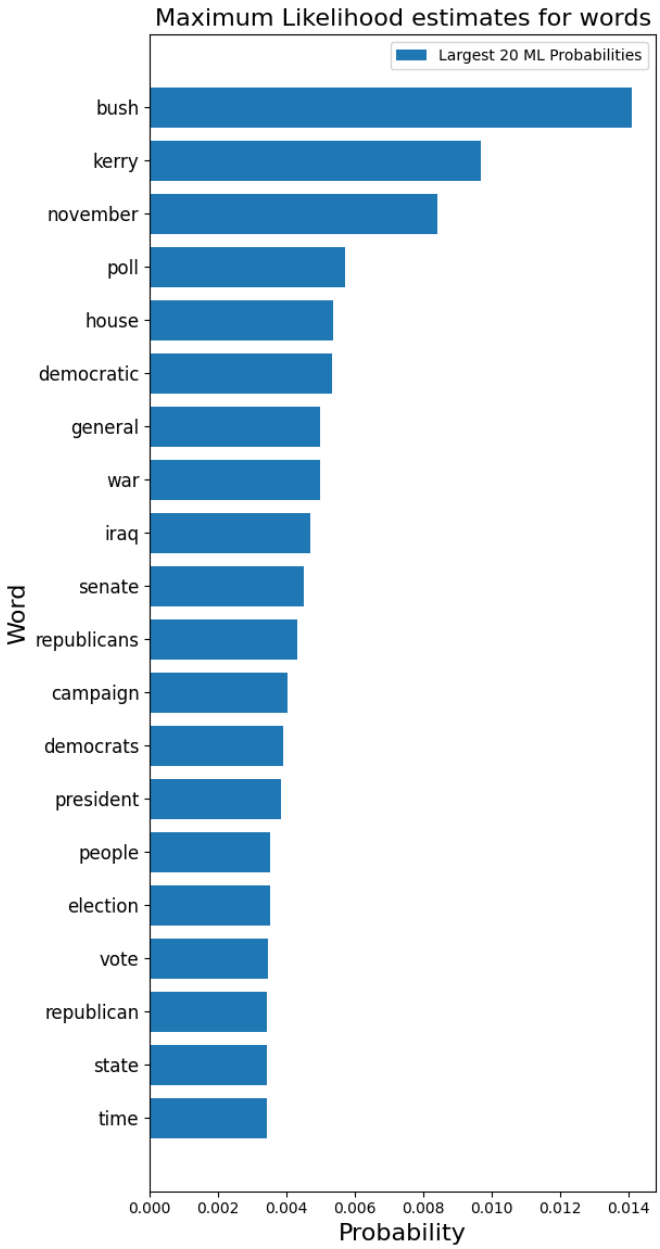

Figure 1: *Sorted Barplot for largest 20 Maximum Likelihood Word Probabilities.*

The Maximum Likelihood estimate for the words in the training set is derived below, where k_s is the number of times word s appears in all of the documents, M is the number of distinct vocabulary words, and $\sum_{i=1}^{M} k_s = N$ is the total number of words in all the documents.

$$\frac{\partial}{\partial \pi_s}\left[\frac{N!}{\prod_{i=1}^{M}(k_i!)}\sum_{j=1}^{M} k_j \ln(\pi_j) + \lambda\left(1 - \sum_{i=1}^{M}\pi_i\right)\right] = 0$$

$$\frac{k_s}{\pi_s} - \lambda = 0$$

$$\pi_s = \frac{k_s}{\lambda}$$

$$\therefore \pi_s^{ML} = \frac{k_s}{N}$$

Given the test set $\mathbf{k}$, it's ML multinomial probability is:

$$p(\mathbf{k}|\boldsymbol{\pi}) = \frac{N!}{\prod_{i=1}^{M}(k_i!)}\prod_{j=1}^{m}\left(\frac{k_j}{N}\right)^{k_j} \tag{1}$$

The lowest test set log probability is $-\infty$, which occurs when a word in the test document does not appear in the training set, and the probability of the test document under the trained model is 0. The highest test set log probability is $\ln(0.014097) \approx -4.262$, corresponding to a test document containing a single count of "bush", since it's the one in the training set with the highest ML probability .

```
pi_ML = [0] * (distinct_num_words)
for word_id in np.unique(A[:, 1]):
    args = np.where(A[:, 1] == word_id)[0]
    pi_ML[word_id - 1] = np.sum(A[args, 2])
pi_ML = np.array(pi_ML)
pi_ML = pi_ML / num_words_in_A
```

Source Code 1: *Code for calculating ML probabilities*

2 Exercise (b)

The predictive probability of word s appearing next in a document under the maximum likelihood method is:

$$p_{ML}(s|\mathbf{k}) = \pi_s^{ML} = \frac{k_s}{\sum_{i=1}^{M} k_i} = \frac{k_s}{N} \tag{2}$$

The Dirichlet distribution is the conjugate prior to the multinomial, hence Bayesian inference yields a Dirichlet posterior when using a Dirichlet prior. The predictive probability for each word s is equal to the expected value of the posterior for that word, where N is the total number of words in the training set:

$$p_B(s|\mathbf{k}) = \frac{\alpha + k_s}{\sum_{i=1}^{M}(\alpha + k_i)} = \frac{\alpha + k_s}{\alpha M + N} \tag{3}$$

The use of the symmetric prior solves the main deficiency of the ML approach. Here, unseen words during training are assigned a non-zero probability, since $\alpha + k_s = \alpha$. The use of the prior, for $\alpha \neq 0$, thus avoids overfitting.

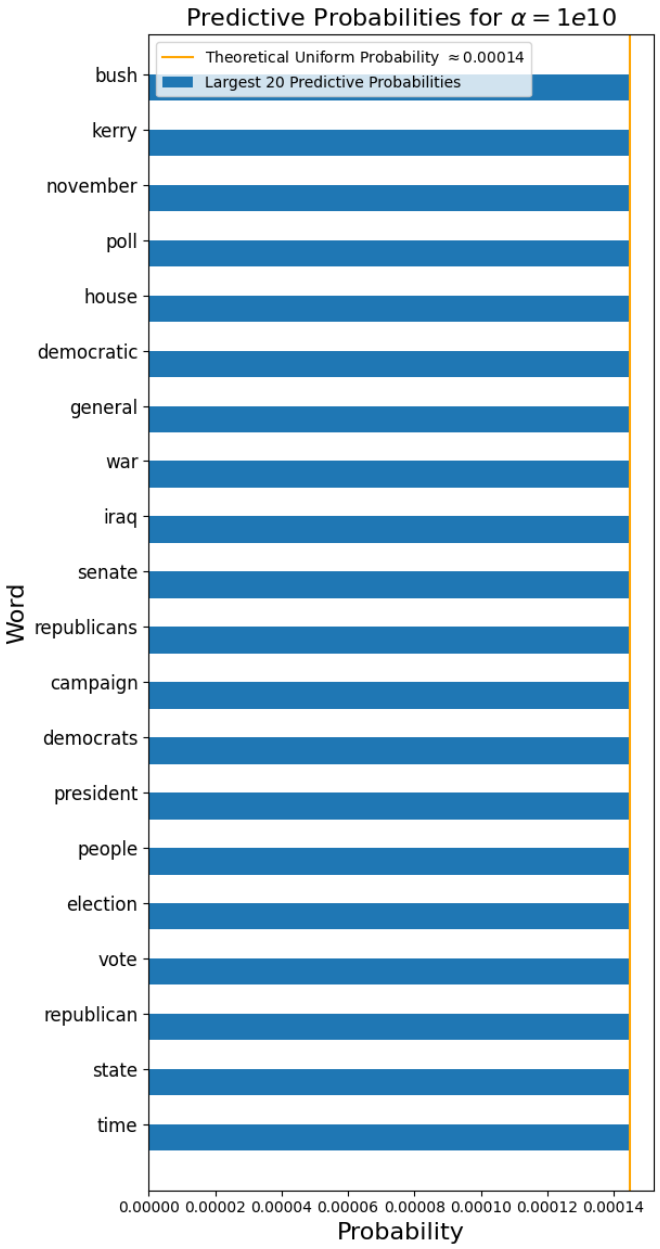

Figure 2: *Sorted Barplot for largest 20 Predictive Probabilities for Large Alpha.*

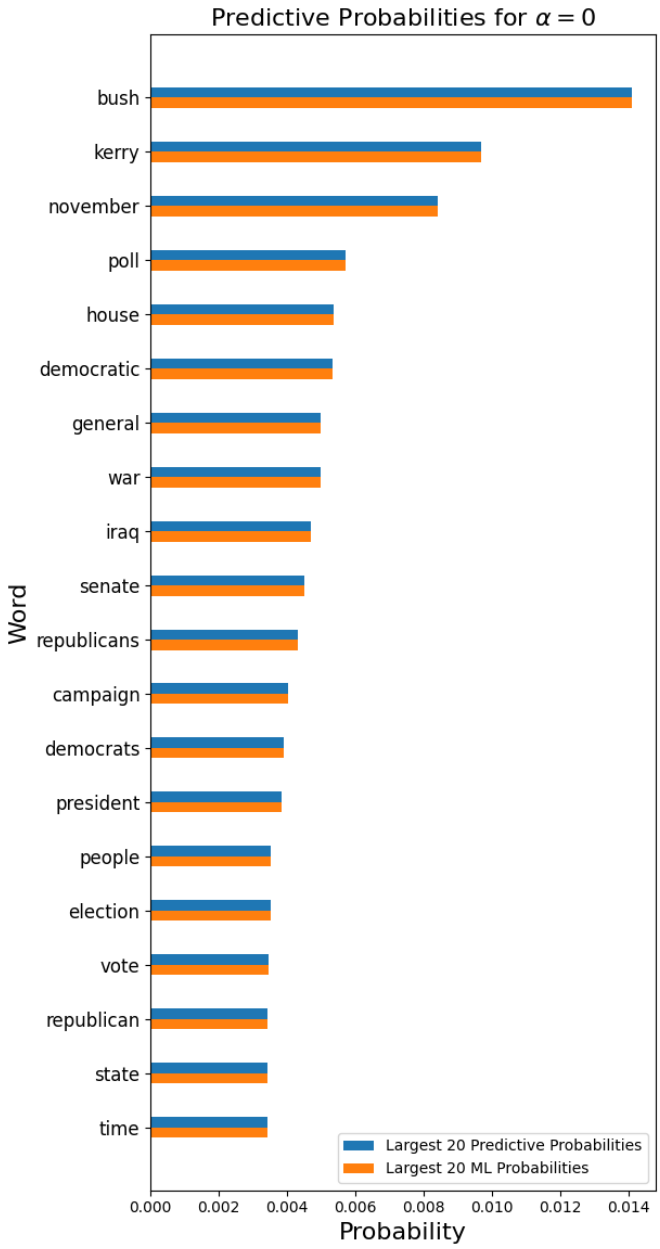

Figure 3: *Sorted Barplot for largest 20 Predictive Probabilities for* $\alpha = 0$*, compared to the ML predictive probabilities.*

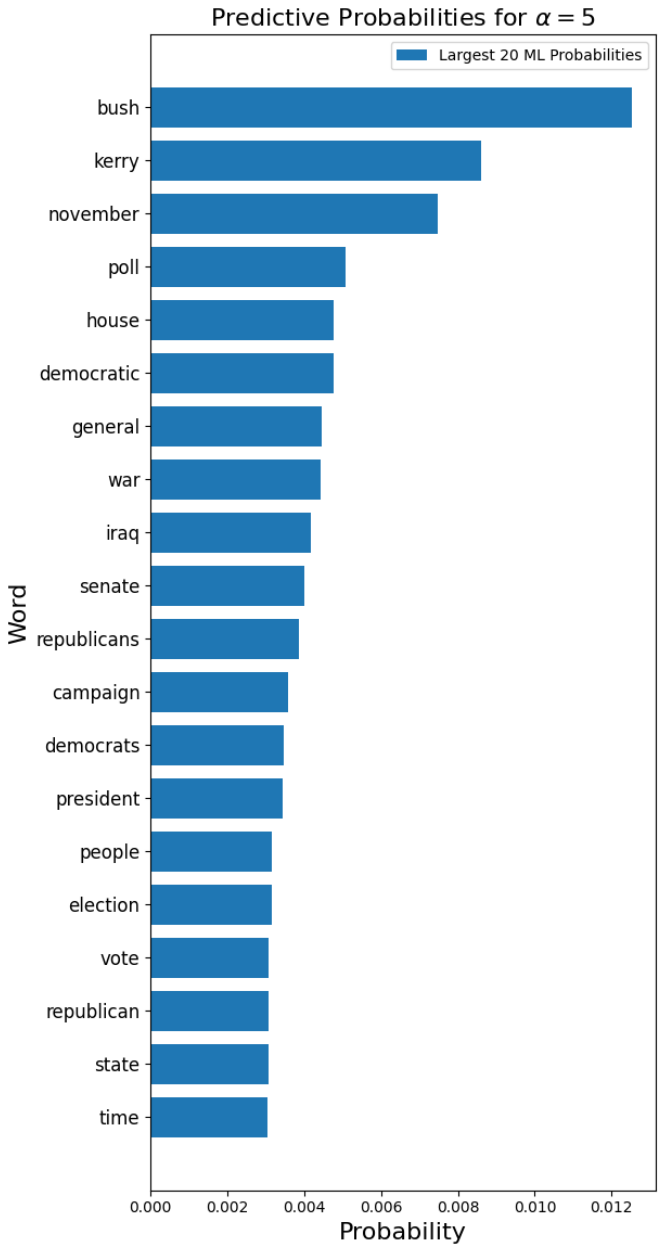

Figure 4: *Sorted Barplot for largest 20 Predictive Probabilities for* $\alpha = 5$.

Clearly, for small α values, we believe the pseudo-counts for each word are a-priori small. This means the posterior is primarily influenced by the likelihood, and, in fact, $\lim_{\alpha \to 0} p_B(s|\mathbf{k}) = \pi_s^{ML}$, meaning we recover the maximum likelihood solution as $\alpha \to 0$, where words have a predictive probabilities proportional to their frequency.

As α increases, the Dirichlet prior is concentrated at the center of the $M-1$ dimensional simplex, representing a strong belief that each of the M prior probabilities take similar values. The posterior concentrates in the same region, and so the influence of the observed data through the likelihood, $p(\mathbf{k}|\boldsymbol{\pi})$, is negligible. In fact, $\lim_{\alpha \to \infty} p_B(s|\mathbf{k}) = \frac{1}{M}$, so each unique word has the same probability of occurring next in a document for

large α.

Figures 2 - 4 show that the relative ordering of words remains unchanged under the Bayesian framework comapred to the ML scenario. To find which words see an increase in predictive probability:

$$\begin{aligned} & p_B(s|\mathbf{k}) > p_{ML}(s|\mathbf{k}) \\ \implies & \frac{\alpha + k_s}{\alpha M + N} > \frac{k_s}{N} \\ & k_s \left(\frac{1}{\alpha M + N} - \frac{1}{N} \right) > -\frac{\alpha}{\alpha M + N} \\ & \therefore k_s < \frac{N}{M} \end{aligned}$$

```
pi = [0] * distinct_num_words
for word_id in unique_A_words:
    args = np.where(A[:, 1] == word_id)[0]
    word_count = np.sum(A[args, 2])
    pi[word_id - 1] = (alpha + word_count) / (alpha * distinct_num_words + num_words_in_A)
for word_id in unique_B_words_not_in_A:
    pi[word_id - 1] = alpha / (distinct_num_words * alpha + num_words_in_A)
```

Source Code 2: *Code for calculating Predictive Probabilities under symmetric Dirichlet Prior*

3 Exercise (c)

A document is a *sequence* of words, so a categorical likelihood is used since the order of words is important. This probability is given by the Equation below, where π_i^{pred} is the probability for word i shown in Equation (3) and N is the number of document words.

$$\begin{aligned} \ln p(\{w_i\}_{i=1}^N | \boldsymbol{\pi}^{pred}) &= \ln \prod_{i=1}^{N} p(w_i|\boldsymbol{\pi}) \\ &= \sum_{i}^{N} \ln \left[p(w_i|\boldsymbol{\pi}) \right] \\ &= \sum_{i=1}^{M} k_i \ln \pi_i^{pred} \end{aligned}$$

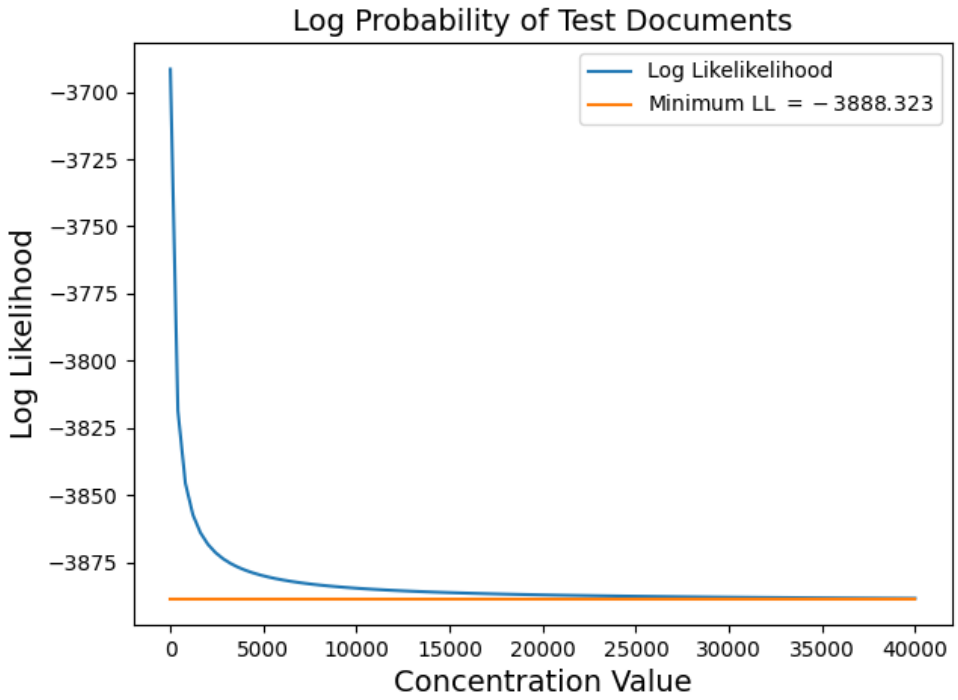

Figure 5: *Log Probability of test document 2001 under symmetric Dirichlet for different concentration values.*

$$\text{perplexity} = \exp\left(-\frac{l}{n}\right) = \exp\left[-\frac{1}{n}\sum_{i=1}^{n}\ln p(w_{i,d}|\boldsymbol{\pi})\right] \tag{4}$$

Equation (4) shows the per-word perplexity for a document d with log-likelihood l, and n words. For a multinomial model with uniform predictive probability π, the log-likelihood is $\ln(\pi^n)$. In the limit as $\alpha \to \infty$, our Bayesian model has a uniform probability of $\frac{1}{6906}$ and the perplexity is equal for all documents. This perplexity, shown below, is the maximum, since it corresponds to a predictive model with equal probability to all words, and hence lowest log-likelihood.

$$\text{perplexity} = \exp\left(-\frac{-n\ln(6906)}{n}\right) = 6906$$

For smaller α, different documents have different perplexities. In general, perplexity will be higher for documents with a high count of low-probability words, and in fact, as $\alpha \to 0$, the perplexity for documents which have words that didn't appear in the training set tends to ∞. This reflects the fact that perplexity measures how well the trained model explains unseen data. As $\alpha \to 0$, the posterior becomes the ML likelihood, and so the model cannot fit documents with unseen words during training.

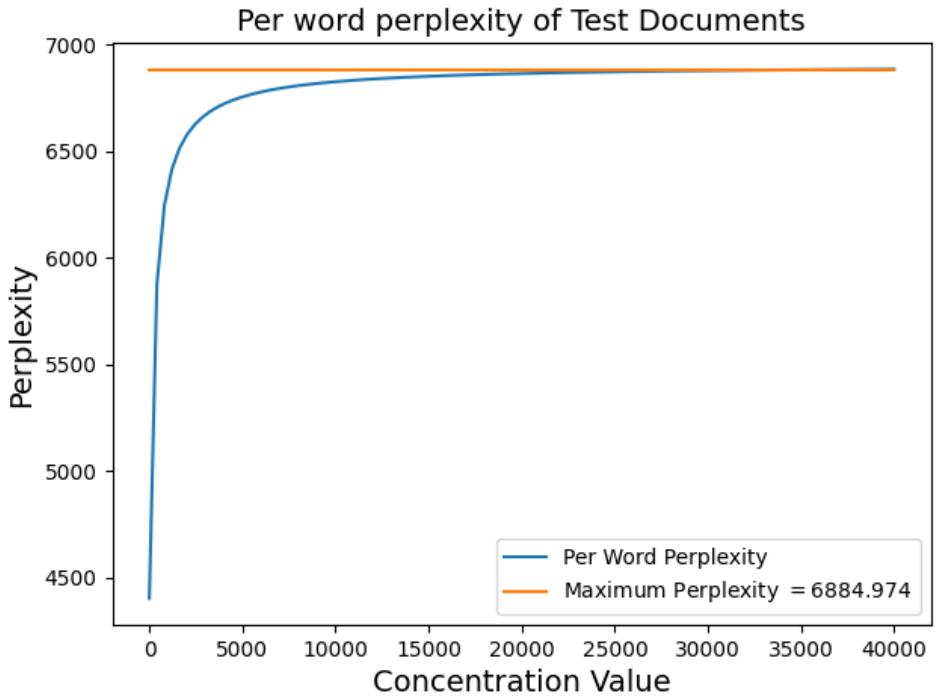

Figure 6: *Per word Perplexity of test document 2001 under symmetric Dirichlet for different concentration values.*

```
doc_2001_predProbs = np.zeros(doc_2001_wordIDs.shape[0])
for i in range(doc_2001_wordIDs.shape[0]):
    arg = np.where(pi[:, 0] == doc_2001_wordIDs[i])[0][0]
    doc_2001_predProbs[i] = pi[arg, 1]
# Log likelihood of document
ll = np.sum(doc_2001_counts.transpose().dot(np.log(doc_2001_predProbs)))
lls.append(ll)
pps.append(np.exp(-ll / num_words_2001))
```

Source Code 3: *Code for calculating log likelihood and perplexity of a document.*

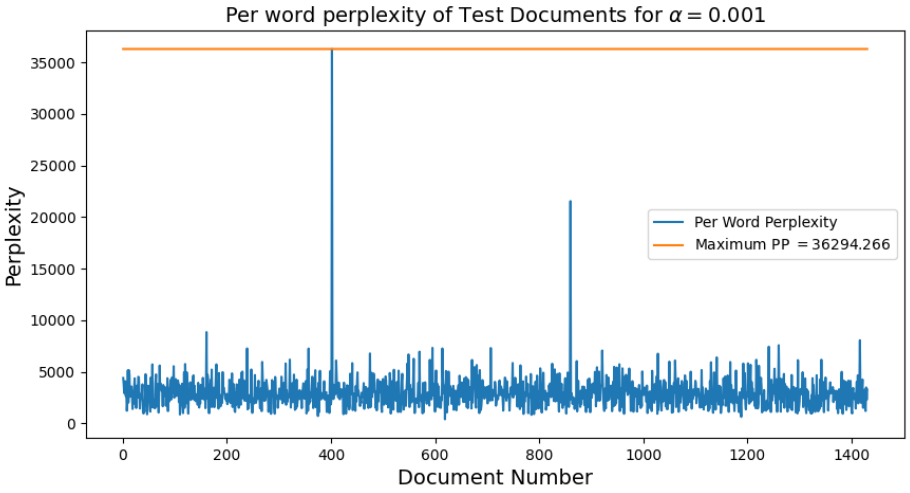

Figure 7: *Per word Perplexity of test documents for small alpha.*

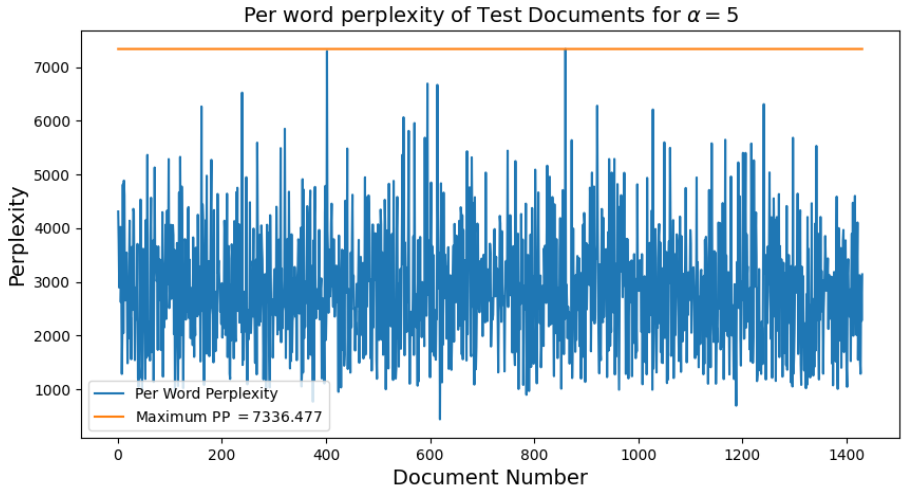

Figure 8: *Per word Perplexity of test documents for* $\alpha = 5$

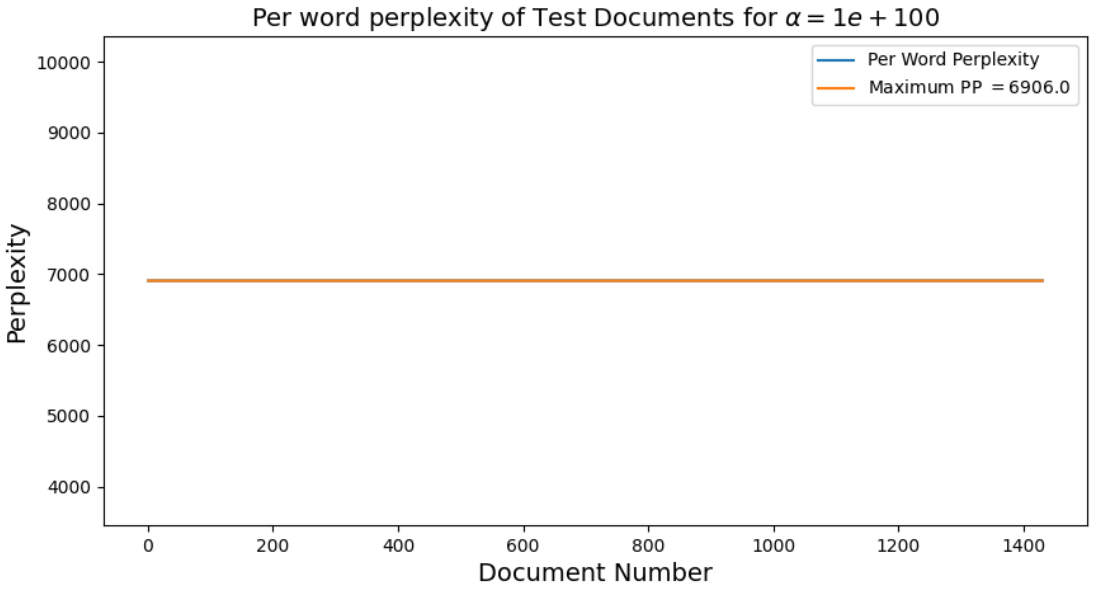

Figure 9: *Per word Perplexity of test documents for large alpha.*

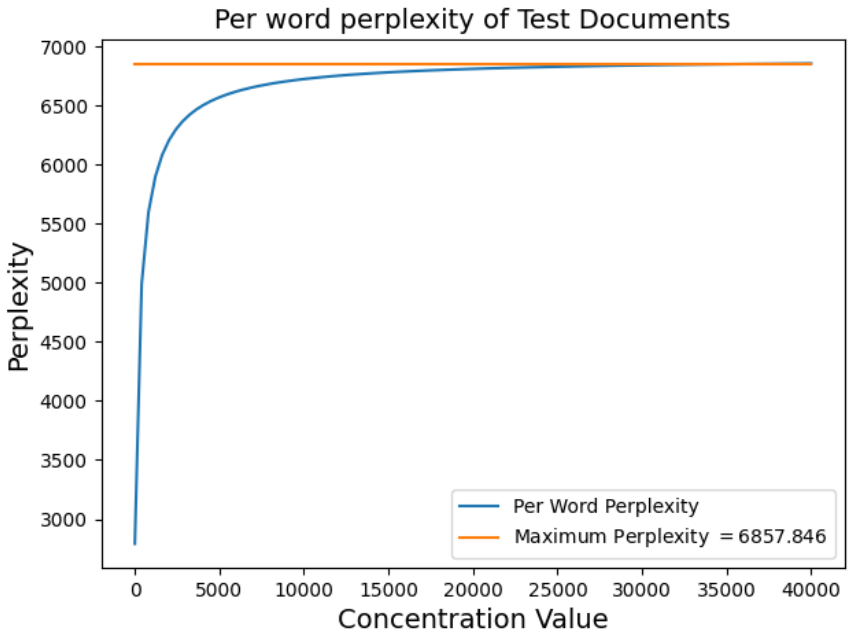

Figure 10: *Per word Perplexity of test documents treated as one large document.*

	Doc 2001	**Test Docs as One**
Per word Perplexity	4310.830	2687.093

Table 1: *Perplexities for fixed* $\alpha = 5$

4 Exercise (d)

The posterior probability of a category in a Bayesian mixture of multinomials model is given by the expected value of the corresponding Dirichlet distribution:

$$p(\theta_k|\mathbf{z},\alpha) = \frac{\alpha + c_k}{\sum_{i=1}^{K} \alpha + c_i} = \frac{\alpha + c_k}{\alpha K + N} \tag{5}$$

Where c_k is the number of documents assigned to category k, K is the number of categories, and $\sum_{i=1}^{K} c_i = N$ is the number of documents.

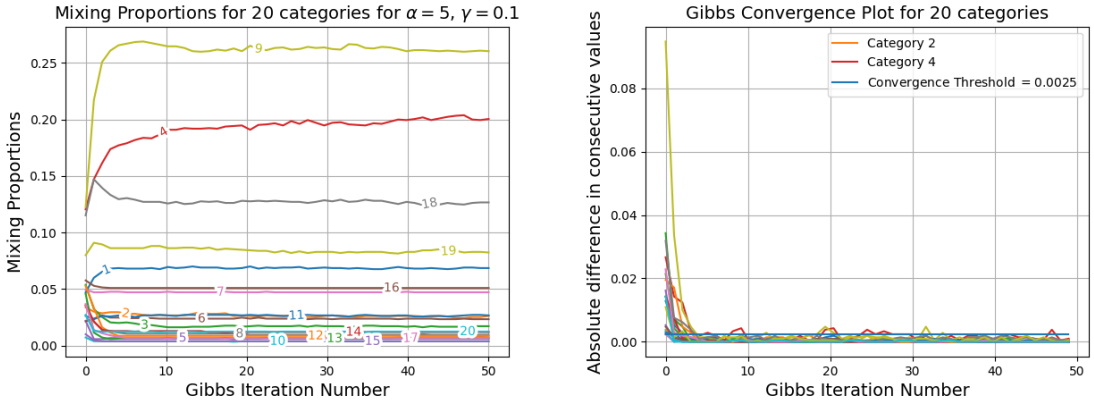

Figure 11: *Mixing Proportion Convergence for 20 categories.*

By changing the initialisation of the Gibbs sampler through a random seed, Figure 12 shows different final configurations for the posterior mixing proportions are achieved, possibly because the posterior distribution is likely multi-modal, so different initialisations cause the sampler to converge to different local optima. Due to the posterior's high-dimensionality, it is impossible to determine whether a global maximum has been reached, and it can only be concluded that the sampler reaches a likely state.

	Seed = 1	**Seed = 10**	**Seed = 100**
Per word Perplexity	2084.923	2113.577	2102.488

Table 2: *Per word perplexity for entire test set, for fixed* $\alpha = 5, \gamma = 0.1$

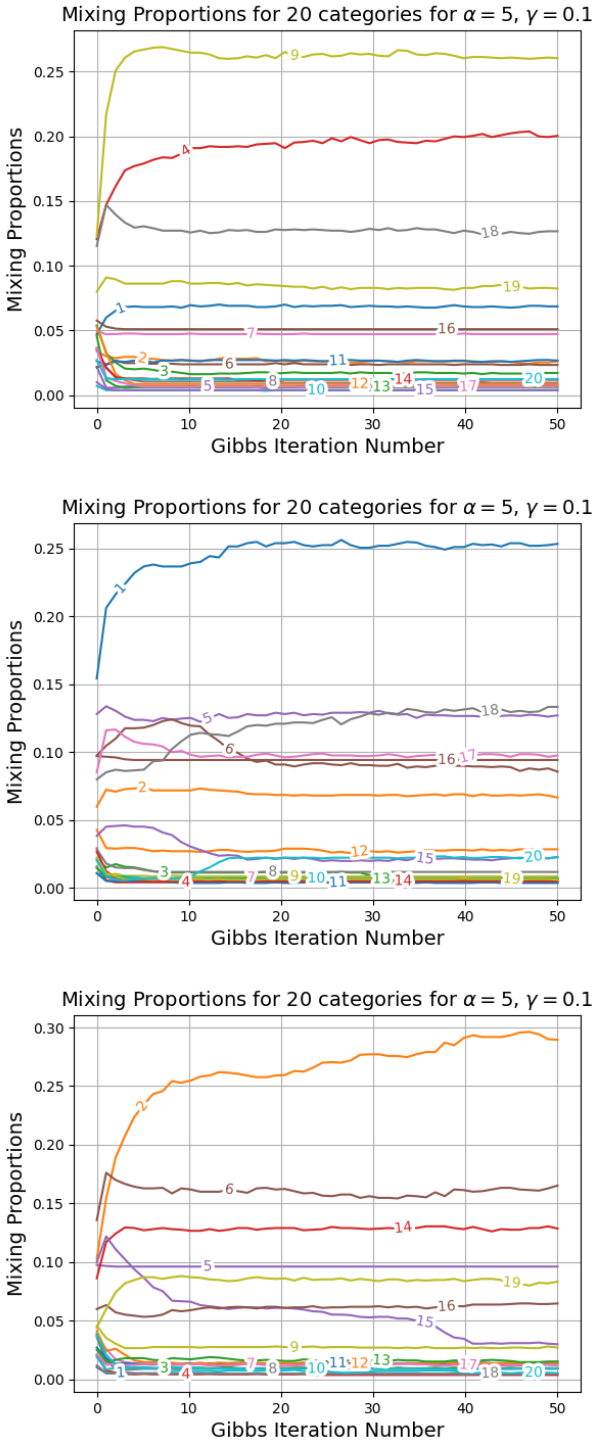

Figure 12: *Mixing Proportion Convergence for 20 categories with seeds 1, 10, 100.*

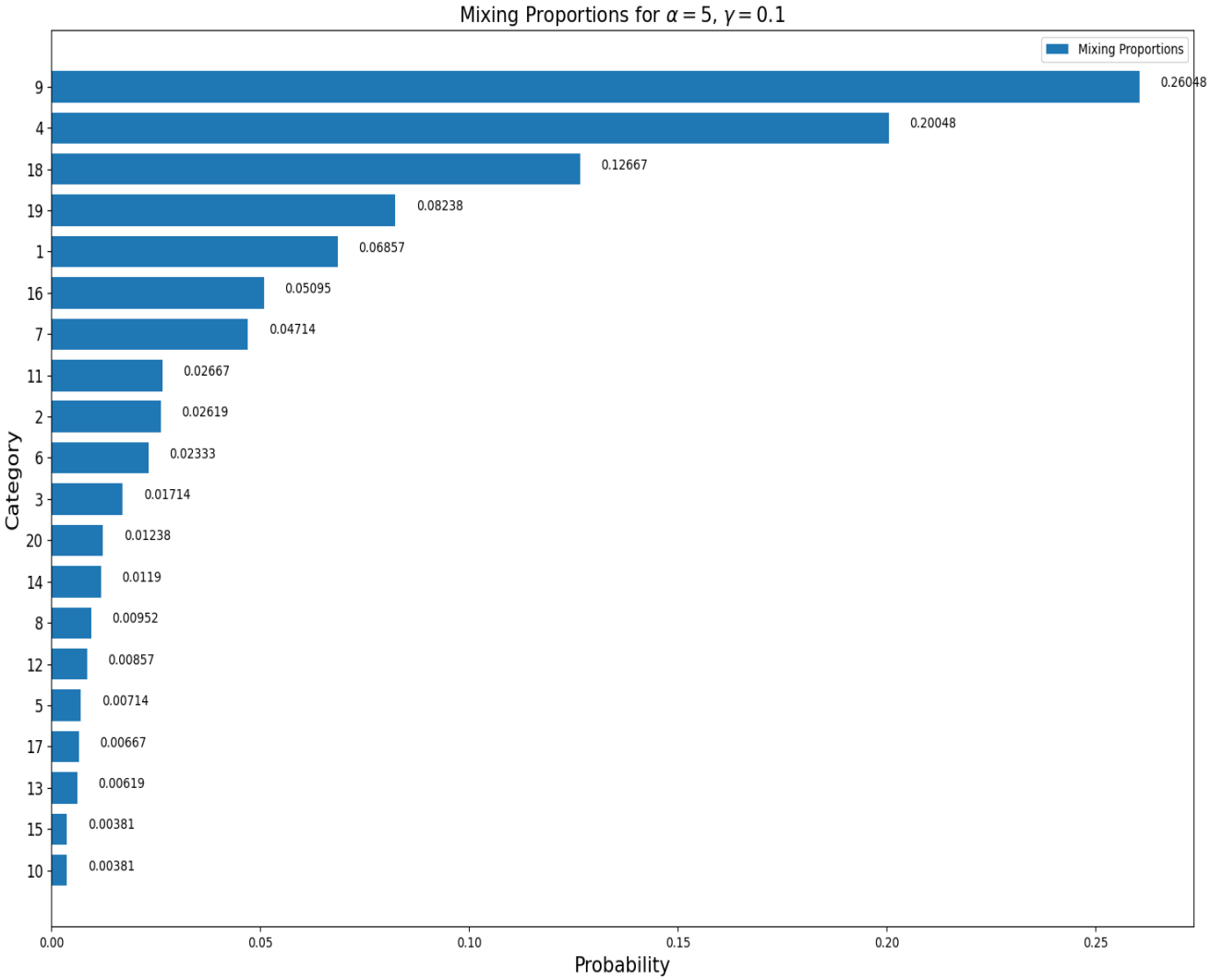

Figure 13: *Mixing Proportion Barplot for 20 categories for seed* $= 1$.

Figure 13 above shows that 17 out of 20 categories are assigned a probability of $\approx 2.3 \times 10^{-3}$. Requiring a mixture assignment probability of at least 5×10^{-3} suggests the number of categories for the data should be closer to $K = 3$. However, the perplexity for the test set under this model is 2235.936, which is $\approx 5.79\%$ higher than the perplexity when $K = 20$, so this suggests, nonetheless, $K = 20$ categories is a better model. To determine the convergence of the Gibbs sampler, Condition 6 can be met, where i is the Gibbs iteration.

$$|\theta_{k,i+1} - \theta_{k,i}| < \epsilon \tag{6}$$

Both plots on the right-hand side of Figure 11 show that the criteria for at least one of the mixing proportions oscillates with low amplitude about the threshold. Increasing the number of Gibbs iterations does not remove this oscillation, so the invariant distribution is reached after 50 iterations, although Figure 11 suggests 20 iterations are sufficient too.

```
    mix_prop_evols[iter, :] = [(alpha + sk_words[i]) / (alpha * K + np.sum(sk_words)) \
                               for i in range(K)]
diff_mix_prop_evols = np.zeros((num_iters_gibbs - 1, K))
```

```
    for i in range(num_iters_gibbs - 1):
        diff_mix_prop_evols[i, :] = np.abs(mix_prop_evols[i + 1, :] - mix_prop_evols[i, :])
```

Source Code 4: Code for calculating mixing proportions and absolute differences.

5 Exercise (e)

In the LDA model, each document is modelled as a mixture of topics, relaxing the unrealistic assumption of the Bayesian mixture of categories model that each document belongs to a single category. This comes at the expense of an exponential increase of the number of latent variables z_{nd}. Consequently, there is a posterior over topics for every document, such that the probability a document d is assigned to a category k is:

$$p(\theta_{d,k}|\alpha, \boldsymbol{Z}) = \frac{\alpha + c_k}{\sum_{i=1}^{K}(\alpha + c_i)} = \frac{\alpha + c_k}{\alpha K + N_d} \tag{7}$$

Where K is the number of topics, c_k is the number of words in document d assigned to class k, and hence N_d is the number of words in document d.

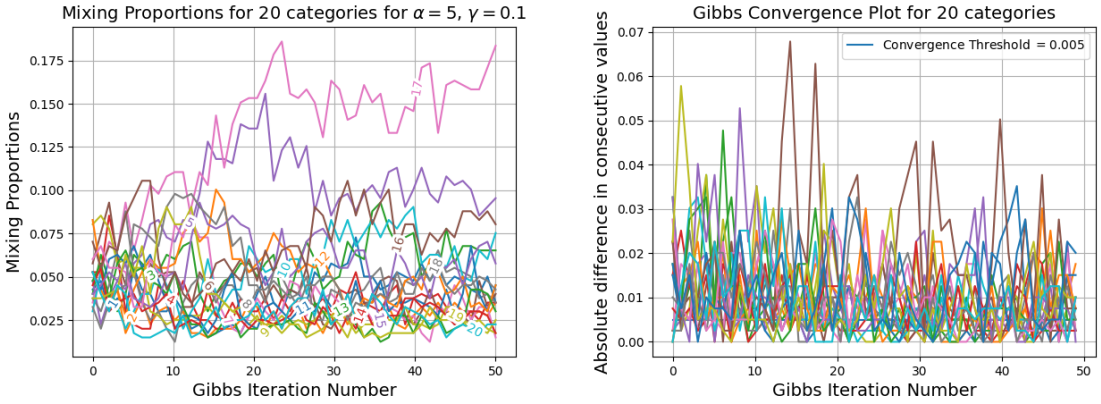

Figure 14: *Mixing Proportion Convergence for document 1 for 20 categories.*

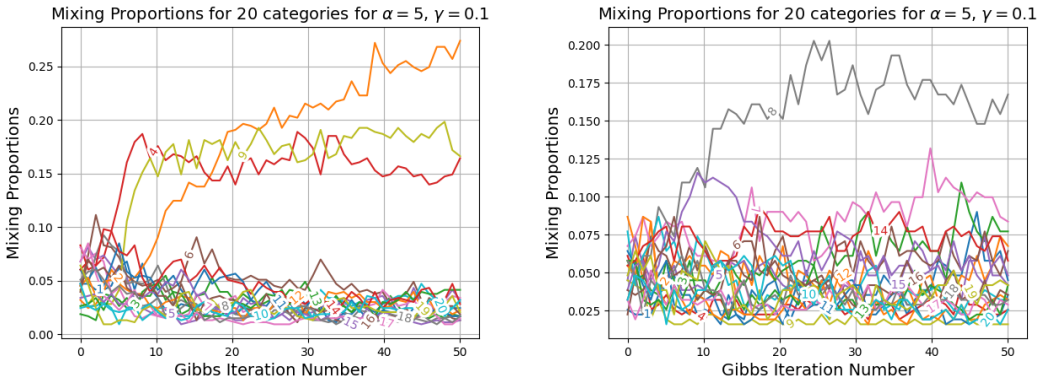

Figure 15: *Mixing Proportions for 20 categories. From left to right: document 17 and 900.*

Figure 14-15 show 3 categories dominate the classification of document 17, suggesting the LDA model is a better suit for document classification, since the Bayesian mixture model would have assigned it a single topic.

Figure 14 shows that the posterior for document 1 fails to converge after 50 iterations, and increasing the iteration number does not stabilise the mixing proportions enough to meet the convergence criteria. By considering the documents separately, it is difficult to determine whether *all* the document topic posteriors have converged. Therefore, Figure 16 below combines all training documents, showing Topic *9* is the most likely, reflecting the rich get richer property of the Gibbs sampler.

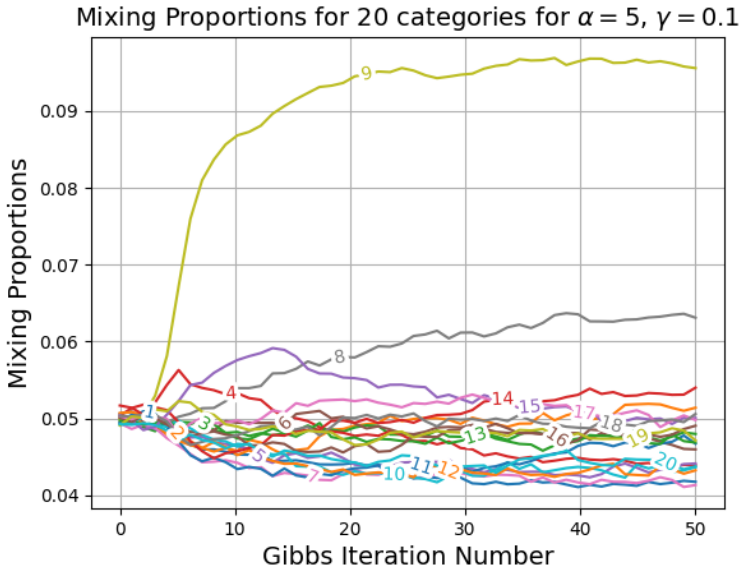

Figure 16: *Mixing Proportions for entire training set for 20 categories .*

However, Table 3 shows that the test set perplexity is lowest for the LDA model. Thus, it doesn't matter convergence isn't fully reached, because the model already outperforms previous models. This also suggests it is the best model to explain the unseen data after training, reflecting the fact that the model's superior flexibility for classification of topics allows for a more accurate model.

	Maximum Likelihood	**Predictive Bayes**	**BMM (seed=1)**	**LDA (seed=1)**
Per word Perplexity	∞	2687.093	2084.923	1909.871

Table 3: *Per word perplexity for entire test set, for fixed* $\alpha = 5, \gamma = 0.1$

The word entropy for each topic refers to the uncertainty of a topic generating a particular word. Therefore, given a generated test word w^* from a document d, the probability it is drawn from a topic k is given by the categorical distribution of topic assignments to a word:

$$\beta_k^{(w^*)} = p(w_d^*|z_{n,d} = k) = \mathbb{E}\left[Cat(\boldsymbol{\beta_k})\right] = \frac{\gamma + \tilde{c}_k^*}{\alpha K + N} \tag{8}$$

Where $\tilde{c}_k$ is the number of times word w^* is assigned to class k over all documents, and N is the total number of words assigned to category k. The word entropy, with units of *nats* when using $\log_e$, for a topic k with d_k words assigned to it is therefore:

$$H(k) = \sum_{i=1}^{d_k} \left(-\beta_k^{(i)} \log_2 \beta_k^{(i)}\right) \tag{9}$$

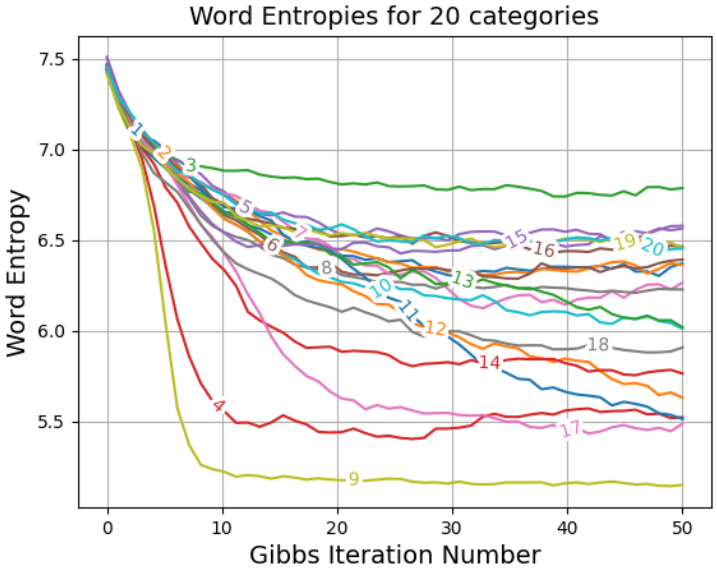

Figure 17: *Word Entropies for 20 categories.*

Figure 17 shows that the word entropies for each topic decrease with the Gibbs iteration number, reflecting the fact that each topic specialises on a set of words and hence the range of possible words is generated is lower, reducing uncertainty. The stabilisation of word entropies also suggests that the Gibbs sampler has in fact converged after 40 iterations.

Words: 1000