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

Module	4F13	Title of report	Coursework 3: Latent Dirichlet Allocation

Date submitted: 02/12/2022	Assessment for this module is ☑ 100% / ☐ 25% coursework of which this assignment forms 33 %

UNDERGRADUATE STUDENTS ONLY		POST GRADUATE STUDENTS ONLY			
Candidate number:	5554D	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:

4F13 Probabilistic Machine Learning
Coursework 3: Latent Dirichlet Allocation

CCN: 5554D
Words: 999

December 2, 2022

1 Task A

The Maximum Likelihood (ML) multinomial over the words in the training set $\mathcal{A}$ can be derived. The log-likelihood is given as

$$\mathcal{L}(\pi) = \sum_{i=1}^{M} k_i \log(\pi_i) \tag{1}$$

where k_i - # word i and M - # unique words. The ML estimate is derived by maximising the log-likelihood with a Lagrange multiplier to impose the sum-to-one constraint for probabilities:

$$\begin{aligned} \frac{\partial}{\partial \pi_m} \left\{ \sum_{i=1}^{M} k_i \log(\pi_i) + \lambda \left[1 - \sum_{j=1}^{M} \pi_j \right] \right\} &= \frac{k_m}{\pi_m} - \lambda = 0 \\ \pi_m &= \frac{k_m}{\lambda} \\ \pi_m^{ML} &= \frac{k_m}{N} \end{aligned} \tag{2}$$

where N is the total number of words. Therefore, the ML estimate is essentially the empirical frequency of each word.

The twenty most common words are presented in Figure 1.

The greatest possible test set log probability is $\log(0.0140972) = -4.262$, occurring when the document contains a singular "bush". The lowest test set log probability is $-\infty$, which occurs when the test set contains a word that is not in set $\mathcal{A}$. This is not suitable - a document with a word not included in the train set should not have a probability of 0, as it clearly exists. The training set is not representative of the population of all words.

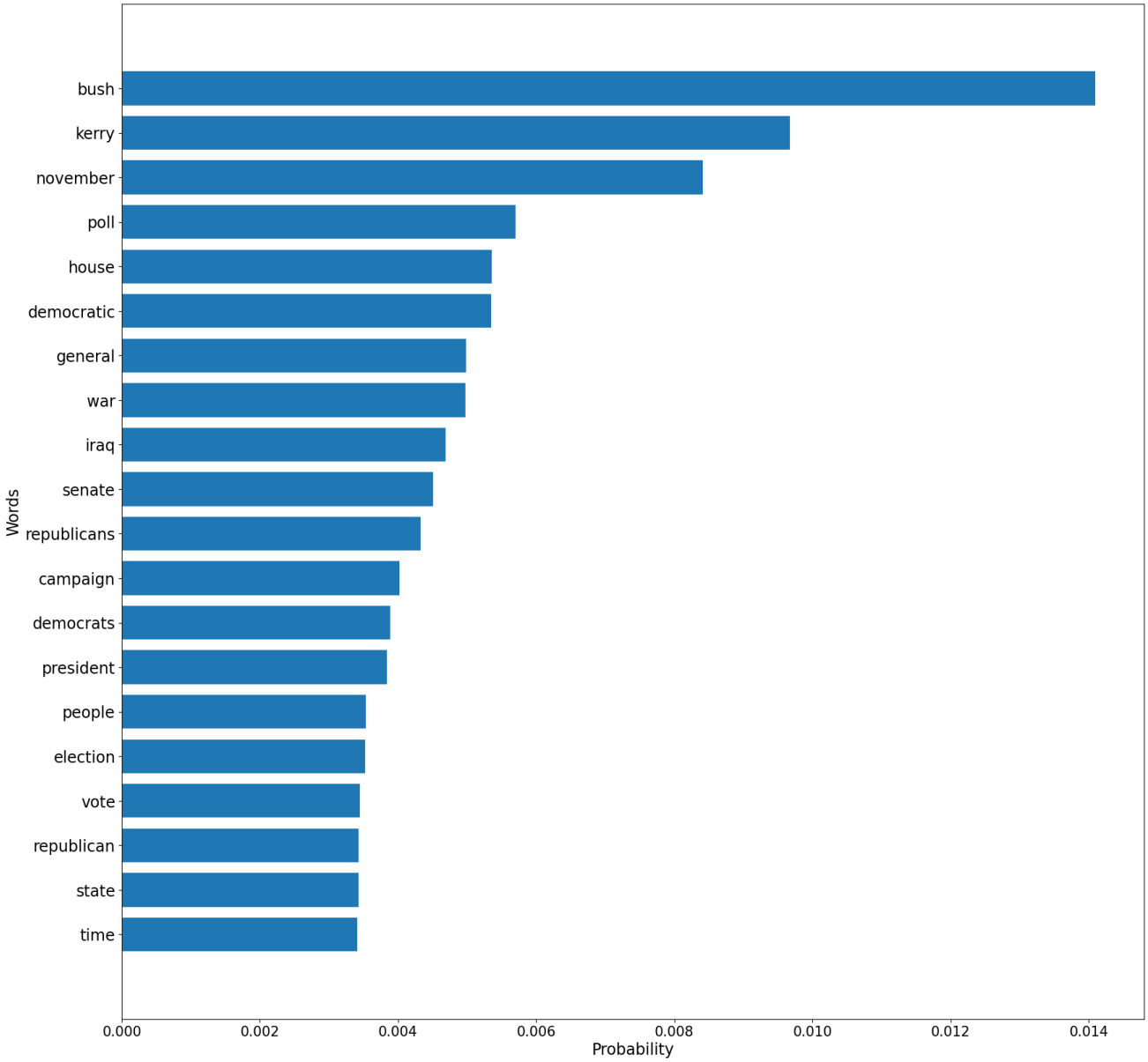

Figure 1: Histogram showing the 20 most common words in the training set based on their ML probabilities.

Listing 1: Code to calculate Maximum Likelihood solution

```
counts = np.zeros(W)
for i in range(N_1):
    counts[A[i,1]-1] += A[i,2]
ml = counts/N_w
```

2 Task B

The predictive word probability for ML inference is

$$p_{ML}(m|\mathcal{A}) = \frac{k_m}{N} \tag{3}$$

To combat the issues of ML estimation, Bayesian inference can be conducted with a symmetric Dirichlet prior, thus yielding a Dirichlet posterior. The predictive probability under this model is given by

$$p_B(m|\mathcal{A}) = \frac{\alpha + k_m}{\sum_{i=1}^{M}(\alpha + k_i)} = \frac{\alpha + k_m}{M\alpha + N} \tag{4}$$

where α is the concentration parameter of the distribution - essentially, a pseudo-count of each word. When α is small, the a-priori count is small and the posterior is heavily impacted by the likelihood (3). As $\alpha \to 0$, the Bayesian solution approaches the ML estimate. When α is very large, the Bayesian posterior is dominated by the prior, and as $\alpha \to \infty$, the posterior approaches a uniform distribution, $p(m) \to 1/M$ (i.e. rare words are equally likely as common words). The prior allows for words that do not appear in the training set to have non-zero probabilities, without changing the ranking of words. These properties are illustrated in Figure 2. Rare words with a count $k_m < N/M$ have a greater probability under the Bayesian model, whilst common words ($k_m > N/M$) have a reduced probability:

$$\begin{aligned} p_{ML}(m|\mathcal{A}) &< p_B(m|\mathcal{A}) \\ \frac{k_m}{N} &< \frac{\alpha + k_m}{M\alpha + N} \\ \Rightarrow k_m &< \frac{N}{M} \end{aligned} \tag{5}$$

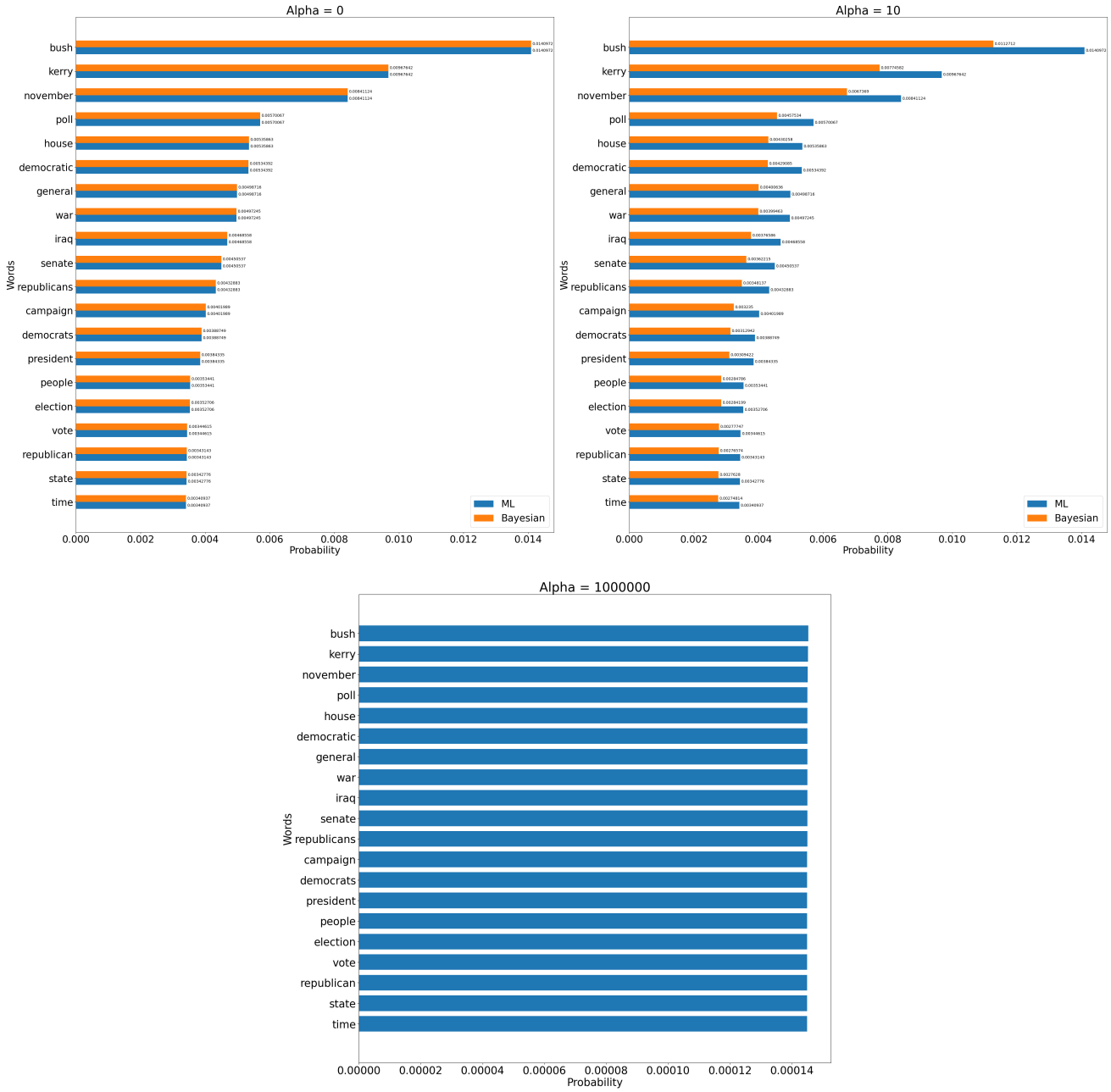

Figure 2: Histograms showing the Bayesian probabilities of the 20 most common words for different values of α: Top Left - $\alpha = 0$, Top Right - $\alpha = 10$, Bottom - $\alpha = 1000000$.

Listing 2: Code to apply Dirichlet prior.

```
post_counts = counts + alpha
probs = np.true_divide(post_counts, np.sum(post_counts))
```

3 Task C

Figure 3 shows the log probability for Bayesian inference on document 2001 in test set $\mathcal{B}$ against α. For $\alpha = 10$, the log probability is -3680.7. When computing the log probability, the categorical distribution function is used, as each document in a *sequence* of words (categorical random variables) - the ordering of words is important! When using a specific selection of words, certain sequences are much more probable than others.

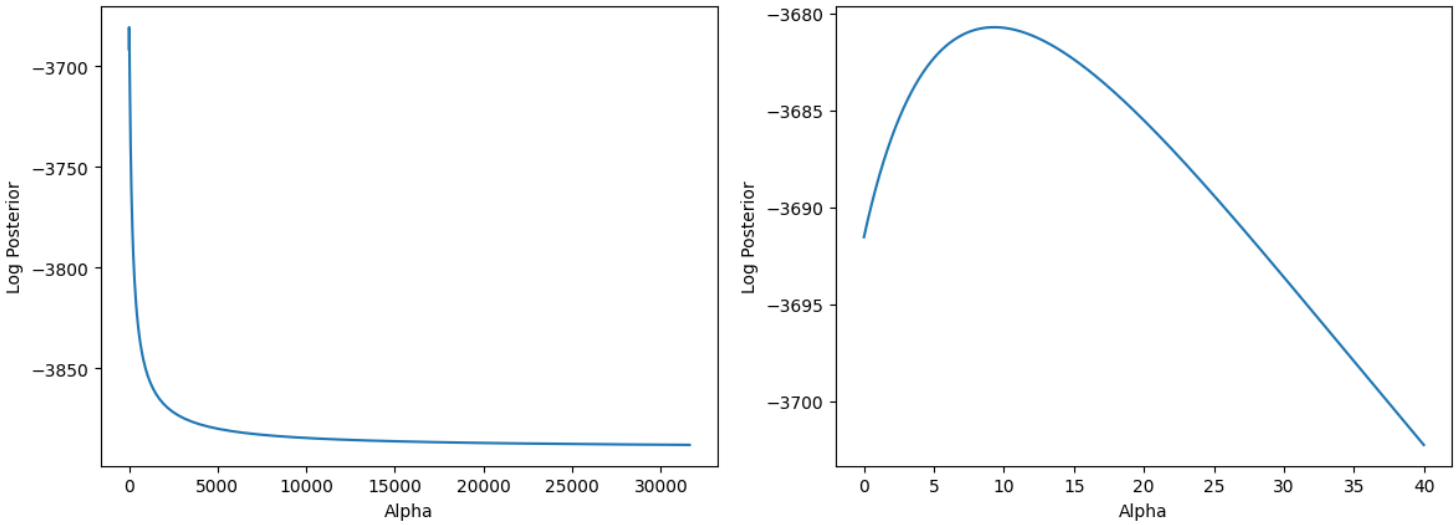

Figure 3: Log predictive probability for Bayesian inference on test document with ID 2001 against the concentration parameter, α, for different ranges of α: Left - $0 < \alpha \leq 10^{4.5}$, Right - $0 < \alpha \leq 40$.

The per-word perplexity for a given document is given by (6), with log-likelihood l and n words. A lower perplexity indicates a better fit of the test data by the predictive distribution.

$$\text{perplexity} = \exp\left(\frac{-l}{n}\right) \tag{6}$$

For a uniform multinomial distribution with probability π_0, the log-likelihood of a document in $\log(\pi_0^n) = n\log(\pi_0)$, and the perplexity is constant for all documents. As $\alpha \to \infty$, the Bayesian model approaches a uniform predictive distribution with $\pi_0 = 1/6906$. The perplexity in this case is:

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

The perplexity as a function of α on document 2001 is shown in Figure 4. As α increases, the perplexity reduces to a minimum of 4295.25 at $\alpha = 9.30$, and then increases to 6906 (asymptote), as the Bayesian predictive distribution approaches a uniform distribution. For small α, the model is poor as the probabilities for words unseen in the training data are too small, and for large α, inference is poor because the probabilities for rare words are too large. At $\alpha \approx 9$, the model perfectly balances the prior and likelihood.

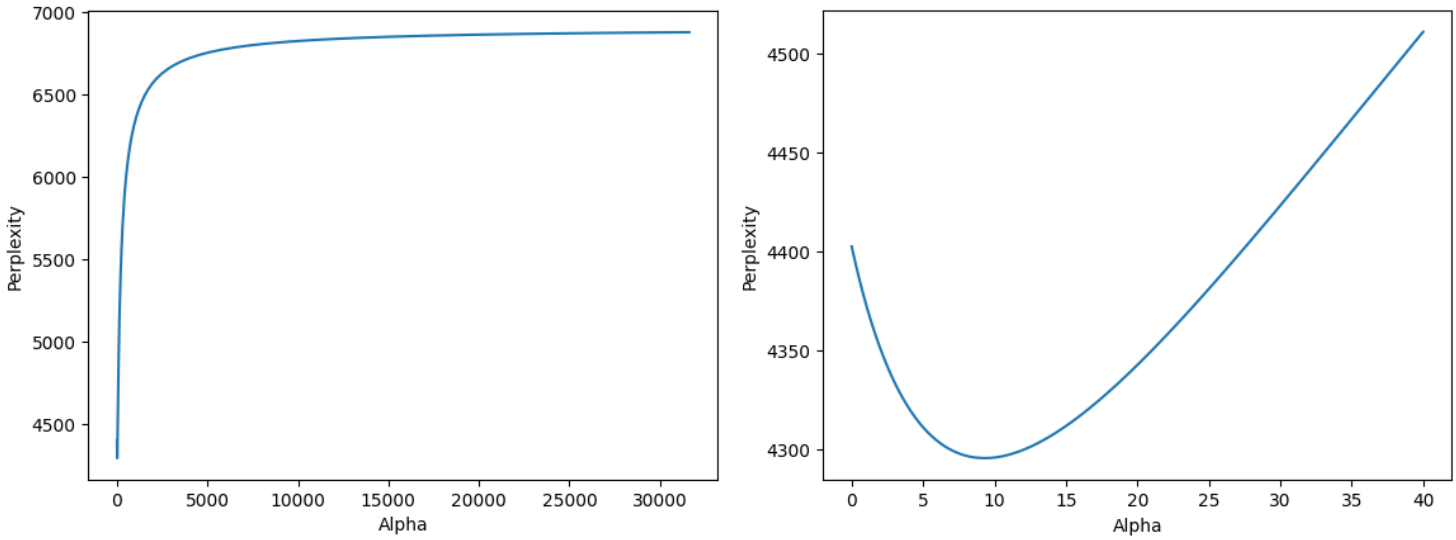

Figure 4: Perplexity for Bayesian inference on test document with ID 2001 against the concentration parameter, α, for different ranges of α: Left - $0 < \alpha \leq 10^{4.5}$, Right - $0 < \alpha \leq 40$.

Figure 5 shows the perplexities across all documents in set $\mathcal{B}$ for four values of α: 0.1, 10, and 1000 and 1000000. Different documents have different perplexities (if α is not very large) as the number of common and rare words vary - each document is unique. Documents with a larger number of common words have higher (log) probabilities and hence lower perplexities. For large α, the Bayesian model is close to a uniform distribution, and therefore the perplexity is essentially constant at 6906. The mean perplexity over all documents in $\mathcal{B}$ for $\alpha = 10$ is 2883.8 - therefore, document 2001 contains a higher frequency of rare words than most documents in $\mathcal{B}$.

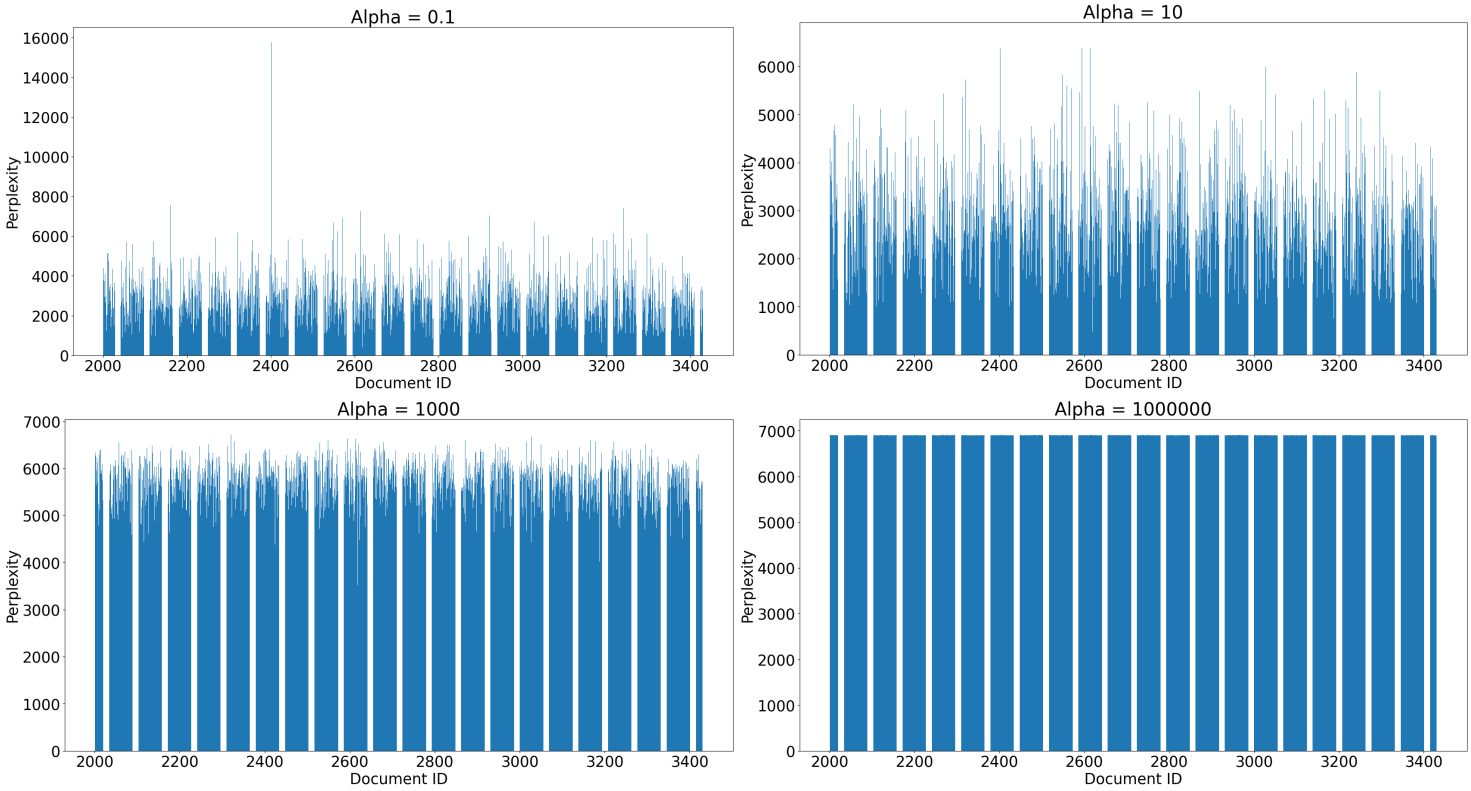

Figure 5: Perplexity on every document in test set $\mathcal{B}$ for different values of α: Top Left - $\alpha = 0.1$, Top Right - $\alpha = 10$, Bottom Left - $\alpha = 1000$, Bottom Right - $\alpha = 1000000$.

Listing 3: Code to calculate log posterior and perplexity for a range of values of α.

```
for alpha in alphas:
```

```
    post_counts = counts + alpha
    probs = np.true_divide(post_counts, np.sum(post_counts))
    log_post_freqs = np.log(probs)
    log_probs.append(np.dot(doc_word_counts, log_post_freqs))
log_probs = np.array(log_probs)
perplexities = np.exp(-log_probs/n)
```

4 Task D

A Bayesian Mixture Model (BMM) can be employed by introducing K distinct categories/topics; each document is assigned to a category, using a latent parameter $z_d \in \{1, 2, ..., K\} \sim \text{Cat}(\theta)$. Each document k is modelled using a topic component β_k. Dirichlet priors, $\text{Dir}(\alpha)$ and $\text{Dir}(\gamma)$ are applied to θ and β, respectively[1]. Gibbs sampling is utilised, as calculating the exact distributions is intractable. The mixing proportions for $K = 20$ categories as functions of Gibbs iteration are shown in Figure 6, with $\alpha = \gamma = 1$, for three different random initialisations of the Gibbs sampler. The mixing proportions are estimated as:

$$\theta_k \approx \frac{\alpha + \sum_{d=1}^{D} \mathbb{1}(z_k = d)}{\alpha K + D} = \frac{\alpha + c_k}{\alpha K + D} \tag{7}$$

where D - # documents, and c_k - # documents assigned to category k.

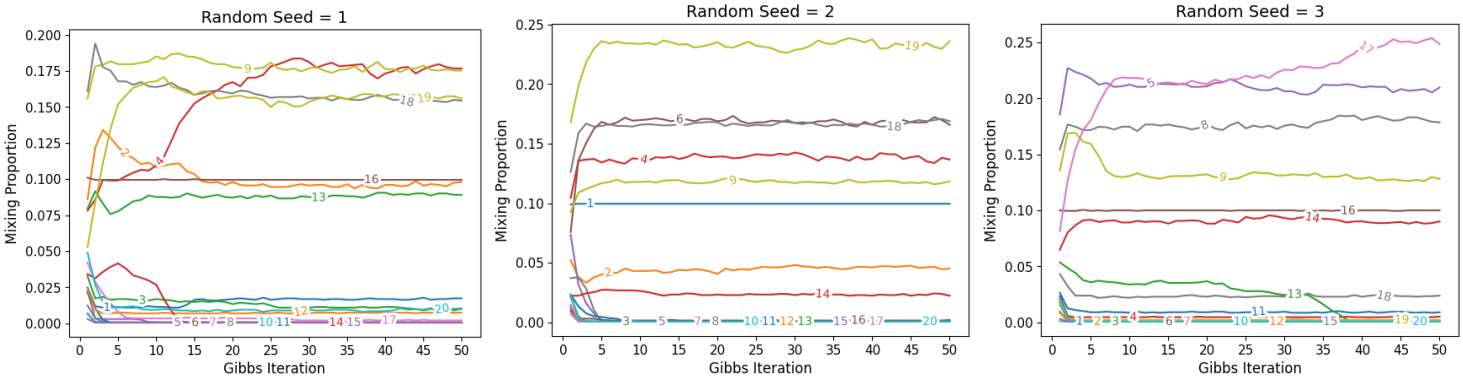

Figure 6: Mixing proportions for a mixture of multinomials model with $K = 20$ categories and $\alpha = \gamma = 1$ as a function of the Gibbs iteration for three different random seeds: 1 (left), 2 (centre), 3 (right).

Most of the mixing proportions seem to stabilise within 50 iterations. The final posterior (stationary distribution) of each mixture component is different in each. Therefore, it can not be confirmed whether the sampler converges to the global optimum (/true posterior), but instead to a local optimum. It is likely that the posterior space is highly multi-modal, resulting in the sampler getting stuck at local minima, with the rich-get-richer property of the collapsed Gibbs sampler enforcing this.

Many of the categories contain very few documents (i.e. small mixing proportions), suggesting that the number of categories, K, may be unnecessarily large.

The perplexities over all documents in $\mathcal{B}$ for the three BMMS are given in Table 1, showing improved performance over simple Bayesian inference.

Random Seed	1	2	3
Perplexity	2022.6	2020.8	2035.4

Table 1: Perplexities of all documents in set $\mathcal{B}$ for Bayesian Mixture Models after 50 Gibbs iterations for three different random initialisations of the Gibbs sampler.

Listing 4: Code snippet added to bmm.py to obtain mixing proportions at each Gibbs iteration.

```
mix_prop_evols[iter, :] = [(alpha + sk_docs[i]) / (alpha*K + np.sum(sk_docs)) for i in range(K)]
```

[1]Note a change in notation. Here, γ is the parameter over mixture components and is equivalent to α from the previous two sections, and α is now a new parameter over words.

5 Task E

Latent Dirichlet Allocation (LDA) is a generalisation of BMM, where documents are modelled as a mixture of topics, rather than a single category. Each word $w_{n,d}$ is assigned a topic $z_{n,d} \sim \text{Cat}(\theta_d)$. Each document's topic posterior is estimated as:

$$\theta_{d,k} \approx \frac{\alpha + \sum_{n=1}^{N_d} \mathbb{1}(z_{n,d} = k)}{\alpha K + N_d} \tag{8}$$

The evolutions of topic posteriors averaged over all documents, for $K = 20$ and $\alpha = \gamma = 1$, are shown below. Again, there are significant differences in the final posteriors between different seeds. However, unlike BMM, the models have not fully converged within 50 iterations, which is expected as the number of words is much larger than the number of documents - 50 Gibbs sweeps are inadequate for convergence to the true posterior.

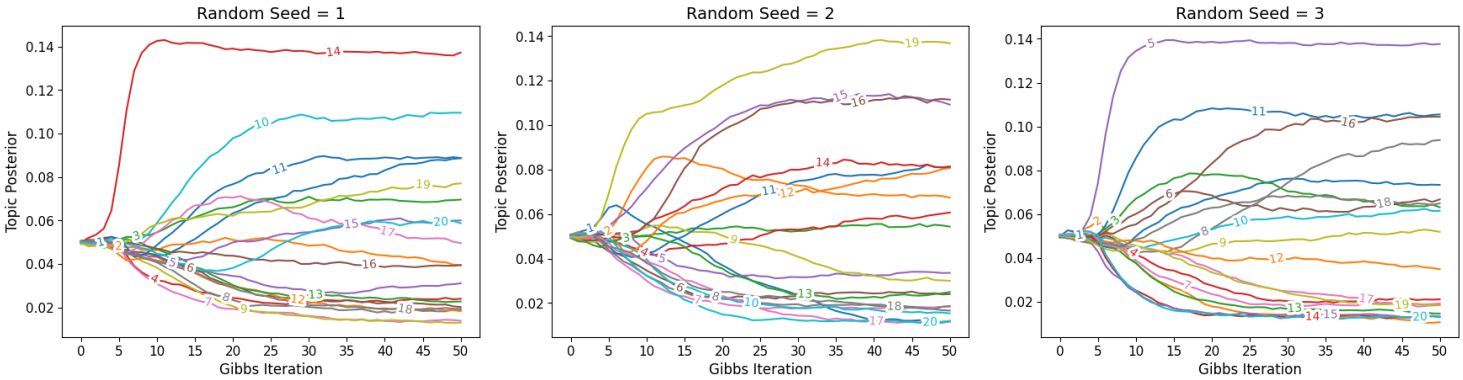

Figure 7: Topic posteriors for LDA against Gibbs iteration for three different (random) initialisations of the sampler. Random seeds: 1 (left), 2 (centre), 3 (right).

The perplexities over all documents in $\mathcal{B}$ are shown in Table 2, with the average perplexity being 1907.0, whilst the average perplexities of BMM and Bayesian inference were 2026. and 2883.8, respectively. Therefore, as expected, LDA fits unseen data the best, being able to classify individual words. 50 Gibbs iterations is thus sufficient.

Random Seed	1	2	3
Perplexity	1899.6	1907.7	1913.8

Table 2: Perplexities of all documents in set $\mathcal{B}$ for Latent Dirichlet Allocation over 50 Gibbs iterations for three different random initialisations of the Gibbs sampler.

Figure 8 shows a general decrease in word entropy (in bits) with Gibbs iteration, which shows that each category specialises towards a smaller vocabulary with higher probabilities (hence smaller uncertainty/entropy). Categories with smaller entropies are more specialised, containing fewer words with higher probabilities, whilst topics with larger entropies are more general. The entropies seem to have stabilised within 50 iterations. The word entropy is upper-bounded by $\log_2(6906) \approx 12.75$bits.

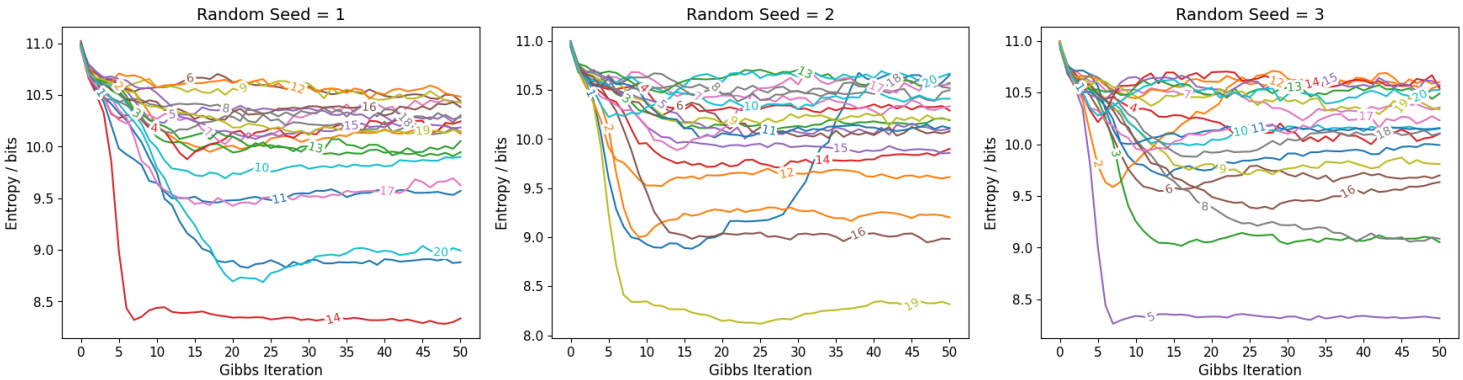

Figure 8: Word entropies (in bits) for each topic in Latent Dirichlet Allocation over 50 Gibbs iterations for three different random initialisations of the Gibbs sampler. Random seeds: 1 (left), 2 (centre), 3 (right).

Listing 5: Code snippet added to lda.py to compute mixing proportions and word entropies at each iteration.

```
mix_prop[iter+1, :] = sk
for k in range(K):
    word_entropy[iter+1, k] = entropy(swk[:, k], base=2)
```