CUED - Engineering Tripos Part IIB 2023-2024 **Module Coursework**

Module	4F13	Title of report	Coursework 3
Date submitted: 1/12/2023		Assessment for this module is ☑ 100% / ☐ 25% coursework of which this assignment forms _33.33__ %	
UNDERGRADUATE and POST GRADUATE STUDENTS			
Candidate number:	5553C	☑ Undergraduate ☐ Post graduate	

Feedback to the student ☐ **See also comments in the text**		Very good	**Good**	Needs improvmt
C O N T E N T	**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:			
P R E	**Attention to detail, typesetting and typographical errors** Is the report free of typographical errors? Are the figures/tables/references presented professionally?			

1 Exercise A

```
for d in range(D):
    training_documents = np.where(A[:, 0] == d+1)
    w = np.array(A[training_documents, 1])  # number of unique words
    c = np.array(A[training_documents, 2])  # counts of words
    word_counts[w-1] += c  # number of times w is assigned
word_freq = word_counts / np.sum(word_counts).astype(int)
```

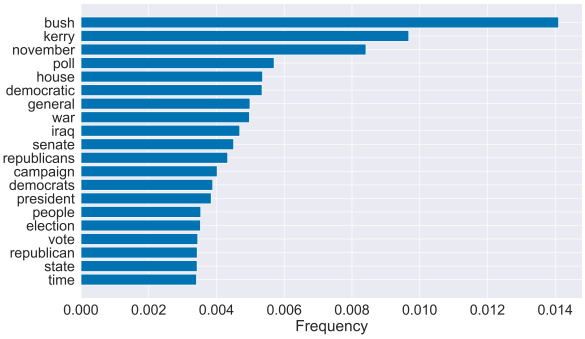
Figure 1: Code for obtaining the maximum likelihood multinomial over words.

The most simple model used to calculate the probability of a document uses the aggregation of all the documents and simply models the global word frequency model (a multinomial distribution). So to achieve the maximum likelihood multinomial over words, we set each parameter equal to the rate of occurrences of the corresponding word, as implemented in Figure 1 and shown in Equation 1. W is the number of unique words in the training set, and k_i is the number of occurrences of each word.

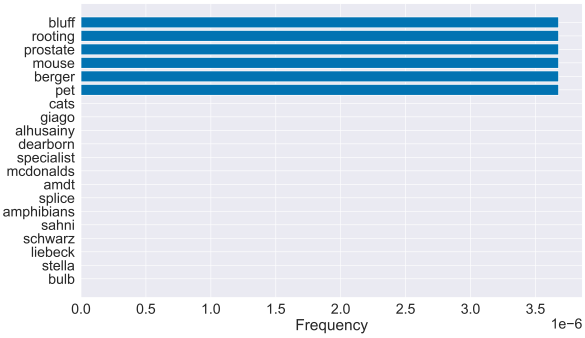
$$\pi_i^{ML} = \frac{k_i}{\sum_{i=1}^{W} k_i} = \frac{k_i}{N} \tag{1}$$

The highest test set log probability that could be returned using this model, is from a document containing a single word 'bush', as it is the most frequent word as shown in Figure 2a. This would yield $LP_{max} = log(0.0141) = -1.85$. The lowest test set log probability could be yielded from a document containing a word not seen in the training set. As observed in Figure 2b, these words have a resulting probability of 0, hence the lowest log probability is $LP_{min} = log(0) = -\infty$.

Using this simple model has three, major drawbacks for modelling documents. Firstly, the document type is not considered; realistically documents have topics which would alter the distribution of words, for example, 'bush' (the president) is extremely unlikely to be mentioned the most in a document about biology (see Sections 4 and 5). Secondly, as all the samples are drawn independently, our model suggests that the document containing 100 words is most likely when every single word is 'bush'; this is simply not true and can be resolved by considering the sequences of words. Lastly, our model is prone to overfitting, as it suggests that any word not in the test set, must simply not be probable, hence all documents containing such words are extremely unlikely with log probabilities equal to negative infinity.



(a) 20 most frequent words



(b) 20 least frequent words

Figure 2: 20 words with the highest and lowest frequencies from the ML model.

2 Exercise B

```
pseudo_counts = alpha * np.ones(W)
posterior_counts = word_counts + pseudo_counts
posterior_freq = posterior_counts / np.sum(posterior_counts)
```

Figure 3: Code implementing a symmetric Dirichlet prior to perform Bayesian inference.

Here, we explore the use of a prior and Bayesian inference in a document model to solve the last of the three shortcomings mentioned in Section 1. The prior to be used is a Dirichlet distribution, given in Equation 2.

$$p(\pi|\alpha) = \left(\frac{1}{\beta(\alpha)}\right) \prod_{i=1}^{W} \pi_i^{\alpha-1} \tag{2}$$

Using Bayes theorem, we obtain the distribution of the parameters given a dataset, as shown in Equation 3

$$\begin{aligned} p(\pi|\mathcal{D}) &= \frac{p(\pi|\alpha)p(\mathcal{D}|\pi)}{p(\mathcal{D})} \\ &\propto \prod_{i=1}^{W} \pi_i^{k_i+\alpha-1} \propto Dir(\pi; \alpha + \mathbf{k}) \end{aligned} \tag{3}$$

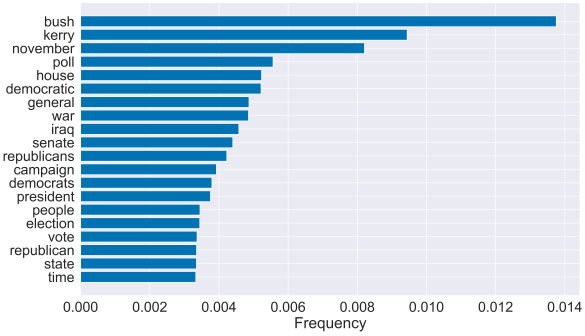
The probability of a new word is now obtained by averaging over all possible values of π_i. As seen in Equation 3, the posterior with a Dirichlet prior, also has a Dirichlet distribution; the mean of which is used to complete Equation 4.

$$\begin{aligned} p(w^* = i|\mathcal{D}) &= \int p(w^* = i|\pi_i)p(\pi_i|\mathcal{D})d\pi_i \\ &= E(\pi_i) \\ \Rightarrow p(w^* = i|\mathcal{D}) &= \frac{\alpha_i + k_i}{\sum_{j=1}^{W} \alpha_i + k_j} \end{aligned} \tag{4}$$

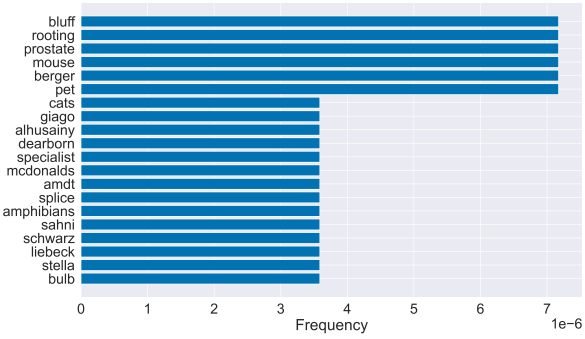
Using a symmetric Dirichlet prior, that is $\alpha_i = \alpha$ for all i, we obtain a clear solution for the maximum a posteriori multinomial over words (see Equation 5, note that W, N and k_i represent the whole training set). A symmetric prior is used, as we don't have prior information on individual words, just the distribution as a whole; lower α indicates that we believe most words will rarely appear, and certain words will be far more dominant, whereas an extremely high alpha would indicate that we believe the distribution of words is uniform. Note the prior does not alter rankings as it brings all word probabilities closer to the mean.

$$\pi_i^{MAP} = \frac{\alpha + k_i}{\alpha W + N} \tag{5}$$

This is seen in Figure 4; now with no zero word probabilities present, and hence less overfitting.



(a) 20 most frequent words



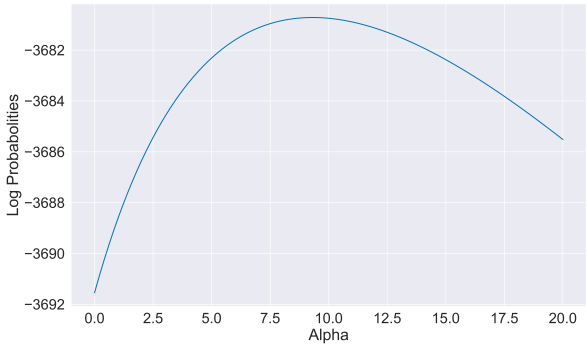
(b) 20 least frequent words

Figure 4: The 20 highest and lowest word frequencies from the MAP model. $\alpha = 1$.

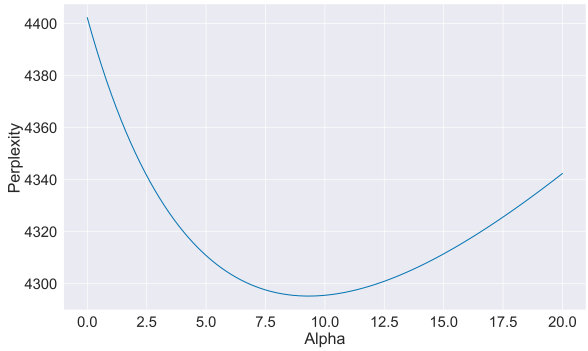
3 Exercise C

```
posterior_freq = posterior_counts / np.sum(posterior_counts)
log_probs = np.dot(test_counts, np.log(posterior_freq[test_doc[:,1]-1]))
perplexities = np.exp(-log_probs / np.sum(test_counts))
```

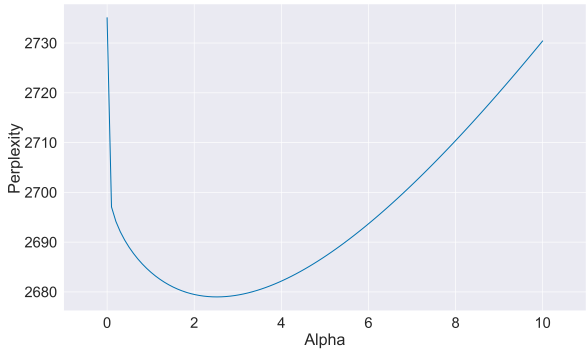
Figure 5: Code for computing log probabilities and perplexity for the test documents.



(a) Log probability of Document 2001



(b) Perplexity of Document 2001



(c) Average perplexity over all test documents

Figure 6: Log probabilities and perplexities for varying concentration parameters using the Bayesian Model.

When modelling the probability of a document, we take $\sum_{i=1}^{W} k_i$ samples sequentially and independently, where k_i is now the number of occurrences for each of the W unique words in a test document; hence the use of a categorical distribution function is appropriate, given in Equation 6.

$$\begin{aligned} p(\mathbf{w}|\pi^{MAP}) &= \prod_{i=1}^{W} \pi_i^{k_i} \\ ln(p(\mathbf{w}|\pi^{MAP})) &= \sum_{i=1}^{W} k_i ln(\pi_i) \end{aligned} \tag{6}$$

Equation 7 gives us the per-word perplexity, a normalised measure which allows us to compare the performance of different language models on different documents. We see in Figure 6 that this follows the same shape, but is mirrored, as the log probability.

$$Perplexity = \exp\left(\frac{-ln(p(\mathbf{w}|\pi^{MAP}))}{\sum_{i=1}^{W} k_i}\right) \tag{7}$$

In Figure 6 the perplexity initially falls, with increasing alpha, as our model is more able to incorporate words from the test set that it has not seen in the training set. It then starts to increase as alpha increases; this performance decrease is due to the prior setting the word probability distribution closer to uniform, which is not likely as most documents are dominated by a small subset of words, not evenly distributed. The substantial increase in perplexity from the average for Document 2001 (see Figure 6) indicates that the model doesn't believe

it is a particularly likely document relative to the rest of the test documents; The higher alpha for Document 2001 ($\alpha \approx 9.2$ vs the average across documents $\alpha \approx 2.4$) indicates that it indeed contains a larger amount of low-frequency words than usual; a higher alpha implies a preference for a flatter word probability distribution, which increases the probability of low-frequency words.

4 Exercise D

```
K = 20  # number of clusters
alpha, gamma = 10, 0.1
perplexity, swk, sk_docs_evolution = BMM(A, B, K, alpha, gamma)
```

Figure 7: Code for obtaining the evolution of the mixing proportions with each Gibbs sweep for the BMM.

Now we introduce the Bayesian mixture model (BMM). This first assigns documents to a latent variable, the mixture/topic, and now tracks how many times visible observations, the words, are assigned to that mixture, instead of just counting them in aggregation in a single mixture (the EM algorithm to find the optimal cluster assignments). Note, that alpha and gamma are the shape parameters for the Dirichlet distributions across categories and words respectively.

With the BMM, documents that previously weren't likely can now be very likely; words in particular documents may be rare in the whole test set, but within the assigned mixture they may be far more common.

The mixing proportions of topics can be seen in Figure 8. We use Gibbs sampling as the exact distribution is intractable. Therefore we know that we have converged on a stationary distribution once the burn-in phase has passed. After 50 sweeps we can comfortably say this has been reached as the mixing proportions remain relatively constant. For a greater degree of accuracy, we could explore the autocorrelation of samples with time-shifted samples; this would indicate whether thinning is required to ensure each sample is independent of the previous samples (If implemented more samples should be taken so the number of post-thinning samples is sufficient).

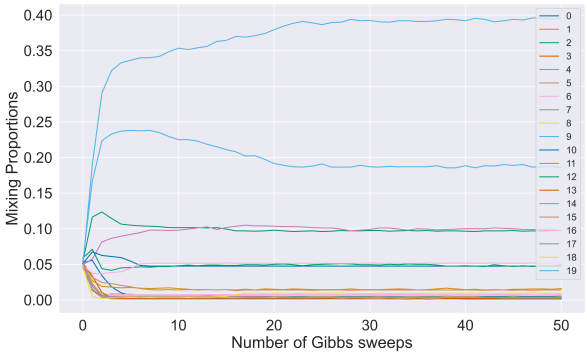


Figure 8: BMM document mixing proportions of topics against Gibbs sweeps. K = 20, alpha = 10 and gamma = 0.1.

The BMM now obtains a per-word perplexity = 2114, which is a 21% decrease from the Bayesian model's per-word perplexity = 2680.

5 Exercise E

```
K, alpha, gamma = 20, 10, 0.1
perplexity, swk, sk_evo, entrp_evo = LDA(A, B, K, alpha, gamma)
```

Figure 9: Code for using LDA.

A shortcoming of the BMM is that each document is assigned to a single topic, and each word is only drawn from a single topic, therefore documents that are split between several topics will be considered far less likely than a document with one dominant topic.

The latent Dirichlet allocation (LDA) model works around this by allowing each word to be sampled from more than one topic, and documents have a distribution over topics, instead of just being assigned to one. Now each word in a given document can be drawn from different topics that the document is distributed over.

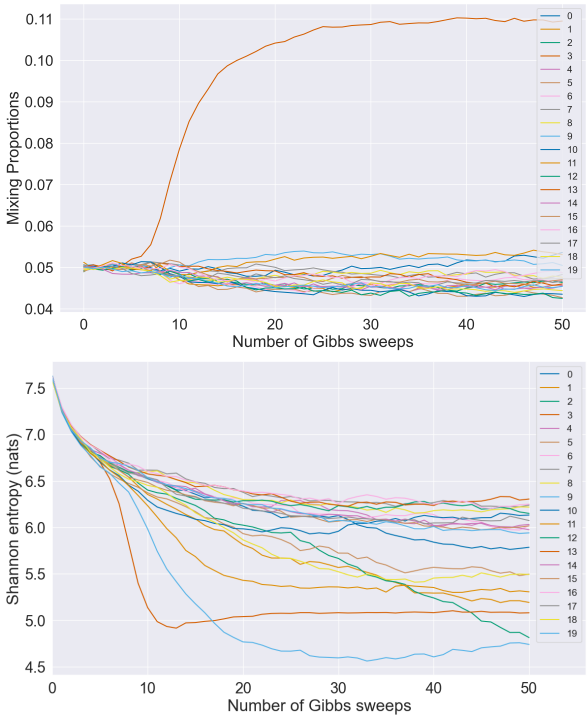


Figure 10: LDA word count mixing proportions of topics (top) and Shannon entropy of topic word counts (bottom) against Gibbs sweeps.

In Figure 8 we see that, for the BMM, few topics dominate the mixing proportions, with a substantial amount left with a mixing proportion close to zero. This doesn't happen in LDA, as can be seen in Figure 10, where we note the topic posteriors are far closer to the initialised state; this indicates that topics are more evenly distributed now that documents have a distribution over them, as less likely topics won't get ignored, and now hold weight in drawing samples.

The initial Bayesian model obtains a per-word perplexity of roughly 2680; using the BMM this is dropped significantly to 2114 as now a mixture of topics is considered; and lastly this is dropped even further to 2072 for LDA. The use of latent topics had the largest performance increase, with the changes introduced by LDA increasing performance to a lesser degree.

Figure 10 shows us the evolution of Shannon entropy of the unique word topic assignment counts with each Gibbs sweep (as base e is used the unit in the probability context is nats). Unique word counts are all initialised randomly and uniformly: each topic will roughly contain a uniform variety of unique words and corresponding counts, hence more disorder is present and is equal amongst topics. Then as we optimise the LDA, the word counts are distributed amongst more likely topics and the unique word variety within a topic decreases (for example types of art all counted under one topic); intuitively less 'surprise' is now present when observing the unique word counts in a particular topic, so the entropy of the topics decreases.