Regardless of the document ID, the words with the same index from different documents are grouped together and their numbers of occurrence are added up to give the total appearance of that distinct word in the dataset A. Adding up all word counts in the 3rd column of the dataset A give the total number of words.

The maximum likelihood multinomial for each distinct word is then obtained.

Based on this model, the training set A which does not contain the word has no count on that word. The probability of that word occurring is zero. Therefore, the test set probability is zero.

It implies if some words do not occur in the training set, the maximum likelihood multinomial model fails to estimate their probabilities of occurrences. Also, it cannot properly estimate the probability of a test set which contains these missing words.

The predictive word probabilities for these two types of inference for the common words look similar. However, the probabilities for rare words (including those non-existing words in training set A) are different, since the MLM model only counts words that exist in the dataset, but the Dirichlet model predicts by the mean of the posterior distribution. So, those words that have never existed in the data A will have zero probabilities in the MLM model but non-zero probabilities in the Dirichlet model. In general, the probabilities for all words are increased.

Using the Bayesian model with the symmetric Dirichlet prior, the log probability without and with combinatorial factor for the test document with ID 2001 are -3691 and -1691 respectively.

Here, the multinomial without the combinatorial factor is used. It is because the document is an article but not a vocabulary list that can draw any word from data and put them together without concerning grammar. Then the model without the factor is a better estimate in predicting words from article than the one with the factor which ignores the word sequencing completely.

For the log multinomial probability without combinatorial factor, the per-word perplexity for the document ID 2001 is 4399. The per-word perplexity over all documents in B is 2697.

In Figure 2, it shows different documents in B have different number of words which do not exist in training set A. This is important since those words which have relatively small probabilities close to zero lead to higher contribution to the perplexity. In Figure 3, it also shows different documents have different occurrences for each word.

So, the perplexities are different for different documents since they have different words distributions.

For a uniform multinomial, those 6906 distinct words have the equal probability of x . The per-word perplexity will become which is greater than that computed by the symmetric Dirichlet prior model. It indicates that, compared with the Dirichlet prior model, the uniform model performs badly in predicting samples.

The Gibbs sampler cannot converge and does not explore the stationary distribution since there are always some slopes for the mixing proportions greater or less than 0.

With different random seeds, the documents are distributed to different mixture components randomly. From Figures 4, 5 and 6, different topic indices are arranged to take the top rank according to their mixing proportions. It indicates the algorithm has no idea what topic each index represents. The mixing proportion separations between topics are not identical for different seeds, for example, the samplings with random seed 10 have the greatest separation between the 1st and 2nd rank mixture components in terms of mixing proportion, meanwhile, the ones with random seed 100 have the smaller separation. It shows the sampler does not explore the stationary distribution for the mixture components.

The Gibbs sampler is not able to converge within 20 sweeps, since it is observed that the posterior distributions of all topics are still not stable. Especially for the topic which was the top posterior at 6th iteration in Figure 7, its slope is significantly lower than 0.

After 20 Gibbs sweeps, the per-word perplexity calculated is 1716. It is lower than 2697, the one calculated in part c. It is possible that Ida.m successfully groups words together according to topics and limits the ways that the words can be used. Therefore, it gives a lower perplexity, which indicates it is a better model at predicting the sample.

From Figure 8, the perplexity is 1617, and the posteriors do not stabilize after 100 iterations. Especially for the topic which just reaches the 2nd top posterior at 100th iteration, its posterior is still growing upwards.

It may indicate that 20 sweeps are not adequate for the algorithm to converge since the perplexity is still decreasing, which means there are still some improvements allowed.

The entropies for all topics decrease and begin to converge as the slopes are less steep. This implies that the general probabilities of the distinct words rise, the distinct words carry less information (less possibilities to occur in any document and topic) because of the correct topic grouping. The word distribution becomes less chaotic.

The ‘Ida.m’ is run once, and the variable ‘b’ is stored as ‘store\_b’ since it contains the probabilities of all distinct words. The script is then modified by putting the following commands into the first ‘iter’ for-loop, and re-run again.

The ‘swk’ variable stores the record of which distinct word i is assigned to the topic j. Based on this record, for each topic j, I initialize a new entropy variable ‘en’ = 0, and for each distinct word i which is assigned to the topic j, I subtract the entropy value of the corresponding distinct word from ‘en’. Then the final value of ‘en’ is assigned back as an element (representing the overall entropy of topic j in a given iteration ‘iter’) to the ‘entropy’ matrix which stores all topics’ entropies for every iteration.

The following command is substituted into the first ‘iter’ for-loop,

It extracts the topic posteriors as a function of iteration.

The following command is substituted into the ‘iter’ for-loop,

It extracts the mixing proportions of all mixing components as a function of iteration.