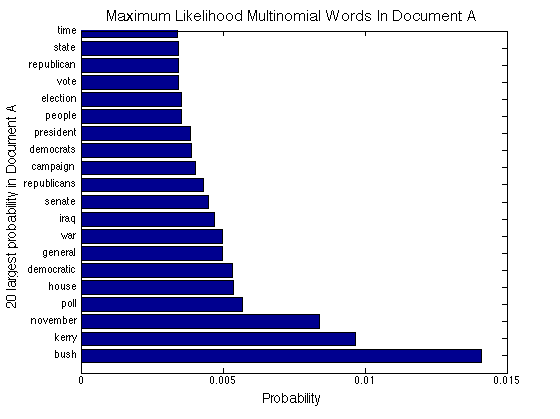
a) The number of documents, words and unique words in training matrix A, testing matrix B and the union of A and B were computed correspondingly in the table below.

|  |  |  |  |
| --- | --- | --- | --- |
| Number of documents, words and unique words in the matrix of A and B | | | |
|  | Documents | Words | Unique words |
| A | 2000 | 271898 | 6892 |
| B | 1430 | 195816 | 6870 |
| Union of A and B | 3430 | 467714 | 6906 |

|  |  |  |
| --- | --- | --- |
| %Training matrix A  %Counts for number of documents  dA = size(unique(A(:,1),'rows'),1);  %Counts for number of words  wA = sum(A(:,3));  %Counts for number of unique words  uwA = size(unique(A(:,2),'rows'),1); | %Testing matrix B  %Counts for number of documents  dB = size(unique(B(:,1),'rows'),1);  %Counts for number of words  wB = sum(B(:,3));  %Counts for number of unique words  uwB = size(unique(B(:,2),'rows'),1); | %The union of matrix A and B  C = union(A,B,'rows');  %Counts for number of documents  dC = size(unique(C(:,1),'rows'),1);  %Counts for number of words  wC = sum(C(:,3));  %Counts for number of unique words  uwC = size(unique(C(:,2),'rows'),1); |

b&c) The 20 largest probability words were illustrated in the histogram below by computing maximum likelihood multinomial over words in training matrix A. The strategy is to find the number of occurrence of unique words in A. Then the maximum likelihood multinomial probability over each unique word can be calculated by computing the division between the number of occurrence of each unique word and the total number of non-unique words in training matrix A. It appears that “bush” is the largest probability item and document A is probably associated with the American politics based on the 20 largest words.

|  |  |
| --- | --- |
| N = size(A,1); % Row size of matrix A  v = size(V,1); % Row size of matrix V  p = zeros(v,2); % Unique word ID and occurrence  for i = 1:v %In the loop of matrix p  for j = 1:N  if i == A(j,2) %If A(j,2) match unique word ID  p(i,1) = A(j,2); % Unique word ID  p(i,2) = p(i,2)+A(j,3);%Number of occurrence  end  end  end | su = sum(p(:,2)); %Total number of occurrence  b = p(:,2)./su; %Multinomial over unique words  [kk, s] = sort(b, 'descend'); %Sorting  kk = b(s,:); %s is the array index  barh(kk(1:20)) %Histogram of 20 Largest words  set(gca,'YTickLabel',V(s),'Ytick',1:20)  axis([0 0.015 0.005 20+0.005]) |

According to the model from part b, the test set probability is negative infinity. This is because given a training multinomial probability distribution in A, the probability of observing an unique word occurrence vector is zero if the test set B contains a word, which is not contained in the training set A, indicating it is impossible to observe an unique word outcome in the test set given a number of independent trials if it is not included in the training multinomial distribution in A.

d) Bayesian inference was implemented by using a symmetric Dirichlet prior with a concentration ratio of 0.1 on the word probability. However, it does not affect word probability too much and the 20 largest probability words remain unchanged in the histogram. The expression of the predictive distribution for a symmetric Dirichlet prior with a concentration is given by the following equation where ci is the number of occurrence of each unique word.

Symmetric Dirichlet distributions are used here when this Dirichlet prior is called for multinomial distribution since there is no prior knowledge favoring one unique word over another. indicates a sparse distribution which illustrates the probability of most of the unique words within data set will be close to 0 while the vast majority of the occurrence will be concentrated in a few unique words. It can be verified by the histogram shown in part b. The central commands are roughly the same apart from incremented value of concentration ratio for each word counts.

e) The log probability and per-word perplexity for the test document with ID 2001 are computed below.

|  |  |
| --- | --- |
| Log probability | Per-word perplexity |
| -3.6912e+03 | 4.3990e+03 |

The two results above were acquired by firstly extracting the unique words with document ID 2001 in B. Then the counts of each unique word, which appears in B, were computed in A by using the symmetric Dirichlet prior shown in part d. Finally, the log probability was calculated by summing up the multiplication between the counts of each unique word and log(). The combinatorial factor was not used when we tried to calculate the log probability for the document with ID 2001. This is because although we have different possible word combinations in a sentence or document, there are only few combinations, which actually make sense in terms of context. For instance, we might switch the names of people in the sentence. However, compared with the majority of possibilities, meaningful combinations still remain extremely little.

|  |  |
| --- | --- |
| dB = size(unique(B(:,1),'rows'),1); dA = size(unique(A(:,2),'rows'),1);  SumB = sum(B(:,3)); SumA = sum(A(:,3));  N = size(B,1); M = size(A,1); alpha = 0.1;  unique\_word\_A = zeros(max(A(:,2)),1);  for i = 1:M%-----Find unique word in A-------%  unique\_word\_A(A(i,2)) = unique\_word\_A(A(i,2))+A(i,3);  end  %-----Extract document with ID 2001 out------%  for i = 1:N  if B(i,1)==2001  doc\_2001(i,1)=B(i,1);  doc\_2001(i,2)=B(i,2);  doc\_2001(i,3)=B(i,3);  else  break  end  end | Sum\_doc = sum(doc\_2001(:,3));%Sum occurrence in doc\_2001  Size\_doc = size(doc\_2001(:,3),1);    %------Calculate Pi probability--------------%  sigma=alpha\*dA+SumA;  Pi=(alpha+unique\_word\_A)./sigma;    %------Compute the sum of log probability----%  Sum\_Pi=0;  for i = 1:Size\_doc  Sum\_Pi=Sum\_Pi+doc\_2001(i,3)\*log(Pi(doc\_2001(i,2)));  end    Log\_probability = Sum\_Pi  Perplexity\_2001=exp(-Log\_probability/Sum\_doc) |

|  |  |
| --- | --- |
| Log probability | Per-word perplexity |
| -1.5380e+06 | 2.5764e+03 |

However, when computing the log probability and perplexity, we need to use the multinomial with combinatorial factor because the order of documents will not affect context. Moreover, since we have 1430 unique documents in B, so the combinatorial factor is 12.

|  |  |
| --- | --- |
| %------find unique word in B  unique\_word\_B=zeros(max(B(:,2)),1);  for i=1:N  unique\_word\_B(B(i,2))=unique\_word\_B(B(i,2))+B(i,3);  end  % Sigma of counts times log().  Size\_B=size(unique\_word\_B,1);  Sum\_Pi\_B=0;  for i = 1:Size\_B  Sum\_Pi\_B=Sum\_Pi\_B+unique\_word\_B(i)\*log(Pi(i));  end | %------Combinatorial factor-------%  unique\_doc\_B=size(unique(B(:,1),'rows'),1);  factor=0;  for i=1:unique\_doc\_B  factor=factor+log(i);  end    Log\_probability = Sum\_Pi\_B+factor  Perplexity=exp(-Log\_probability/SumB) |

f) The log probability and perplexity of a uniform multinomial over all documents in B are shown below. It turns out that perplexity is equal to the number of unique words in B, which is much larger than the two values calculated in the previous part. This is because perplexity corresponds to the uncertainty associated with the number of unique word outcome in text modeling. Furthermore, the probabilities of in the uniform multinomial are the same so each unique word outcome is independent with others and also we do not need to consider combinatorial factor in this case due to the equal probability. However, in the previous part, the multinomial distribution was computed by considering counts of each unique word and total counts in B, which meant there were correlations between each probability so the uncertainty would decrease in terms of the perplexity over all documents in B. In addition, due to , it indicates a sparse distribution, which means the model is certain on the unique word outcome. As for the test document with ID 2001, since we only consider only a small amount of unique words in B so uncertainty is larger than perplexity over all documents in B while it is smaller than the maximum perplexity of 6870 in this part.

|  |  |
| --- | --- |
| Log probability | Per-word perplexity |
| -1.7300e+06 | 6.8700e+03 |

|  |  |
| --- | --- |
| uwB = size(unique(B(:,2),'rows'),1);  unique\_word\_B=zeros(max(B(:,2)),1);  SizeB=size(unique\_word\_B,1);  SumB=sum(B(:,3));  N=size(B,1);  %Compute unique words in B  for i = 1:N  unique\_word\_B(B(i,2))=unique\_word\_B(B(i,2))+B(i,3);  end | %The probability of is 1/uwB for uniform distribution  %Compute log probability and perplexity  for i = 1:SizeB  total=total+unique\_word\_B(i)\*log(1/uwB);  end    log\_probability = total  perplexity = exp(-log\_probability/SumB) |

g) The evolution of the mixing proportions versus the number of Gibbs sweeps up to 10 and 40 iterations is demonstrated in the graph below. The perplexity for the final state reached after 10 Gibbs sweeps is 2.1285e+03, which is lower than the ones acquired from part e and f. This is because all documents are modeled by the global word frequency distribution in e and f. This generative model does not specialize since it is possible that different documents might be about different topics so we have less information and a larger uncertainty on the assignments of unique words and documents into a topic distribution without consideration. However, the mixture of multinomial model assumes each unique word in a document is clustered with one specific topic distribution so we have more information about words and documents within a topic in an article.

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
| s=zeros(10,K);  total = 0;  %Store 20 mixture components for each iteration  for i = 1:K  s(iter,i) = sk\_docs(i,1);  total = total + s(iter,i);  end  %Calculate mixing proportion for each iteration  s(iter,:)=s(iter,:)./total; | figure  hold on  title('Mixing proportions as 10 Gibbs iterations 0', 'FontSize', 15,'FontWeight', 'bold')  xlabel('Iterations', 'FontSize', 13);  ylabel('Mixing proportions', 'FontSize', 13);  for i=1:20  plot(s(:,i),'Color',[1-i/20 i/20 1]) %Plot 20 mixing proportions using RGB colors  end  hold off |

h) MCMC can converge to drawing samples from the correct stationary distribution in the limit of a large number of samples. It demonstrates that if we are given a prior Dirichlet-multinomial distribution, then we can recursively generate other samples based on the multinomial conditional distribution assuming aperiodicity and irreducibility, which means we can eventually get from any state to any other state. Then after a long iteration, the samples are from the stationary distribution. After the model has converged, samples from the conditional distributions are then used to summarize the posterior distribution of parameters of interest. From this point on it stays in this distribution and moved about sub-space forever.

According to the evolution of mixing proportions from part g, the Gibbs sampler converges to the stationary distribution since all 20 mixing proportions move around their saturated values after 10 iterations. After restarting with different random seed, the Gibbs sampler does not seem to explore the posterior distribution since there is still some tendency for some individual mixing proportion evolution although the majority of curves converge to the stationary distribution and does not move around its equilibrium shown in the right graph of g. Therefore, although the model has converged, the Gibbs sampler still needs extra iterations to summarize and explore the posterior distribution since samples from this mixture of multinomial model are highly dependent.

i) The evolutions of the topic posteriors versus the number of Gibbs sweeps up to 10 and 30 iterations are demonstrated in the graph below. The perplexity for the final state reached after 10 Gibbs sweeps is 1.895e+03, which is lower than the ones acquired from part e, f and g. This is because latent Dirichlet allocation considers common topics in documents and unique words, assuming that any document and unique word can potentially contain more than one topic. In addition, for each document, it has its own distribution of topic with which gives a latent description of the document in terms of its topic item so we have more information on words and documents over their respective topic distribution. In the mixture of multinomial model, some documents span more than one topic but these blurred topics have not been learnt. Therefore, the perplexity is smaller using LDA compared with the model of mixture of multinomial and simple model in e and f.

|  |  |
| --- | --- |
|  |  |

Finally, 10 Gibbs iterations are not enough after restarting with different random seed since the topic posterior tends to stabilize when iterations reach around 30. More adequate iterations mean the Markov chain will reach equilibrium with higher probability.

|  |  |
| --- | --- |
| topic=zeros(10,K);  total = 0;  %Compute topics and sum of topics  for i = 1:K  topic(iter,i) = sum(swk(:,i));  total = total + topic(iter,i);  end  %Topic posteriors for each iteration  topic(iter,:)=topic(iter,:)./total; | figure  hold on  title('Topic posteriors as 10 Gibbs iterations 0', 'FontSize', 15,'FontWeight', 'bold')  xlabel('Iterations', 'FontSize', 13);  ylabel('Topic posteriors', 'FontSize', 13);  for i=1:20  plot(topic(:,i),'Color',[1-i/20 i/20 1])%Plot 20 different topic posteriors together  end  hold off |

j) The word entropy for each topic as 10 and 30 Gibbs iterations are illustrated below. It tends to decrease from beginning then start to saturate and reach the equilibrium of stationary distribution given enough iteration of 30. Entropy is used to evaluate the correlation among the words of each topic, which represents the intra-topic quality. Entropy decreases from beginning since we try to find common topics in document. Moreover, although we have various topics, the deviations of reduction are different. We can conclude some topics have higher quality with larger reduction while others have relatively low intra-topic quality. The topics with higher intra-topic quality tends to concentrate on a small group of unique words

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
| word\_topic=zeros(10,K); % Word entropy in all documents for each topics  swk\_2=zeros(6906,K); % Unique word entropy in each mixing proportions  for i = 1:K %Iterating through  swk\_2(:,i) = swk(:,i)./sum(swk(:,i)); %Probability of each unique word  for j = 1:6906 %Through unique word topics iterations  if swk\_2(j,i)~=0 %Compute word entropy for topics in each iteration  word\_topic(iter,i) = word\_topic(iter,i) - swk\_2(j,i)\*log(swk\_2(j,i));  end  end  end | figure  hold on  for i=1:20  plot(word\_topic(:,i),'Color',[1-i/20 i/20 1])  %Plot the word entropy for 20 iterations  end  hold off |