4F13 Machine Learning: CW3: Latent Dirichlet Allocation

Vera Johne

Definitions

Throughout this report, the following definitions hold:

- M: number of words in vocabulary
- N_d: the number of words in document d
- W_{nd} is the n^{th} word in document d, M is the number of words in the vocabulary

a

```
NA = sum(A(:,3,:));        %number of words
NB = sum(A(:,3,:));

N = NA + NB

W = max([A(:,2); B(:,2)]); %number of unique words
WA = size(unique(A(:,2,:)))
WB = size(unique(B(:,2,:)))

DA = max(A(:,1));          %nuber of documents
DB = max(B(:,1));
D = DA + DB
```

	A	B	A U B
Nr. words	271898	195816	467714
Nr. unq. words	60892	6870	6906
Nr. docs	2000	3430	5430

Table 1: Frequency observations in document collection A and B and the union of the two collections.

b

Here the simple document model outlined at the beginning of lecture 12 is used. This is not topic modelling, merely a frequentist model over the words. To find the maximum likelihood multinomial over all words in the vocabulary we simply find the frequentist statistics. Words that are not observed will have a zero probability. Note that M, the vocabulary size, is taken as the union of all the words in A and B. The equations still hold if M is larger than this.

Figure 1 shows the 20 words with the highest probability.

```
word_id = [1:W]';
for i = 1:W %W = 6906
    j = word_id(i) == A(:,2);
    s = sum(A(j,3));
    beta(i) = s/NA;
end
```

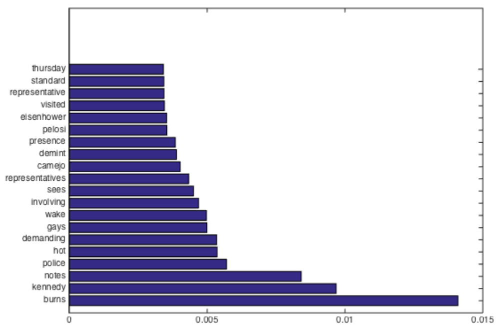

Figure 1: The 20 words with the highest probability over all the documents in A

c

The model should perform well on unseen documents. The simple model described in b is based on the gloabl word frequency of the words in training set A, and we trained the beta vector of word probabilities.

Now we want to take a new set of documents and maximize the log probability. The equations below show how we will fit beta from the training by maximising the likelihood.

$$\hat{\boldsymbol{\beta}} = argmax_{\boldsymbol{\beta}} \prod_{d=1}^{D} \prod_{n=1}^{N_d} Cat(w_{nd}|\boldsymbol{\beta}) \tag{1}$$

$$Cat(w_{nd}|\boldsymbol{\beta}) = \prod_{m=1}^{M} \beta_m^{I(w_{nd}=m)} \tag{2}$$

$$\hat{\boldsymbol{\beta}} = argmax_{\boldsymbol{\beta}} \prod_{d=1}^{D} \prod_{n=1}^{N_d} \prod_{m=1}^{M} \beta_m^{I(w_{nd}=m)} \tag{3}$$

$$= argmax_{\boldsymbol{\beta}} \prod_{m=1}^{M} \beta_m^{c_m}, \quad c_m = \sum_d \sum_n I(w_{nd} = m) \tag{4}$$

$$L(\hat{\boldsymbol{\beta}}) = argmax_{\boldsymbol{\beta}} \sum_{m=1}^{M} c_m log(\beta_m) \tag{5}$$

In equation 4 we see that the updated beta for word m is merely the beta value for word m from the training set, A, to the power of the number of times word m was observed in the set of documents in the testing set B. There is a beta value for each word m in the vocabulary. The problem is that if a word was not observed in the training, it will have a beta value of zero. Hence, if any of the documents in B contains a word that was not observed in A, this word from the training has a beta value of zero, and zero raised to the number of times this word was observed in B is zero.

A beta value of zero has serious implications on the log likelihood. Equation 5 gives the log likelihood equation. Log of zero is negative infinity, and hence we will have a negative log likelihood overall.

Furthermore, the fact that we cannot update the beta values for words observed in B, that were unobserved in A is also not ideal (their beta remains zero). This will result in probabilities of words for documents that are skewed from reality. We need a system that will not assign a zero probability to a word just because it is not in the training corpus.

One way to work around this implication is to add smoothing, whereby a count is added to every word and normalization is applied. This solves the issue of having words with zero probability. However, this 'hack' is not ideal, as it skews the data from what is actually observed in terms of counts.

Document collection	lp	Perplexity
Doc2001	-3692	4402
B	-inf	inf

Table 2: Log probability and perplextity of document collection B, and document 2001 in B using the frequentist model. Observe that because collection B has words not seen in A, the llh is negative infinity, whereas document 2001 only contains words observed in A, and therefore has a non infinity value.

d

$$p(\boldsymbol{w}|A,\alpha) = \int_{|\beta|=1} p(\boldsymbol{w}|\boldsymbol{\beta})p(\boldsymbol{\beta}|A,\alpha)d\boldsymbol{\beta} = \frac{1}{ZB(\boldsymbol{\alpha})}\int_{|\beta|=1}\prod_{m=1}^{M}\beta_m^{\gamma_m-1}d\boldsymbol{\beta} \tag{6}$$

$$p(\boldsymbol{\beta}|A,\alpha) = \frac{p(A|\boldsymbol{\beta})p(\boldsymbol{\beta}|\alpha)}{Z} \tag{7}$$

$$p(\boldsymbol{\beta}|\boldsymbol{\alpha}) = \frac{1}{B(\boldsymbol{\alpha})}\prod_{m=1}^{M}\beta_m^{\alpha_m-1} \quad p(A|\boldsymbol{\beta}) = \frac{1}{B(\alpha)}\prod_{m=1}^{M}\beta_m^{C_{Am}} \tag{8}$$

$$p(\boldsymbol{w}|\boldsymbol{\beta}) = \frac{1}{B(\boldsymbol{\alpha})}\prod_{m=1}^{M}\beta_m^{C_{Dm}} \tag{9}$$

$$\text{Let } \gamma_m = C_{Dm} + C_{Am} + \alpha \tag{10}$$

Note that we do not need to integrate over all possible values of beta, as the betas must sum to 1.

$$\text{Eq. 6} = \frac{1}{ZB(\boldsymbol{\alpha})}\int_{\beta}\prod_{m=1}^{M}\beta_m^{\gamma_m-1}\delta(1-\sum_{m}^{M}\beta_m)d\boldsymbol{\beta} \tag{11}$$

$$\delta(1-\sum_{m}^{M}\beta_m) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{-ik(1-\sum_m^M \beta_m)} \tag{12}$$

By rearranging the integrals, and doing a Laplace transformation, followed by inverse Laplace transform, it is clear that

$$p(\boldsymbol{w}|A,\alpha) = \frac{\prod_m^M \Gamma(\gamma_m)}{ZB(\boldsymbol{\alpha})\Gamma(\sum_m^M \gamma_m)} = \frac{B(\boldsymbol{\gamma})}{ZB(\boldsymbol{\alpha})} = \frac{B(\boldsymbol{\gamma})}{B(\boldsymbol{\eta})} \tag{13}$$

$$Z = p(A) = \int_{|\beta|=1} p(A|\boldsymbol{\beta})p(\boldsymbol{\beta}|\alpha)d\boldsymbol{\beta} = \frac{B(\boldsymbol{\eta})}{B(\boldsymbol{\alpha})} \text{ Where } \eta_m = C_{Am} + \alpha \tag{14}$$

The expression for the predictive posterior was written down and derived into a form of two beta functions. This is simple to calculate in MatLab (see question e). Observe that because we are using a prior, the problmem encountered in the frequentist model of zero log probabilities for document collections containing words that were unseen in the training corpus is avoided.

e

In d an expression for the probability of a collection of words $\boldsymbol{w}$, where w_m is the number of times word m was observed. Equation 15 expresses the log probability. The expression can be simplified further, but this expression works well for the purposes and in MatLab.

$$L(\boldsymbol{w}) = \Big(\sum_{m=1}^{M} log\Gamma(\gamma_m) - log\Gamma(\sum_{m-1}^{M}\gamma_m)\Big) - \Big(\sum_{m=1}^{M} log\Gamma(\eta_m) - log\Gamma(\sum_{m-1}^{M}\eta_m)\Big) \tag{15}$$

The code below computes the log likelihood of document2001 and collection B, as well as their respective perplexity values (results in Table 3).

```
doc2001 = B(1:232,:);
word_id = [1:W]';
CA = zeros(1,W);
CD = zeros(1,W);
CB = zeros(1,W);
for i = 1:W
       j = word_id(i) == A(:,2);
       s = sum(A(j,3));
       CA(i) = s;
       j = word_id(i) == doc2001(:,2);
       s = sum(doc2001(j,3));
       CD(i) = s;
       j = word_id(i) == B(:,2);
       s = sum(B(j,3));
       CB(i) = s;
end
%doc2001
gamma_1 = CA + CD +  0.1;
eta = CA + 0.1;
log_beta_gamma = sum(gammaln(gamma_1)) - gammaln(sum(gamma_1));
log_beta_eta = sum(gammaln(eta)) - gammaln(sum(eta));
llhD2001 = log_beta_gamma - log_beta_eta;
perplexityD2001 = exp(-llhD2001/440);
%B
gamma_B = CA + CB + 0.1
log_beta_gamma = sum(gammaln(gamma_B)) - gammaln(sum(gamma_B));
log_beta_eta = sum(gammaln(eta)) - gammaln(sum(eta));
llhB = log_beta_gamma - log_beta_eta;
perplexityB = exp(-llhB/NB);
```

Document collection	lp	Perplexity
Doc2001	-3601	4289
B	-1.5×10^6	2653

Table 3: Log likelihood and perplexity using bayesian inference on the simple model.

Multinomial Coefficient (e)

Consider a collection of words. This is a collection of counts of each word over the vocabulary. Regardless of the sequence of a collection of words, any collection with the same counts will have the same probability under the model. This means that we are not really considering different documents over the collection of counts. Hence the combinatorial factor becomes redundant, and is not included when calculating the log probability.

Perplexity (e)

The word perplexity tells us something about how confused the model is about a particular word. Given a perplexity value, p, for a particular word this means that the model could equally have chosen that word amongst p other words, uniformly. The model is equally as confidence in word w as it would have been with p other words. I.e. the lower the perplexity the more certain the model is about that particular word.

Collection B has a perplexity of infinity in the frequentist model. This really makes no sense, as M, the vocabulary size, is not infinite. When

using bayesian inference, we avoid this problem of the zero betas, as we are integrating over the betas. This results in a perplexity of 2653. This means that the model could equally have generated 2652 other words instead of word m.
The frequentist model did give us a non infinite log likelihood value for document 2001, because its words are all observed in A, as described in section c. It is observed that the perprexity of this document is only slightly lower when using the bayesian inference (113 less confusion). This is because in the frequentist model the model has fixed betas. The model gives high certainty to its beta values- the probability of a word being observed. On the other hand, in the bayesian inference case we consider all possible values of betas, contrained on the fact that they sum to 1. I.e. the model is not fixed on one particular beta value.

f

The uniform multinomial has probabilities of each word being equivalent, i.e. $\forall_m \beta_m = \frac{1}{M}$, and the likelihood of a document collection becomes:

$$p(\boldsymbol{w}|\boldsymbol{\beta}) = \prod_d^D \prod_m^M \beta_m^{C_m} = \prod_d^D \prod_m^M \frac{1}{M^{C_m}} = \prod_d^D \frac{1}{M^{N_d}} = \frac{1}{M^{N_D}} \qquad (16)$$

$$L(\boldsymbol{w}) = log(\frac{1}{M^{N_D}}) = -log(M^{N_D}) = -N_D log(M) \qquad (17)$$

```
llh_D = −440*log(W);
llh_B = −NB*log(W);
perD = exp(−llh_D/440);
perB = exp(−llh_B/NB);
```

Observe that in this model, a longer sentence is less likely to be observed given the vocabulary. Therefore we already know that document 2001 will have a higher likelihood, then the collection of documents B.
The log likelihood is negative infinity for both document D and collection B. This makes sense because M is large, and the number of words in both these word collections is large, and hence MatLab gives the infinity value.

Document collection	lp	Perplexity
Doc2001	-inf	inf
B	-inf	inf

Table 4: Log likelihood and perplexity of the uniform multinomial model.

The log likelihood value results in a perplexity of infitity for both B and document 2001. We observed a perpexity of infinity similar to the other frequentist model for document collection B in question c. This uniform model is not capturing any information from the training set (no training is necessary). There is no prior information present. From equation 16, it is clear that any document will have a low probability (a document with 1 word has a probaility of 0.0001). The longer the documents the lower the chance, and hence all 'long' document collections will have a log likelihood of negative infinity, and perplexity of infinity. The frequentist model in c, gave a llh of negative infinity and perplexity of infinity for collection B. This is because the beta values were only capturing information about the training corpus. This did not happen with document 2001 because it contained no words outside of the training corpus, hence the overfitting. The Bayesian inference model does not get the infinite perplexity values namely because of the prior, it is capturing information from the trining corpus, but also including the symmetric Dirichlet prior, to diminish overfitting.

g

In the BMM each document is generated by one topic, and each word in the document is generated from that topic's word dustribution. The topic of the document is the laten variable, z_d. The goal is to find the posterior probability (the mixture propoportions$\boldsymbol{\theta}$) over the topics given the training corpus and a fixed number of topics (using collapsed Gibbs).

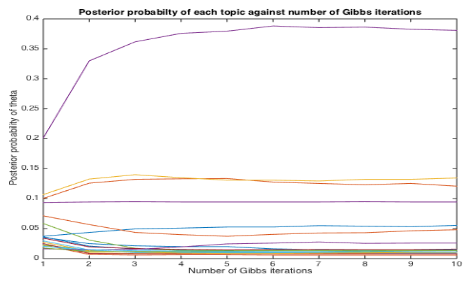

Figure 2: Each line represents one of the twenty topics. Observe that as the number of Gibbs sweep increase, the posterior probability of each topic appears to stabalise.

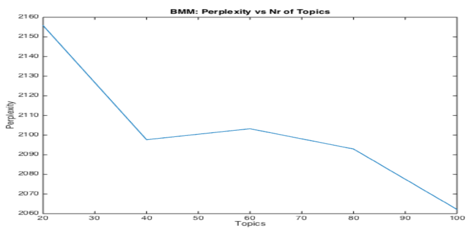

Figure 3: Observe that as number of topics increase the perplexity decreses. The plot is not monotonically decreasing as the perplexity increases slightly from 40 topics to 60 topics.

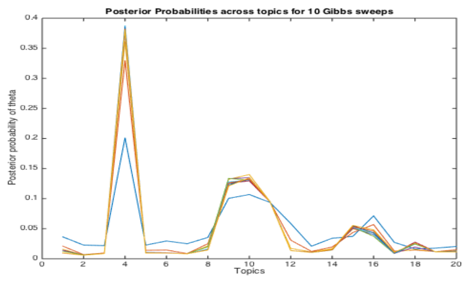

Figure 4: Each line represents one of 10 iterations. The graph shows that the posterior probability increases with iterations. A topic with higher posterior probability (e.g. 4), means that it is more likely that an unseen document was generated by this topic. Note that it is unknown that exactly topic 4 is, so we look for a peak, meaning that one topic is more likeli to have generated the document over the other topics.

```
theta(:,iter) =
( sk_docs(:) + alpha ) / (sum( sk_docs(:) + alpha)) ;
```

Figure 2 and 4 are plots of theta and theta transformed calculated in the MatLab snippet. Figure 3 was generated by saving the perplexity values at intervals mod 20 from 20 to 100.

Document collection	lp	Perplexity
B	-1.5×10^6	2120

Table 4: Perplexity and l using BMM

Observe that the perplexity has decreased from all the previous categorical models. In question e, the perplexity was 2653, now the perplexity is 2120 for collection B. We can explain this through the strucutre incorporated in the model. The previous models considered a collection of documents as a whole, it did not model individual documents. The BMM model encapsulates document structure. In a way, the model is narrowing down its 'search window' of words because of the increase in strucutre, and hence the lower perplexity. Narrowing down the search window happens naturally with the added structure where documents are generated by a topic with its own distribution over words. The structure the model incorporates is nicely demonstrated in the graphical models of the models (see lecture notes, due to limited space).

h

Convergence of MCMC means the distribution converges to a stationary distribution (the target distribution, the underlying markov chain). When the stationary distrubtion is reach the mean and variance values are constant in further samples.
It is a known difficult practical problem to know when exactly an MCMC has converged

Convergence of MCMC means the distribution converges to a stationary distribution with a constant mean and variance values. Using Gibbs sampling for BMM, it does converge since based on Figure 2, we can see that some topics have a higher posterior probability across documents compared to others. If we had used fewer than 10 Gibbs sweeps, almost all topics are assigned some relatively higher probabilities initially. However, as the number of Gibbs sweeps are increased, the topic posteriors spead out, showing that some topics are assigned higher posteriors (which is close to the true posterior distribution) compared to other topics. The Gibbs sampler, once converged, samples more frequently around the modes of the posterior distribution.
In bmm.m each documents mixture components are randomly initialised. This is affected by the seed. Therefore, the random seed only affects the proportions assigned to each topic, but these should eventually converge to the same proportions.

i

In the BMM each document was allocated one topic from the global distibution of topics. In LDA, each document containts a distribution over topics, and words are generated from a topic drawn from the document specific distriubution over topics.
LDA proposes an inference problem (an intractable posterior), becuase of the vast number of possible latent variables: z_{nd}, the topic for word n in document d. This key problem in LDA is solved by collapsed Gibbs sampling which integrates out the betas (each topics probability distribution over words) and the thetas (each documents topic proportions), giving us the predictive distribution for z_{nd}:

$$p(z_{nd} = k | z_{-nd}, w, \gamma, \alpha) \propto \\ p(z_{nd} = k | z_{-nd}, \alpha) p(w_{nd} | z_{nd} = k, w_{-nd}, z_{-nd}, \gamma) \tag{18}$$

The figures below shows the predictive posterior over topics after different numbers of Gibbs sweeps. It is observed that LDA converges slower than BMM (compare Figure 2 and 5), and that 10 Gibbs sweeps is not enough (compare figure 6 and 7). This makes sense because the posterior in LDA is much more complex compared with BMM, therefore we need more samples in order to converge to the distribution.

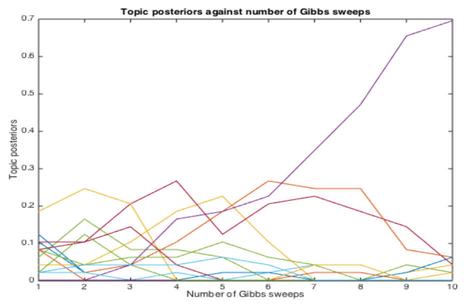

Figure 5: Plot for topic posteriors in document 2. Each color represents a different topic. We observe that the model is still uncertain about the topic posterios for document 2 after 10 iterations (no apparent convergence).

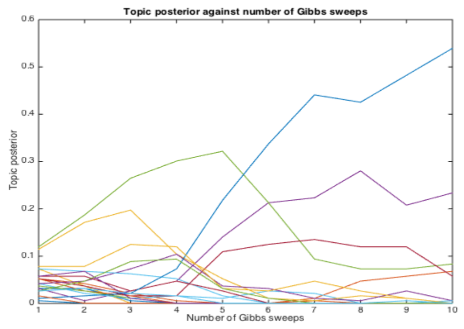

Figure 6: Same as Graph 5, but for Document 200. Observe that the topic posteriors are different, as expected.

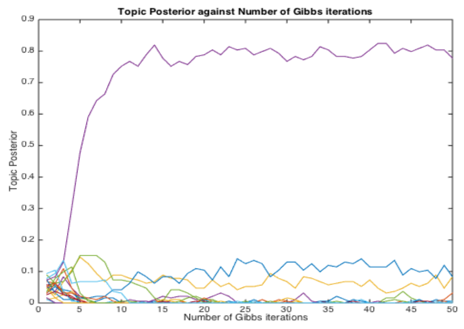

Figure 7: Document 200. Observe that the topic posterior is different over this document, this is as expected. (50 iterations)

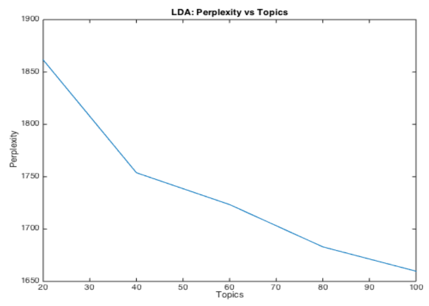

Figure 8: The perplexity is monotonically decreasing as the number of topics increases.

Sweeps	Perplexity
10	1866
30	1682
50	1652

Table 5: Perplexity after different nr. of sweeps

The argument for why LDA has a lower perplexity compared with BMM, is similar to the discussion given in g. LDA incorporates more structure, and hence is more certain on its words. On the other hand, LDA also handles ambiguity, which is something the other models fail to do. Words can belong to multiple topics, and have multiple meanings. Language is ambiguous. In LDA several different topics, can generate a single word, whereas in BMM all words were generated by one topic's distribution over words. To find the topic of a word in a document LDA uses the other words in the document to look at the topic distribution for the document.

This results in a better informative model, a better generative process of creating the document. Therefore, for test set documents, LDA generalizes well and obtains higher likelihood values for unseen documents, leading to lower perplexity measures. On the other hand, the BMM has a generative process that partitions the documents to produce disjoint subsets of clusters. Since in BMM, we assign a single topic to a document, with a measure of uncertainty, it is a less informative generative process due to which BMM generalizes poorly compared to LDA. For unseen documents, BMM obtains a lower likelihood value compared to LDA, and hence perplexity measures are higher. This shows that LDA is a better model compared to BMM.

j

Word entropy

$$H(w) = -\sum_k p(k|w)log(p(k|w)) \tag{19}$$

Word entropy is a measure of uncertainty that expresses the disorder of words within each topic where for each topic, we can compute the word entropy. The word entropy decreases as we have more Gibbs iterations (LDA converging). With higher number of iterations, our informative measures about the topics generating the document increases. Similarly, with Gibbs iterations, the words become more informative about the topics it represents (less disorder). The decrease in word entropy as shown in the figure further means that our uncertainty measure about which words are responsible for representing the particular topic decreases (as shown by decrease in word entropy). The word entropy measure is the expected value of the information contained within each topic. With convergence, the words representing the topic becomes more aligned, informative and certain about the topic it represents. The figure shows the word entropy across number of Gibbs iterations. Similar to previous figuure (CITE LDA FIGURE), it also shows that two topics contain more informative measures of words that it contains - suggesting that for those two topics (CITE TWO TOPICS WHICH HAVE LOWEST PEAK - FIGURE MUST ALIGN WITH LDA FIGURE), the words representing that topic are more informative and certain compared to the words in other topics.

```
theta(:,itr1)=(skd(:,200)+alpha)/(sum(skd(:,200)+alpha));
word_entropy = zeros(K,itr2);
beta = zeros(W,K);
for o = 1:K
    %probability of words for each topic
    beta(:,o) = (swk(:,o)+gamma)/(sum(swk(:,o)+gamma));
end

for i = 1:K
    entropy = 0;    %word entropy for topic
    for j = 1:size(beta,1)
        entropy = entropy + (-beta(j,i)*log(beta(j,i)));
    end
    word_entropy(i,itr1) = entropy;
  end
```

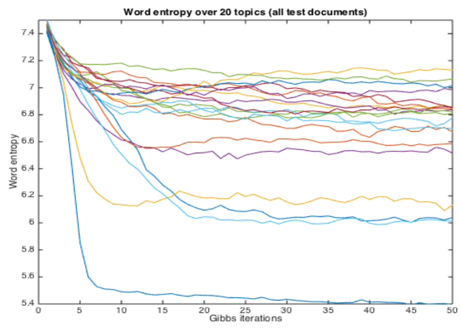

Figure 9: INSERT CAPTION

k

Since LDA is a generative process, the more the number of topics we have, the better is the generative process and we are more informative about the document structure. Therefore, for unseen test documents, having the model trained with higher number of topics leads to lower perplexity values due to better model generalization capabilities. Compared to that, since in BMM we find how likely it is for the document to be generated by some topics, for unseen documents having the model trained with higher number of topics is not guaranteed to monotonically decrease the perplexity. This is because, it maybe that the trained document set had a higher posterior probability for some topics, but those topics may not be relevant (or more likely) for an unseen test document. Having more topics in BMM model, still does mean we obtain a better generative model for unseen documents compared to few topics (where we are less informative about the document structure). The more topics we have for BMM, the better we can do unsupervised clustering of documents, leading to higher likelihood values for test documents. However, since the BMM model is better for generalization compared to the categorically model, as shown in the figures, we do get a decreasing perplexity measures. The decrease in perplexity is much steeper for LDA compared to BMM since LDA is a better model even with higher number of topics.