Mixed Membership Modeling via Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) is a probabilistic model for document analysis. It is an example of a class of methods called mixed membership modeling.

EM with mixed Gaussians captured soft cluster assignments (modeling uncertainty in cluster assignment), but the underlying model still assumes the observation belongs to a single cluster. Taking our document example, what if an article is actually about two topics? Rather than modeling uncertainty in cluster assignment, we would prefer to build a model that indicates the article belongs in two topics and learn the prevalence of those topics. Mixed Membership models allow modeling of this type.

An alternative document clustering model

The EM with mixed Gaussians model that was discussed in the prior module involved documents represented as tf-idf vectors (or we could use word count vectors) with one entry per word. We then used some set of Gaussians to discover clusters in the tf-idf space and model soft cluster assignments with uncertainty.

The EM (Expectation Maximization) algorithm can be used with probability distributions other than Gaussians.

For the task of finding topics in a corpus of documents, we can model probability distributions as topic specific vocabulary distributions as topic probability vectors. Each topic is associated with a topic probability vector over the words in the vocabulary; a vector of probabilities, one per word, with the probability of the word being observed in that topic. (I assume that these probabilities add up to one). This probability vector is called a multinomial probability distribution.

Topic 1	
experiment	0.18
observation	0.14
discover	0.02
hypothesize	0.01
…	…

Example of a Topic Probability Vector

We use a bag of words model for documents where we take the words of a document ignoring ordering in the document and allowing for duplicates in the set (this is an unordered multiset of words – it can contain multiple instances of a word).

So given the bag of words model and the topic probability vectors, we can derive an EM algorithm that maximizes the likelihood of the bag of words given the mixed multinomial distribution (the topic probability vectors). Rather than computing the likelihood of a tf-idf vector under each Gaussian in a mixed Gaussian model, we calculate the likelihood of the bag of words under each topic probability vector. This is resulting model can be referred to as a mixture of multinomials model.

The set of prior likelihoods for each cluster, π_k, is the same in both the Mixed Gaussians and the Topic Vector models – it is a vector of likelihoods, one per cluster, where each element is the likehood that the document is in that cluster.

The important point here is that the entire document is assigned, not just part of it. The document's assignment is modeled with uncertainty, but it is assigned in whole. In In LDA, we will find that each word in the document can be assigned to a cluster.

Components of latent Dirichlet allocation model

In LDA, we use the bag of words model for documents, which ignores word ordering in the document but retains information on how many times each word appears. Additionally, we use topic

specific probability distributions, as in the model just discussed. In this model, rather than assign the entire document to a topic, we assign each word in the document to a topic.

LDA inputs:

- unordered multiset of words per document in the corpus

LDA outputs:

- Corpus-wide document vocabulary distributions
- Topic assignments per word per document
- Topic proportions per document

For the corpus, the algorithm learns the corpus' vocabulary distributions (word probabilities) per topic:

Topic 1	
word 1	0.18
word 2	0.14
word 3	0.02
word 4	0.01
…	…

Topic 2	
word 1	0.01
word 2	0.09
word 3	0.03
word 4	0.04
…	…

Topic 3	
word 1	0.11
word 2	0.14
word 3	0.01
word 4	0.01
…	…

Corpus-wide vocabulary distribution per topic

For each document, the algorithm learns document specific topic assignments for each individual word in the document.

document 1	
Word	Topic
about	1
experiment	2
journey	1
…	…
experiment	3
descent	4

document 2	
Word	Topic
adjunct	2
dealer	1
collapse	4
…	…
stalled	4
more	4

…

document N	
Word	Topic
random	2
travel	1
measure	2
…	…
random	2
moment	1

Topic assignments per word per document

For each document, the algorithm learns a topic proportion vector (also called document topic prevalence) which tells us what portion of the document is in each topic. For each document, topic membership is expressed as a vector of weights that sum to one; the magnitude of each weight indicates the degree to which the document represents that particular topic:

$$\pi_i = [\pi_{i1}, \pi_{i2}, \ldots, \pi_{ik}]$$

where

π_i The topic proportion vector for the i[th] document

π_{ik} The proportion of the i[th] document in the k[th] cluster

document	Topic 1	Topic 2	…	Topic K
1	0.09	0.51	…	0.23
2	0.32	0.02	…	0.44
3	0.67	0.03	…	0.01
4	0.02	0.03	…	0.11
…	…	…	…	…
N	0.17	0.21	…	0.13

Document Topic Prevalence

Goal of LDA inference

Once the algorithm has run, we can interpret the outputs in qualitative ways.

We can order each topic vocabulary distribution by descending word probability so that the most probable words are listed first. If the model is working well, then we can identify some coherence in the most probable words and assign some descriptive name to the topic. LDA tends to provide better coherence for this kind of labeling that does EM because it models assignment to multiple documents rather than uncertain assignment to a topic.

The document specific topic proportions can be used to:

- Relate documents to each other. For instance, in a retrieval task, find documents in a specific topic.

- Relate user preferences (the documents that a user chooses to read) to topics.
- File documents in multiple categories when offering documents to a user.

Note that the topic assignment vectors (the probability of a word in a topic) are not generally interesting post-facto, although they are critical in calculating the other artifacts of the model that are interesting.

Bayesian Inference via Gibbs sampling

LDA is typically specified as a Bayesian model which accounts for uncertainty in the parameters by effectively averaging over this uncertainty when making predictions. This approach naturally regularizes the parameter estimates, resulting in what is called a maximum a-posteriori estimate. However, in the context of EM, the algorithm using this Bayesian specification becomes intractable because the E step has an intractable expectation. In practice an approximation is introduced to handle the very complicated expectation. This is often called Variational EM.

For our purposes, we will avoid Variational EM and use another approach to Bayesian inference called Gibbs sampling.

Gibbs sampling from 10,000 feet

Gibbs sampling can be used for Bayesian inference. It iteratively calculates hard assignments, but the assignments are drawn randomly from a distribution. Gibbs sampling models the uncertainty in the parameters by taking random samples of the model parameters and assignment variables.

Whereas Variation EM is a complicated algorithm, with Gibbs sampling:

- Update steps are intuitive
- Straightforward to implement

In EM, we calculate the likelihood of the observations given the model. However, we know that there is some uncertainty in the model itself. This Bayesian approach includes the uncertainty of the model as part of a joint probability. The joint model probability is the probability of the observations given the model parameters AND the probability of the model parameters themselves.

We can plot the joint probability over the algorithm iterations. What we will see is that the probability grows over time, but NOT monotonically. It is NOT an optimization algorithm, as is k-means and EM; it is an randomized algorithm. As the Gibbs sampling proceeds, it is sampling within the probability distribution; it is effectively exploring the solution space. During the process, the joint probability may go up and down, but will generally increase as it 'burns in' and moves towards a 'correct' Bayesian estimate. It takes a relatively high number of iterations, but the algorithm eventually gets to a bumpy plateau where it goes up and down randomly around a mean maximum joint probability. We can take a range of samples in this plateau area to calculate statistics about the joint probability. The maximum probability in this region will be the a-posteriori estimate; taking the model parameters at this iteration provides us with our 'good' solution.

A standard Gibbs sampler for LDA

Gibbs sampling, in its most standard implementation, just cycles through all of the assignment variables and model parameters and randomly samples each one from a conditional distribution conditioned on the previously sampled values of all the other model parameters and assignment variables, which change in each iteration, and on the observations which do not change.

Where EM and K-means update the assignment variables and the model parameters in two separate steps, the Gibbs sampler algorithm treats them the same.

Remember that we instantiate of each of these on every iteration of the algorithm:

- assignment variables

- model parameters
    - the corpus-wide topic vocabulary distributions
    - the document specific assignment variables of words to topics
    - the document specific topic proportion factors (topic prevalences)

In the next iteration, we wish to reassign all the assignment variables of words in a document to a topic.

Step 1: Randomly reassign all word indicator variables (assignment variables), Z_{iw}, based on conditional distribution where we're fixing the values of:

- document specific topic proportion factors
- corpus-wide topic vocabulary distributions

Step 2: Randomly reassign document topic prevalences based on assignments Z_{iw} in the current document.

Step 3: Repeat for all documents

Step 4: Randomly reassign corpus-wide topic vocabulary distributions based on assignments Z_{iw} in the entire corpus.

Step 5: Repeat steps 1..4 until computational budget is reached.

Step 1:
We randomly select a word in document i, we call it w^{th} word in document i. Then the probability of assigning word w in document i to the a given topic (let's use topic 2 here), $Z_{iw} = 2$ is R_{iw2} which is the prior probability of choosing a random word and having it be from topic 2, and that is the prevalence of topic 2 in document i.

$$R_{iw2} = \frac{\pi_{i2} \times p(word|Z_{iw} = 2)}{\sum_{j=1}^{K} \pi_{ij} \times p(word|Z_{iw} = j)}$$

Where

π_{i2}	the prior probability (prevalence) of topic 2 in document i from the prior iteration
Z_{iw}	topic assignment of word w in document i
$p(word\|Z_{iw} = 2)$	the likelihood of word w of document i being in topic 2. This is taken by looking up the word in the corpus-wide topic vocabulary distribution for topic 2 and using that probability.
$\sum_{j=1}^{K} \pi_{ij} \times p(word\|Z_{iw} = j)$	the sum of all possible topic assignments; the total likelihood of seeing the word in any topic. This is used to normalize to the result is a probability between zero and 1.
K	Total number of topics

This is very much like the responsibilities calculated in EM, except here when we're looking at the prior probability on a given assignment we're looking specifically within this document using document specific topic prevalences and then we're scoring our word under a given topic probability vector instead of scoring a whole tf-idf vector under a given Gaussian distribution.

This example is for topic 2, but in this same way, we would compute the responsibility for every possible topic for this specific word; topic 1..K, then normalize so that the whole vector sums to one. This then represents what is called a probability mass function; a distribution over a set of integers, one to K. We can then draw a value randomly from this distribution in order to reassign the word to a topic.

We repeat this procedure for every word on this document; reassigning each word in the document by pulling it from a distribution that we calculation from priors. This assigns each word in the document to a random topic based on the distribution.

Note that resampling the word indicator variables is a data parallel problem. We could reassign each word indicator variable completely in parallel. All that is needed is to condition on the topic vocabulary distributions and the document specific topic proportions and then everything decoupled. There is no dependence on the assignments to other words in the document or in the corpus.

Step 2:
Once this is done, we can use this to reassign the document topic proportions for this document given the set of word assignments that were just made. This is done by counting how many times a given topic is used in this document. But these counts are going to be regularized by a Bayesian prior. We can think of the Bayesian prior as introducing a set of what are called pseudo-observations. In this case, it's as every topic has a fixed number of observations in that topic that bias the distribution from just using the observed counts in this document. So we use these counts, both the observed counts in this document given this sample set of word assignment variables as well as the parameters of our Bayesian prior, to form a distribution over these topic proportions, and then we sample these topic proportions from that distribution (the particular form of that distribution is not discussed here).

Step 3:
We repeat sampling the word assignment variables and the topic proportions for each document in our entire data set.

Step 4:
Next we reassign the corpus wide topic vocabulary distributions. To figure out how probable our words within a given topic, we can look at our word assignment variables in the entire corpus and determine of how many times a word was assigned to topic one, and we can use that information to inform us of how probable the word is under topic one. We do this for every word in our vocabulary for each one of these different topics. As with the word assignments, these counts of topic usage within the corpus are realized by Bayesian priors placed over the topic vocabulary distributions.

The Gibbs sampling algorithm repeats these steps again and again; resampling our word assignment variables, our document specific topic proportions, and our corpus wide topic vocabulary distributions until we run out of computational budget (typically maximum number of iterations).

Collapsed Gibbs sampling for LDA

Based on the special structure of LDA, the standard Gibbs sampler can be simplified (collapsed) such that only the indicator variables, Z_{iw}, need be sampled. So we do not need to sample the corpus-wide vocabulary distributions and the per-document topic prevalences. In practice, this leads to much better performance because we are examining uncertainty in a much smaller space.

We are randomly reassigning Z_{iw} based on the current assignments Z_{jv} of all other words in the corpus. This means that we are resampling our word assignment variables for every word in the document and then every word in the corpus which means we must sequentially resample the words given all the other words in the corpus; this is not a data parallel task. In the standard Gibbs sample, we had the model parameters as fixed values which could be used to determine the word assignment variables. But in this collapsed model, we don't have that those as fixed model parameters. All we have is the other words and the other assignments that were made to those words. We never have to sample our topic vocabulary distributions or the document specific

topic proportions; we just sample the word indicator variables (the topic assignment for each word in each document). However, we must do so sequentially, losing the ability to parallelize across that operation.

A worked example for LDA: Initial setup

Select a document

document:	document i				
topic:					
word:	Epilepsy	dynamic	Bayesian	EEG	model

5-word document

Initialize the topics

document:	document i				
topic:	3	2	1	3	1
word:	Epilepsy	dynamic	Bayesian	EEG	model

document topic assignment by word (word indicators)

In this case, we have randomly assigned the initial topics, but there are other approaches. Initialization does matter in practice. Initialization may not matter if the algorithm can be run long enough, but that is not always possible.

Repeat this for all documents in the corpus

document i						
3	document j					
Epilepsy	1	document x				
	model	1	document y			
		test	2	3	2	1	1
			cross	test	likelihood	validate	data

Maintain local statistics

For a given document, we use the document's topic assignment per word to count the number of words in each topic for the document.

	Topic 1	Topic 2	Topic 3
document i	2	1	1

document topic counts

Maintain global statistics

For each word in the corpus (in the entire vocabulary), we count the number of times it appears in each topic.

	Topic 1	Topic 2	Topic 3
epilepsy	1	0	35
Bayesian	50	0	1
model	42	1	0
EEG	0	0	20
dynamic	10	8	1
…			

Corpus wide topic count by word

Gibbs Sampling

Randomly reassign topics, word by word.

Randomly choose a document and a word in the document, and remove it's topic assign, decrementing the document topic count for that topic and the corpus topic count for that word.

document:	document i				
topic:	3	~~2~~	1	3	1
word:	Epilepsy	dynamic	Bayesian	EEG	model

document topic assignment by word during resampling

Now the document topic count for Topic 2 must be decremented:

	Topic 1	Topic 2	Topic 3
document i	2	0 ~~1~~	1

document topic counts during resampling

Now the corpus wide topic count for Topic 2 must be decremented:

	Topic 1	Topic 2	Topic 3
epilepsy	1	0	35
Bayesian	50	0	1
model	42	1	0
EEG	0	0	20
dynamic	10	7 ~~8~~	1
…			

Corpus wide topic count by word during resampling

Reassign the word to a randomly selected topic
At this point, the word we chose from the document has no topic assignment, Z_{iw} = ? so we want to reassign it.

document:	document i				
topic:	3	?	1	3	1
word:	Epilepsy	dynamic	Bayesian	EEG	model

document topic assignment by word before reassignment

We randomly reassign Z_{iw} with a probability distribution of the form p(Z_{iw} | every other word's assignment in the corpus, Z_{jv}, and the words in the corpus)

Deriving the resampling distribution

The resampling distribution involves two probability terms. **The first term takes into account the document's affinity for the topic** by

taking the ratio of the number of words in the document assigned to the topic over the total number of words in the vocabulary. We are using our decremented counts in this calculation.

$$\frac{n_{ik} + \alpha}{N_i - 1 + K\alpha}$$

where

n_{ik}	number of assignments in document i to topic k (using the current decremented counts)
N_i	number of words in document i
K	number of topics
α	Bayesian prior smoothing parameter

Notes:

- We do not need to regularize like we did with EM, because our terms cannot go to zero due to the Bayesian prior parameters.
- We use N_i *-1* because we have removed one word from the total number of words in the document during resampling. This will get added back in the next step.

The second term takes into account the topic's affinity for the word by using the ratio of the number of times this word was assigned to topic k in all documents and the total assignments of all words in the corpus to topic k:

$$\frac{m_{word,k} + \gamma}{\sum_{w \in V} m_{w,k} + V\gamma}$$

where

$m_{word,k}$	number of assignments corpus-wide of the word to topic k
$\sum_{w \in V} m_{w,k}$	The total corpus-side number of assignments of all words to topic k
V	The number of words in the vocabulary
γ	the Bayesian prior smoothing parameter for this term

The product of these two probabilities are used when drawing a new topic assignment for the word.

$$\frac{n_{ik} + \alpha}{N_i - 1 + K\alpha} \times \frac{m_{word,k} + \gamma}{\sum_{w \in V} m_{w,k} + V\gamma}$$

α (alpha) and γ (gamma) can be thought of as smoothing parameters when we compute how much each document "likes" a topic (in the case of alpha) or how much each topic "likes" a word (in the case of gamma). In both cases, these parameters serve to reduce the differences across topics or words in terms of these calculated preferences; alpha makes the document preferences "smoother" over topics, and gamma makes the topic preferences "smoother" over words.

So we use this probability when drawing a new topic assignment for the word. In our example with document i, the word is 'dynamic'. Let's say we draw topic 1. Now we can fill in the topic in the document's word assignments;

document:	document i				
topic:	2	**1**	1	3	1
word:	Epilepsy	dynamic	Bayesian	EEG	model

document topic assignment by word after reassignment

and we can increment the number of words in topic 1 in the documents topic counts;

	Topic 1	Topic 2	Topic 3
document i	3 ~~2~~	0	1

document topic counts after resampling

and finally we can increment the number of corpus-wide assignments for the word to topic 1;

	Topic 1	Topic 2	Topic 3
epilepsy	1	0	35
Bayesian	50	0	1
model	42	1	0
EEG	0	0	20
dynamic	11 ~~10~~	7	1
…			

Corpus wide topic count by word after resampling

Repeat for all words in the document and for all documents

So, we've resampled the second word in document i. Now we would do the same thing for the rest of the words in document i. Then we would do it for all words in document i + 1 until we do all documents. Then we would do it all again until we have consumed our computational budget.

Using the output of collapsed Gibbs sampling

Remember that in the standard Gibbs sampler, we learn two other useful pieces of information; the corpus-wide vocabulary distributions per topic and the topic prevalence for each document.

Topic 1	
word 1	0.18
word 2	0.14
word 3	0.02
word 4	0.01
…	…

Topic 2	
word 1	0.01
word 2	0.09
word 3	0.03
word 4	0.04
…	…

Topic 3	
word 1	0.11
word 2	0.14
word 3	0.01
word 4	0.01
…	…

Corpus-wide vocabulary distribution per topic

document	Topic 1	Topic 2	…	Topic K
1	0.09	0.51	…	0.23
2	0.32	0.02	…	0.44
3	0.67	0.03	…	0.01
4	0.02	0.03	…	0.11
…	…	…	…	…
N	0.17	0.21	…	0.13

Document Topic Prevalence

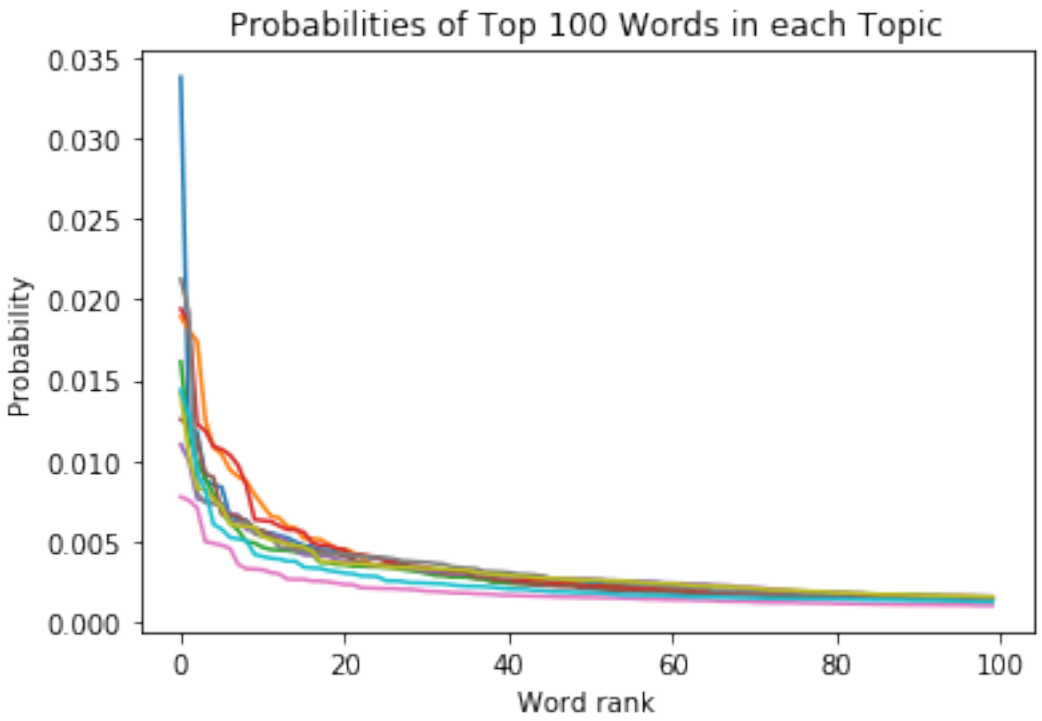
Using the topic assignments per word per document (the word assignments variables) that we have in the Collapsed Gibbs sampler, we can create these distributions after the collapsed Gibbs sampler is done. Post-facto, we choose a sample (our 'best' sample) and use it to create the conditional distributions based on the word indicators.

Now that we have the 3 pieces of the standard Gibbs sampler, we can add new documents to our collection by running the standard (uncollapsed) Gibbs sampler on the documents using the fixed topics and fixed topic vocabulary distributions. This can be done on new documents in parallel.

To get a document-specific topic proportion vector post-facto, we draw this vector from the conditional distribution given the sampled word assignments in the document. Notice that, since these are draws from a distribution over topics that the model has learned, we will get slightly different predictions each time. To get a more robust estimate of the topics for each document, we can average a large number of predictions for the same document.

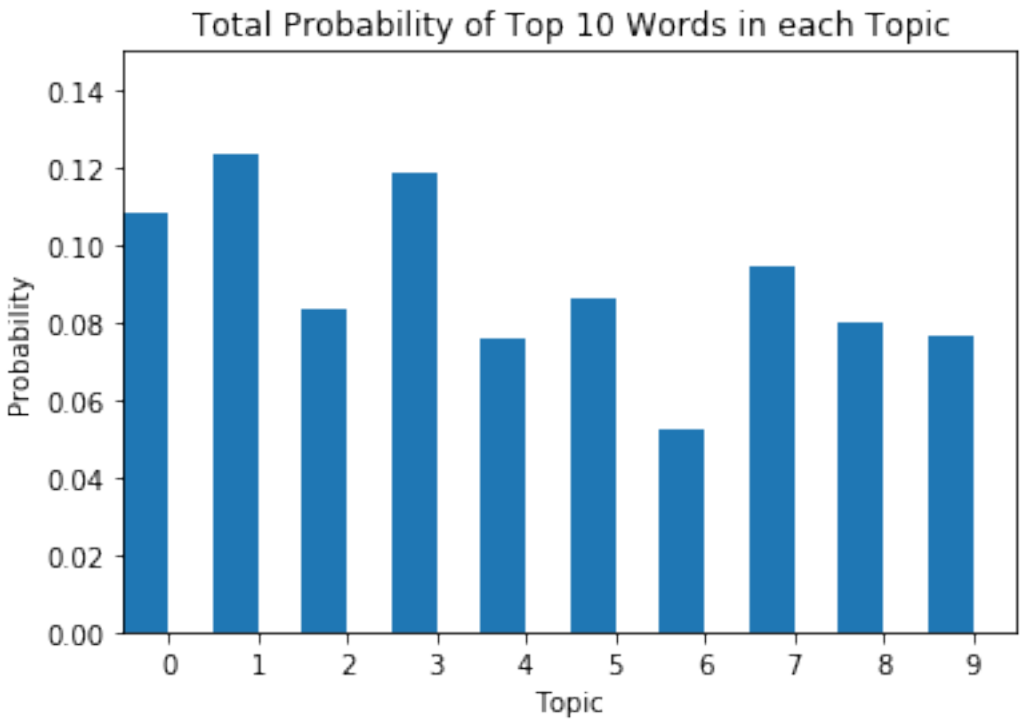
Measuring the importance of top words

Below is a plot for the top 100 words by weight in each of 10 topics clustered with a collapsed Gibbs sampler. Each line corresponds to one of the ten topics.



Notice how for each topic, the weights drop off sharply as we move down the ranked list of most important words. This shows that the top 10-20 words in each topic are assigned a much greater weight than the remaining words. However, it is important to realize that the vocabulary has 547462 words in total, so the total weight of the top words is still low.

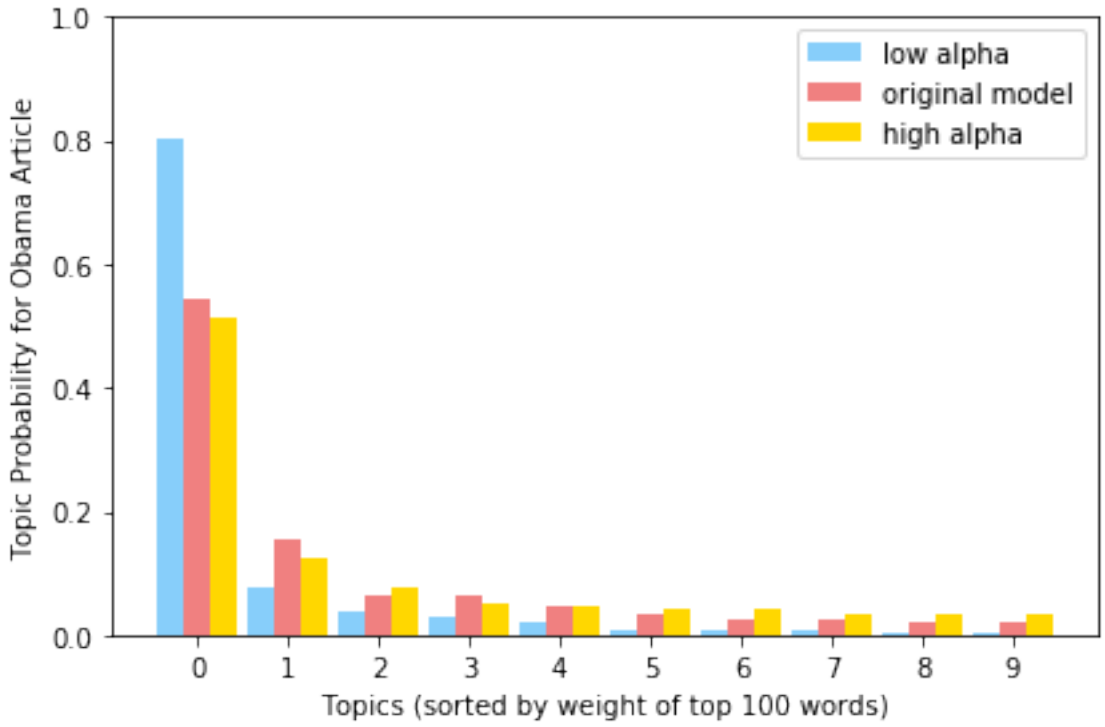
Below is a plot the total weight assigned by each topic to its top 10 words.



This shows that, for this topic model, the top 10 words only account for a small fraction (in this case, between 5% and 13%) of their topic's total probability mass. So while we can use the top words to identify broad themes for each topic, we should keep in mind that in reality these topics are more complex than a simple 10-word summary.

Effect of Alpha

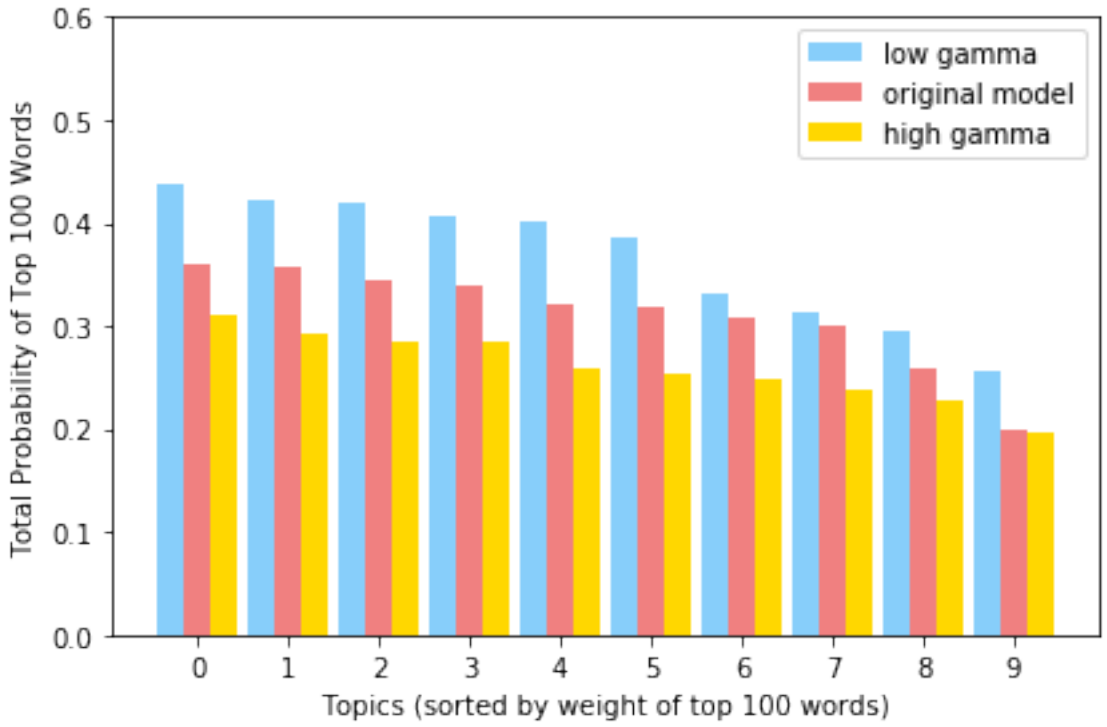
Alpha is responsible for smoothing document preferences over topics. Changing its value is visible in the plot of the distribution of topic weights for the same document under models fit with different alpha values.

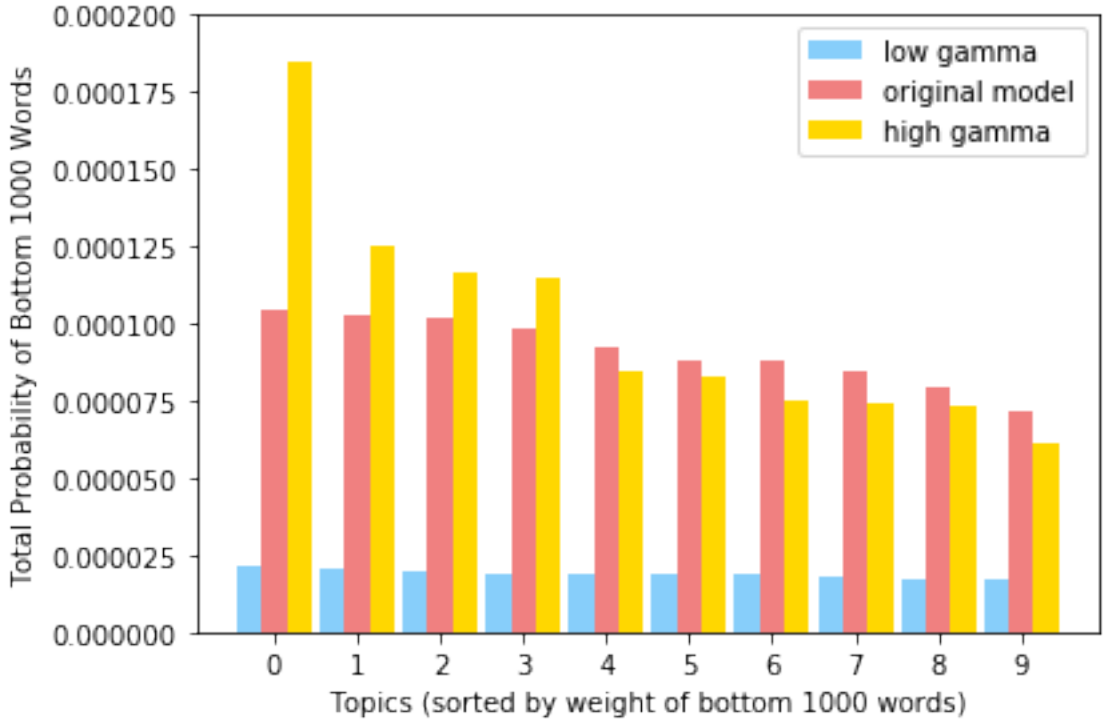


The plot shows that when alpha is low most of the weight in the topic distribution for this article goes to a single topic, but when alpha is high the weight is much more evenly distributed across the topics.

Effect of Gamma

We can visualize the impact of changing gamma by plotting word weights for each topic. Below is a plot of the total weight of the top 100 words and bottom 1000 words for each topic in high, medium, and low gamma models





These two plots show that the low gamma model results in higher weight placed on the top words and lower weight placed on the bottom words for each topic, while the high gamma model places

relatively less weight on the top words and more weight on the bottom words. Thus increasing gamma results in topics that have a smoother distribution of weight across all the words in the vocabulary.