Harvard University

Applied Mathematics 207 - Monte Carlo Methods, Stochastic Optimization

Final Project

Collapsed Gibbs Sampler for LDA to Classify Books by Thematic Content

Authors:
Cole Diamond
Raphael Pestourië
Wei Dai

Professors:
Pavlos Protopapas
Verena Kaynig-Fittkau
Teaching Fellow:
Rafael Martinez Galarza

May 11, 2015

1 Abstract

The Latent Dirichlet Allocation is a generative mixed membership model for topic recognition. In LDA, each document is assigned a distribution over a set of latent topics, and in turn, each topic is assigned a distribution over the corpus' vocabulary. We use collapsed Gibbs sampling to sample from the posterior of the LDA distribution. Thereafter, we train on half of the pages of ten classic novels, and perform inference on the remaining half. Using the topic distribution as a unique signal for classification, we perform nearest neighbors on the queried topic distributions of our test set to choose the closest match in our trained dictionary of topic distributions. We achieve 100% accuracy in recovering the true labels of the test set.

2 Introduction

Significant progress has been made on this problem by researchers in the field of Information Retrieval. The sections below describe the evolution in the methodology for text-mining from the most naive to the state-of-the-art.

2.1 TF-IDF

The text-frequency inverse-document frequency scheme TFIDF [**1**] is a numerical statistic that is intended to reflect how important a word is to a document in a corpus. The tf-idf value increases proportionally to the number of frequency of a word in a document, but is offset by the frequency of the word in the corpus. This accounts for words that appear more frequently in general. Tf-idf reduction can perform basic identification of sets of words that are discriminative for documents in the collection. However, tf-idf provides a relatively small amount of reduction in description length and reveals little by way of the way of inter or intradocument statistical structure.

2.2 LSI

Latent Semantic Indexing LSI [**2**] uses a singular value decomposition of the X matrix to identify a linear subspace in the space of tf-idf features that captures most of the variance in the collection. The reduction in dimensionality addresses the first problem of tf-idf but still does not give information on the inter and intra-document structure

2.3 pLSI

The probabilistic Latent Semantic Indexing pLSI [**3**] model attempts to relax the simplifying assumption made in the mixture of unigrams model that each document is generated from only one topic.

pLSI posits that a document label d and a word w_n are conditionally independent given an unobserved topic z.

$$p(d, w_n) = p(d) \sum_z p(w_n|z)p(z|d)$$

Each word is generated from a single topic, and different words in a document may be generated from different topics. Moreover, each document is represented as a list of mixing proportions for these mixture components and thereby reduced to a probability distribution on a fixed set of topics. Although pLSI does give information on the inter and intra-document structure, it has the following shortcomings

(1) The number of parameters in the model grows linearly with the size of the corpus, which leads to over-fitting.
(2) It is not clear how to assign probability to a document outside of the training set.

2.4 LDA

In the generative LDA process. documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words. LDA overcomes both of pLSI's problems by treating the topic mixture weights as a k-parameter hidden random variable rather than a large set of individual parameters which are explicitly linked to the training set. LDA [**4**] Consequently, LDA generalizes easily to new documents. Furthermore, the parameters in a k-topic LDA model do not grow with the size of the training corpus. Gibbs sampling can be used to perform learning with LDA. The GibbsLDA [**5**]. LDA assumes the following generative process for a corpus D consisting of M documents each of length N_i:

- Choose $N \sim poisson(\xi)$.
- Choose $\theta \sim Dir(\alpha)$.
- For each of the N words w_n:
  - Choose a topic $z_n \sim Mult(\theta)$.
  - Choose a word w_n from $p(w_n|z_n, \beta)$, a multinomial probability conditioned on the topic z_n.

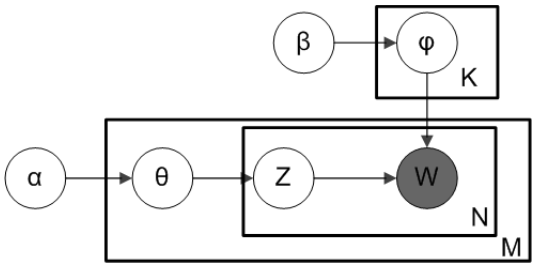

Figure 1: Plate Notation for LDA

2.5 Goal

We hypothesize that thematic content can be used as a signal for recovering a document's label. Our theory is predicated on the belief that topics across a book will remain consistent. To test our theory, we use LDA to identify the topic distribution of excerpts from ten classic novels publicly available on Project Gutenberg. These novels are Beowulf, The Divine Comedy by Dante Alighieri, Dracula by Bram Stoker, *Frankenstein* by Mary Shelley, *The Adventures of Huckleberry Finn* by Mark Twain, *Moby Dick* by Herman Mellville, Sherlock Holmes by Sir Arthur Conan Doyle, *Tale of Two Cities* by Charles Dickens, *The Republic* by Plato, *Ulysses* by James Joyce.

Thereafter, we perform inference on a set of unseen excerpts from these same documents. To do so, we use our trained model on the unobserved documents to determine their respective topic distributions. For each test document, we compare its topic distribution with our training data, and assign its label to the closest match.

3 Methodology

3.1 Gibbs Sampling Update Derivation

In order to uncover the latent topic assignments for each word in our set of documents, we will sample from the posterior of the LDA distribution using Gibbs sampling. To that end, we will derive the update step for the topic assignment.

3.1.1 Notation

- α is the parameter of the Dirichlet prior on the per-document topic distributions,

- β is the parameter of the Dirichlet prior on the per-topic word distribution,
- θ_i is the topic distribution for document i,
- ϕ_k is the word distribution for topic k,
- z_{ij} is the topic for the jth word in document i, and
- w_{ij} is the specific word.
- $n^i_{j,r}$ is the 3D matrix where i stands for an assigned topic for the rth word in the vocabulary in the document number j.
- $n^{i,-(m,n)}_{j,r}$ is a matrix to store the counts of a specific word (m, n) - the nth word in the mth document

3.1.2 Joint probability

Drawing from the Bayesian network in Figure 1, the joint probability of our model is :

$$P(\boldsymbol{W}, \boldsymbol{Z}, \boldsymbol{\theta}, \boldsymbol{\varphi}; \alpha, \beta) = \prod_{i=1}^{K} P(\varphi_i; \beta) \prod_{j=1}^{M} P(\theta_j; \alpha) \prod_{t=1}^{N} P(Z_{j,t}|\theta_j) P(W_{j,t}|\varphi_{Z_{j,t}}),$$

We integrate out the variables $\{\boldsymbol{\varphi}\}$ and $\{\boldsymbol{\theta}\}$ so that we do not need them in our updates

$$P(\boldsymbol{Z}, \boldsymbol{W}; \alpha, \beta) = \int_{\boldsymbol{\theta}} \int_{\boldsymbol{\varphi}} P(\boldsymbol{W}, \boldsymbol{Z}, \boldsymbol{\theta}, \boldsymbol{\varphi}; \alpha, \beta)\, d\boldsymbol{\varphi}\, d\boldsymbol{\theta} \tag{1}$$

$$\tag{2}$$

First, we consider the integration of independent θs

$$\int_{\boldsymbol{\theta}} \prod_{j=1}^{M} P(\theta_j; \alpha) \prod_{t=1}^{N} P(Z_{j,t} \mid \theta_j)\, d\boldsymbol{\theta} \tag{3}$$

Since the θs are independent, we can move the product over the topic probabilities outside of the integral. Then, it is evident that the resulting expression is a Dirichlet distribution.

In the right term of the integral, the product of the conditional probabilities over all the words in the document is equivalent to the product over the topics of the of the values of the θ vector for this document at the power of the number of word we count in this topic in the count matrix: $\prod_{i=1}^{K} \theta_{j,i}^{n^i_{j,(\cdot)}}$.

So the θ_j integration formula can be changed to:

$$\prod_{j=1}^{M} \int_{\theta_j} \frac{\Gamma\left(\sum_{i=1}^{K} \alpha_i\right)}{\prod_{i=1}^{K} \Gamma(\alpha_i)} \prod_{i=1}^{K} \theta_{j,i}^{n^i_{j,(\cdot)} + \alpha_i - 1}\, d\theta_j. \tag{4}$$

We have now shown that the the multinomial and the Dirichlet are conjugate distributions. Manipulating the coefficients of the Dirichlet and using the fact that a density of probability sums up to 1.

The integration of the product of the independent theta terms is therefore:

$$\int_{\theta_j} \prod_{j=1}^{M} P(\theta_j; \alpha) \prod_{t=1}^{N} P(Z_{j,t} \mid \theta_j)\, d\theta_j = \prod_{j=1}^{M} \frac{\Gamma\left(\sum_{i=1}^{K} \alpha_i\right)}{\prod_{i=1}^{K} \Gamma(\alpha_i)} \frac{\prod_{i=1}^{K} \Gamma(n^i_{j,(\cdot)} + \alpha_i)}{\Gamma\left(\sum_{i=1}^{K} n^i_{j,(\cdot)} + \alpha_i\right)}$$

In a similar manner, the integration of the product of the independent ϕ terms is:

$$\int_{\boldsymbol{\varphi}} \prod_{i=1}^{K} P(\varphi_i; \beta) \prod_{j=1}^{M} \prod_{t=1}^{N} P(W_{j,t} \mid \varphi_{Z_{j,t}})\, d\boldsymbol{\varphi} = \prod_{i=1}^{K} \frac{\Gamma\left(\sum_{r=1}^{V} \beta_r\right)}{\prod_{r=1}^{V} \Gamma(\beta_r)} \frac{\prod_{r=1}^{V} \Gamma(n^i_{(\cdot),r} + \beta_r)}{\Gamma\left(\sum_{r=1}^{V} n^i_{(\cdot),r} + \beta_r\right)}. \tag{5}$$

As a reminder, the conditional probability is the product of the the θ and ϕ terms.

3.1.3 Derivation of the Conditional Probability

We now use collapsed Gibbs Sampling to approximate the distribution of $P(\boldsymbol{Z} \mid \boldsymbol{W}; \alpha, \beta)$.

We want to find the conditional probability of the topic of the nth word in the mth document knowing not only all the topics of the other words in this document, but also all the topics of the word in all other documents. Without loss of generality, we state that the (m, n) word is the v'th word in the corpus' vocabulary.

$$P(Z_{(m,n)} \mid \boldsymbol{Z_{-(m,n)}}, \boldsymbol{W}; \alpha, \beta) = \frac{P(Z_{(m,n)}, \boldsymbol{Z_{-(m,n)}}, \boldsymbol{W}; \alpha, \beta)}{P(\boldsymbol{Z_{-(m,n)}}, \boldsymbol{W}; \alpha, \beta)}, \tag{6}$$

We now consider $P(Z_{(m,n)} = k \mid \boldsymbol{Z_{-(m,n)}}, \boldsymbol{W}; \alpha, \beta)$. From now onward, we only consider proportionality, which allows us the liberty to drop normalization factors.
Writing the joint probability and keeping only the terms depending on m and v that are not normalization factors, we find:

$$P(Z_{(m,n)} = k \mid \boldsymbol{Z_{-(m,n)}}, \boldsymbol{W}; \alpha, \beta) \propto \prod_{i=1}^{K} \Gamma(n^{i}_{m,(\cdot)} + \alpha_i) \prod_{i=1}^{K} \frac{\Gamma(n^{i}_{(\cdot),v} + \beta_v)}{\Gamma\big(\sum_{r=1}^{V} n^{i}_{(\cdot),r} + \beta_r\big)}.. \tag{7}$$

Now, using the notation mentioned above and using the factorial property of the gamma function: $\Gamma(n+1) = (n)!$. The $+1$ in the Gamma function comes from the fact that we assign this word to a topic, thus incrementing the matrix by one unit.

$$P(Z_{(m,n)} = k \mid \boldsymbol{Z_{-(m,n)}}, \boldsymbol{W}; \alpha, \beta) \propto \prod_{i \neq k} \Gamma(n^{i,-(m,n)}_{m,(\cdot)} + \alpha_i) \prod_{i \neq k} \frac{\Gamma(n^{i,-(m,n)}_{(\cdot),v} + \beta_v)}{\Gamma\big(\sum_{r=1}^{V} n^{i,-(m,n)}_{(\cdot),r} + \beta_r\big)} \tag{8}$$

$$\times \Gamma(n^{k,-(m,n)}_{m,(\cdot)} + \alpha_k + 1) \frac{\Gamma(n^{k,-(m,n)}_{(\cdot),v} + \beta_v + 1)}{\Gamma\big((\sum_{r=1}^{V} n^{k,-(m,n)}_{(\cdot),r} + \beta_r) + 1\big)} \tag{9}$$

$$P(Z_{(m,n)} = k \mid \boldsymbol{Z_{-(m,n)}}, \boldsymbol{W}; \alpha, \beta) \propto (n^{k,-(m,n)}_{m,(\cdot)} + \alpha_k) \frac{n^{k,-(m,n)}_{(\cdot),v} + \beta_v}{\sum_{r=1}^{V} n^{k,-(m,n)}_{(\cdot),r} + \beta_r}. \tag{10}$$

The is the conditional probability derived from the count matrix that we will be using for our updates.

With our conditional probability update in hand, we can implement the algorithm in Figure 2 to train our model.

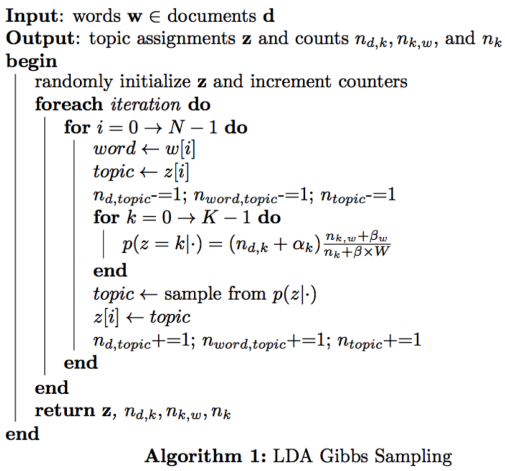

Figure 2: Collapsed Gibbs Algorithm from Darling [**5**]

3.2 Pre-processing

We sanitize our documents to reduce training time and improve the predictive capability of our model. First, we remove punctuation and numbers from our books. Additionally, we remove stop words, or words that have little lexical meaning, ie: "the, is, at, which, on..." We use porter stemming to normalize plurals and conjugated verbs to the same token, thereby reducing the size of our vocabulary. We then map documents to their index in the vocabulary dictionary to produce document vectors. By translating our documents into the language of numbers, we are able to perform operations on our data more easily.

Next, we remove words that appear less than ten times in all documents, and words that appear in more than 90% of the documents. We posit that such frequent intra-document words, and rare tokens will contribute very little to the signal we use to distinguish documents. Thereafter, we allocate half of the words in each of the books to the training set and test set. Lastly, we construct a count matrix by setting each row equal to the frequency of a token in a document. The LDA algorithm will use the count matrix as input.

3.3 Libraries

Although we wrote the collapsed Gibbs Sampler by hand, we used several libraries to ameliorate implementation and analysis, which are listed in the table below.

Library	Purpose
Numpy	Matrix multiplication; serialization; array manipulation
Scipy	Statistical sampling
Codecs	Decoding text as UTF-8
NLTK	Porter Stemming, Stop Words
Matplotlib	Plotting (line charts, bar charts, autocorrelation, 3D)
Seaborn	Plotting (heatmap)
Pandas	Visualizing data as a table
Collections	Default dictionary, and counter
Wordcloud	Create word-cloud of per-topic word distribution

4 Results and Discussion

4.1 Convergence

4.1.1 Log Likelihood

We verify that the likelihood that our model generated the data increases over ever iteration. For convergence, we want to see a plateau, such that we are seeing diminishing gains in our log likelihood. We can see in the graph below that our log likelihood converges asymptotically at around 10-15 iterations. As the graph below illustrates, this is exactly the case.

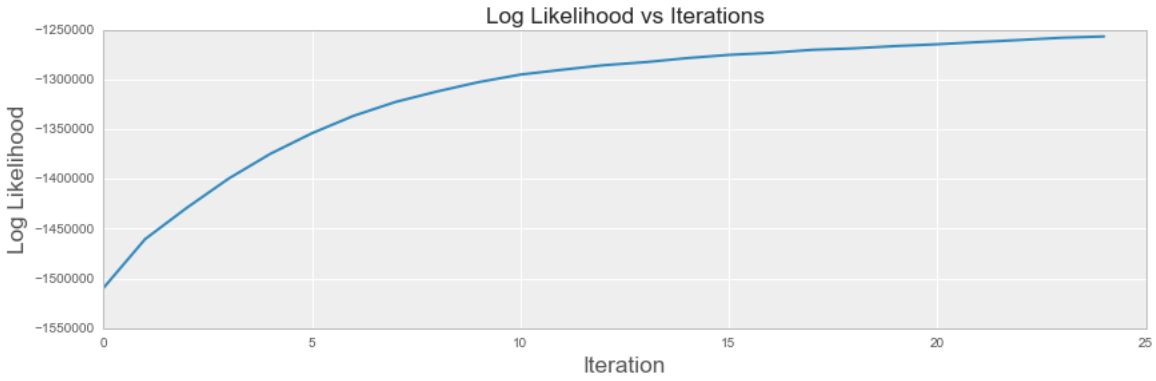

Figure 3: Log Likelihood for the Collapsed Gibbs Algorithm

4.1.2 Aggregate Swaps

We present a custom statistic to measure the total number of words whose topic assignment changed between iterations. We know that if the algorithm converges, the number of swaps every iteration should level out. The graph below illustrates this trend.

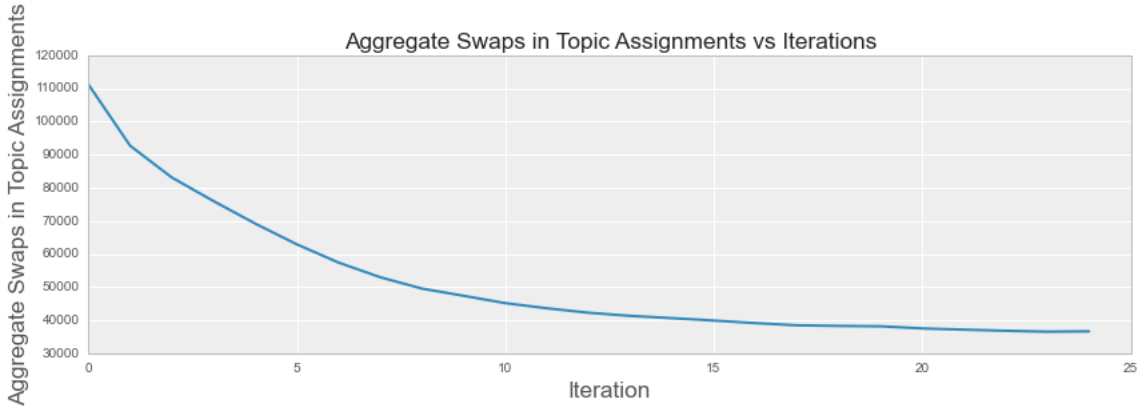

Figure 4: Aggregate Topic-Swaps in the Collapsed Gibbs Algorithm

We also verify that the autocorrelation of the number of swaps decreases with a larger number of lags.

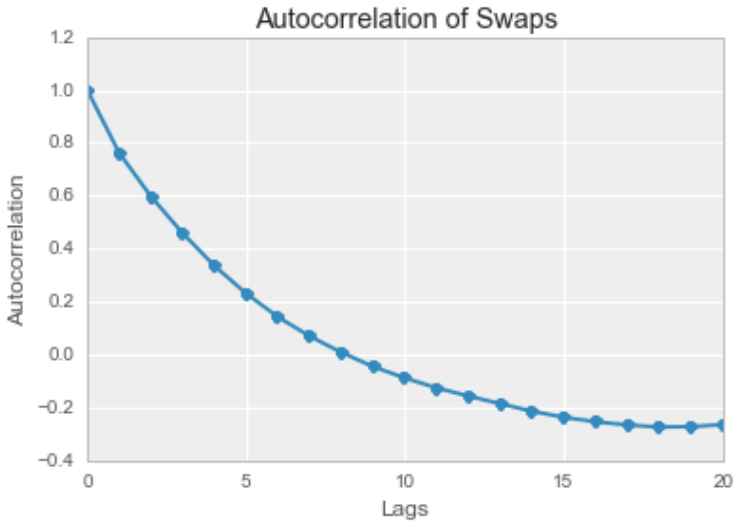

Figure 5: Auto-correlation of Swaps

4.2 Visualization

4.2.1 Topics as a Distribution over Words

In order to gain insight about the quality of topics that our algorithm detects, we can visualize topics as a word-cloud of its constituent vocabulary. In the word-clouds below for two randomly selected topics, the size of each of the words is directly proportional to its importance in the topic. We also use the visualization to verify that we observe only a few high-mass words per topic as we set our beta parameter to a small number (.05).

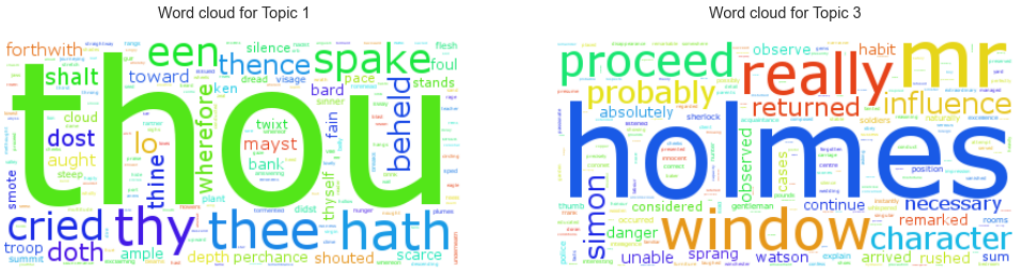

Figure 6: Word Cloud for Two Topics

4.2.2 Documents as a Distribution Over Topics

We calculate each train document's topic distribution. We then verify that each document has a unique topic distribution signature, allowing for classification. The visualizations below also illustrate that we observe few high-mass topics per document since we set our alpha parameter to a small number (.10).

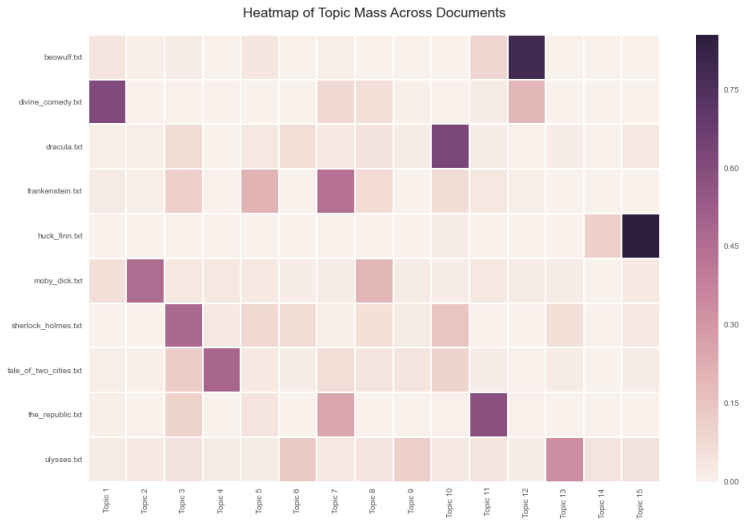

Figure 7: A Heatmap of our Topics over Documents

The stacked barchart below also represents the normalized weight of topics in a document.

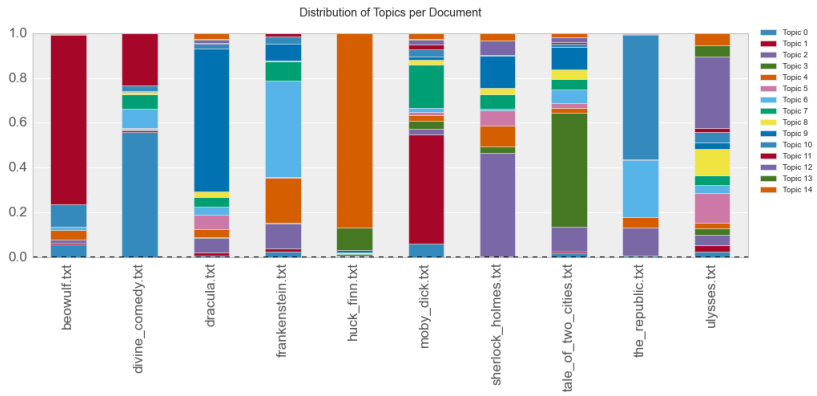

Figure 8: Stacked Barchart of the Normalized Topic Weight Document

We also create a landscape to visualize the density of topics for documents.

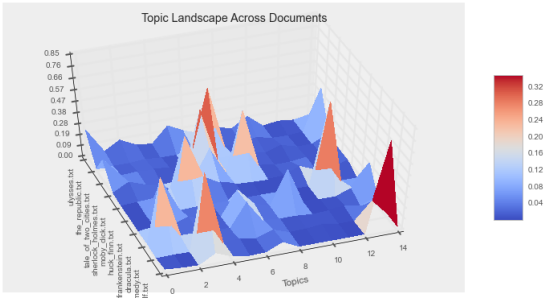

Figure 9: Topic Landscape Across Documents

4.3 Prediction

As a reminder, our assumption is that topics across a book will remain consistent. Intuitively, if the distribution of topics of our test document is very similar to one of the reference topic distribution, it is more likely that the test document actually comes from this reference book. We model this 'similarity' using the L2 norm between the test distribution against the reference distribution of each of the 10 books. In order to obtain a probability, which we deem the predictive probability, we take the inverse of this distance and normalize it so that the sum of the probabilities of the test document being from any of the books in our test set is 1.

For each word, we take the topic that maximizes the conditional distribution for each word. Thereafter, we can find the document with the most similar topic distribution simply by computing the frobenius norm. We repeat the above steps several times so that we can have means and standard deviations for our predictions.

Since we may have different votes per iteration, we choose the mode of the prediction for each book. The figure below illustrates the probability for classification of each label for each of the test documents.

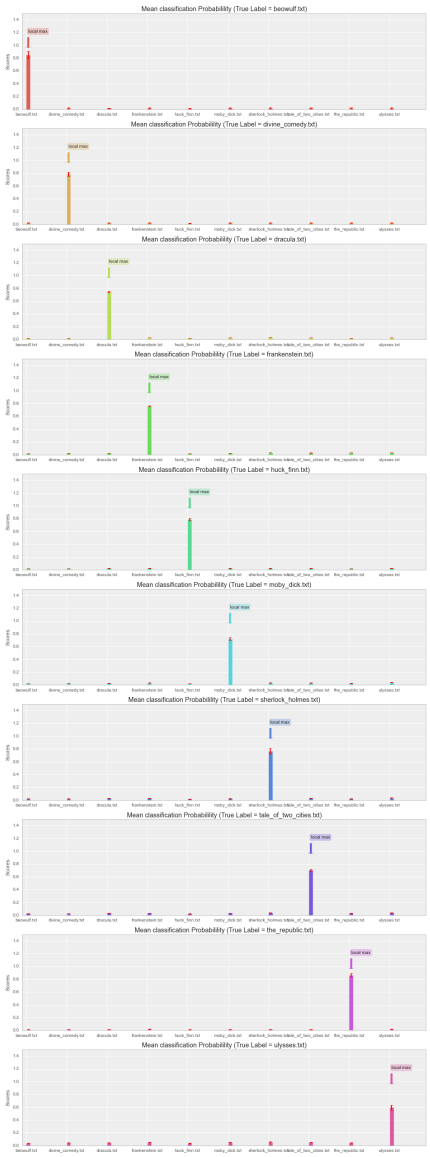

Figure 10: Mean Classification Probability for Each Test Document

4.4 Sensitivity of Model to Length of Test Data

We want to quantify the robustness of our model to the length of the test data. The test phase proceeds as follow. We take a test excerpt, with a given length from one of the ten books. We run the LDA model on the test data, fixing the word $\times$ topic matrix to the matrix obtained from the training phase. Iterating this process across all words in the test document, we recover the distribution of the topics in this test document.

We calculate the predictive probability for test documents of different lengths. For each length, we start the test document at 30 different random points in order to get a mean prediction probability and a standard deviation. The figure below is a plot of the predictive probability against the length of the test set. We can see a logarithmic evolution of the probability against the number of words in the test set.

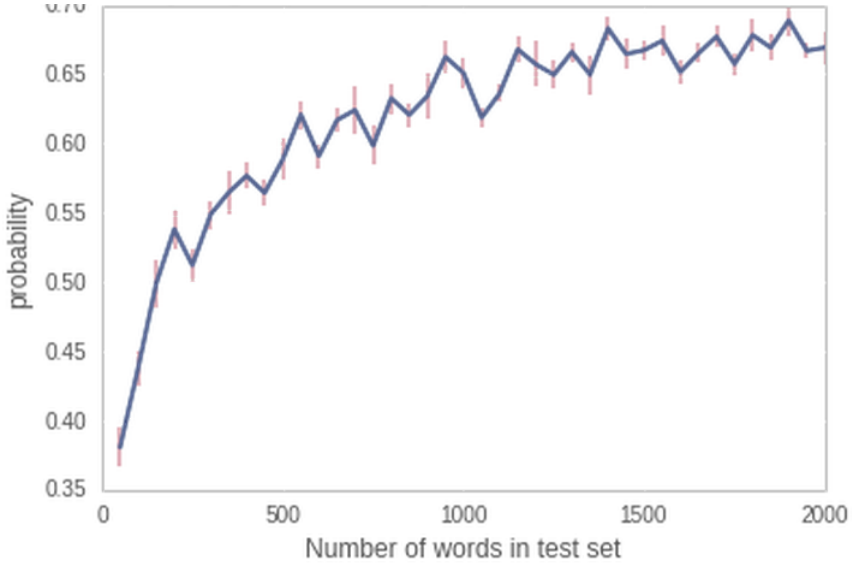

Figure 11: Prediction Probability of True Label vs Length of the Test Set

4.5 Document Generation

In this section, we apply the LDA Generative Model to "create" new pages of *The Adventures of Huckleberry Finn*. To generate words from a document, we apply the following scheme.

```
For s sentences:
    For n words:
        1. Sample a topic index from the topic proportions found in Huckleberry Finn
        2. Sample a word from the Multinomial corresponding to the topic index from 2)
```

Some of the sentences that we generated are listed below. Note the fact that we use a bag of word approximation means that the model has no clue about grammar or order a word. For the LDA model a book, and a book with all its word sorted is the same thing, even though the latter is not readable for a human. A possible way forward -that we could not achieve because of lack of computational power and memory- would be to build a transition matrix between any word to any other word, as we did in the homework about Viterbi algorithm on 'eggs and ham'.

```
Pewter handy funeral im listened kinder warnt average theyve reckon.
Shaming furnaces scared canoe canoe im dont sneak dont aint.
Slip couple tom jane confining spoon mouth stopped blowing flabby.
Erroneously invite maybe theyre endowed disappears desperadoes pretty outwardly tom.
Cant dropped cabin madam susan aint count borrowing hiding picked.
```

5 Conclusion

We trained an LDA model on half the pages of ten classic books, the other half is used for testing. Given the test data and our model, we perform inference on the new text to determine the topic distribution. We compare the queried topic distribution with our training data, and assign it to the closest match. Our hypothesis that thematic content would be a good signal for identifying texts was valid. We achieved a perfect classification of our query text. We also explored the sensitivity of our model to number of words, and used the LDA generative model to construct new sentences from a given book. Future work may use bigrams or n-grams to map to topics, instead of unigrams to perform classification.

Appendix

Data

1. Frankenstein - https://www.gutenberg.org/cache/epub/84/pg84.txt
2. The Adventures of Sherlock Holmes - https://www.gutenberg.org/cache/epub/1661/pg1661.txt
3. A tale of two cities - https://www.gutenberg.org/cache/epub/98/pg98.txt
4. Moby Dick - https://www.gutenberg.org/cache/epub/2701/pg2701.txt
5. Beowulf - https://www.gutenberg.org/cache/epub/16328/pg16328.txt
6. Dracula - https://www.gutenberg.org/cache/epub/345/pg345.txt
7. The Adventures of Huckleberry Finn - https://www.gutenberg.org/cache/epub/76/pg76.txt
8. Ulysses - https://www.gutenberg.org/cache/epub/4300/pg4300.txt
9. The Republic - https://www.gutenberg.org/cache/epub/1497/pg1497.txt
10. The Divine Comedy - https://www.gutenberg.org/cache/epub/8800/pg8800.txt

References

[1] G. Salton and M. McGill, editors. Introduction to Modern Information Retrieval. McGraw-Hil. 1983.

[2] S. Deerwester, S. Dumais, T. Landauer, G. Furnas, and R. Harshman. Indexing by latent semantic analysis. Journal of the American Society of Information Science, 41(6):391–407, 1990.

[3] T. Hofmann. Probabilistic latent semantic indexing. Proceedings of the Twenty-Second Annual International SIGIR Conference, 1999.

[4] Blei, David M and Ng, Andrew Y and Jordan, Michael I, Latent dirichlet allocation. The Journal of machine Learning research, (3):993–1022, 2003

[5] Darling, W. M. A Theoretical and Practical Implementation Tutorial on Topic Modeling and Gibbs Sampling. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, (Portland, Oregon, USA, 2011).