Comparing Topic Models on Classification Tasks

Thomas W. Jones

06 May, 2019

Abstract

Topic models are hard to evaluate. The variables they measure cannot be observed. Most evaluation has focused on coherence metrics to mimic human judgment. Yet topic models are often used to support classification tasks, where ground truth exists. This research compares three topic models based on how well they support a classification task. I perform two experiments. In the first, hyperparameters for each model are fixed, putting models on equal footing. In the second, an optimization service—SigOpt—aids hyperparameter settings so that each model is free to be its best. LDA performs consistently better than both LSI and ProdLDA. ProdLDA, in spite of having high coherence scores, performs poorly on classification.

1 Introduction

Evaluating topic models is problematic. Topic models are latent variable models, meaning the "topics" they create cannot be observed in the real world. To date, there is no single acceptable method for evaluating topic models. The closest to a consensus is a class of measures called "coherence". Coherence measures calculate the degree to which the highest scored terms in a topic belong together. Coherence metrics purport to correlate highly with human judgement. Yet this approach is not holistic. Coherence does not measure goodness-of-fit of the topic model, nor does coherence measure how well a topic model aids a task to which it is applied (e.g. document classification).

In this paper, I take a step towards a more holistic means of evaluating topic models. I examine two gold-standard topic models—Latent Dirichlet Allocation (LDA) and Latent Semantic Indexing (LSI)—and a new neural network-based topic model—ProdLDA. I evaluate these models across three dimensions: a coherence metric, a goodness of fit metric, and a classification error metric. These metrics and the experiments are described in more detail throught the paper.

The remainder of this paper is organized as follows: Section 2 gives a more-detailed statement of the problem and research focus of this paper. Section 3 reviews several previous works relevant to this research. Section 4 provides an overview of computational methods employed. Section 5 details the experiment and results. Section 6 gives some conclusions and directions for future work.

2 Background

2.1 Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) is a Bayesian model of language. It models an idealized stochastic process for how words get on the page. Instead of writing full, syntatictically-coherent, sentences, the author samples a topic from a multinomial distribution and then given the topic samples a word. The process for a single draw of the n-th word for the d-th document, $w_{d,n}$, is

1. Sample a topic $z_{d,n} \sim \text{Multinomial}_K(1, \boldsymbol{\theta}_d)$
2. Given topic $z_{d,n}$, sample a word $w_{d,n} \sim \text{Multinomial}_V(1, \boldsymbol{\phi}_{z_{d,n}})$

The variable $z_{d,n}$ is latent. The author repeats this process N_d times until the document is "complete". For a corpus of D documents, V unique tokens, and K latent topics, the goal is to estimate two matrices: $\boldsymbol{\Theta}$

and $\mathbf{\Phi}$. The d-th row of $\mathbf{\Theta}$ comprises $\boldsymbol{\theta}_d$, above. And the k-th row of $\mathbf{\Phi}$ comprises $\boldsymbol{\phi}_k$. LDA estimates these parameters by placing Dirichlet priors on $\boldsymbol{\theta}_d$ and $\boldsymbol{\phi}_k$.[1] Namely, $\boldsymbol{\theta}_d \sim \text{Dirichlet}_K(\boldsymbol{\alpha})$ and $\boldsymbol{\phi}_k \sim \text{Dirichlet}_V(\boldsymbol{\beta})$.

Taken together, the dot product of $\mathbf{\Theta}$ and $\mathbf{\Phi}$—along with the length of each document—form the expected term frequencies under the model. Specifically,

$$\mathbf{E}(\mathbf{W}) = \mathbf{n} \odot (\mathbf{\Theta} \cdot \mathbf{\Phi}) \tag{1}$$

where $\mathbf{W}$ is a matrix of counts of terms across documents, $\mathbf{n}$ is a vector of document lengths, and $\odot$ denotes elementwise multiplication.

2.1.1 Computing LDA

Two common methods for estimating LDA are Gibbs sampling and variational Bayesian inference (VI). There are tradeoffs between the two methods from both computational and theoretical perspectives. VI can be theoretically faster to compute and scale better with large data. However, unlike Gibbs, it does not guarantee convergence to the true posterior as specified.

Gibbs sampling is a Markov chain Monte Carlo (MCMC) algorithm. MCMC is used for Bayesian inference in cases—such as LDA—where a closed form solution for the posterior of a model is unobtainable. MCMC algorithms are sequential and obtain a sequence of posterior samples. When the samples are independent, the algorithm is said to have converged. The Gibbs algorithm, in its simplest form—where μ is the parameter of interest, γ is a hyperparameter for the prior on μ, and X is the observed data—is as follows:

1. Initialize $\mu^{(0)}$ at random
2. For each iteration t do: Sample $\mu^{(t)}$ from $P(\mu^{(t-1)}|X, \gamma)$
3. Assess convergence and discard samples prior to the algorithm converging

Gibbs sampling for LDA was proposed by Griffiths and Steyvers in 2004. The specific details for Gibbs in LDA aside, the process is generally the same as the above. Unfortunately, Gibbs is slow. Each iteration requires a pass through each topic, each document, and each occurrence of each token within that document. Some efforts have been made to speed up this process through parallelism. (Wang et al., 2009) But speed remains a concern.

An alternative is VI, which was used by Blei et al. in their 2003 work introducing LDA. VI is an optimization method and is easily parallelized. The objective is to minimize the KL divergence between the posterior and prior whiel maximizing the likelihood of the data given the model. This results in an approximate posterior over which the optimization problem is defined. Like most optimization methods, VI may find a local, rather than global, optimum. A form of VI using a variational autoencoder is used to estimate ProdLDA, described below.

2.2 Latent Semantic Indexing

LSI—also referred to as latent semantic analysis (LSA)—is arguably the first topic model. Deerwester et al. published the paper introducing LSI in 1988. LSI uses a single-value decomposition of a document term matrix (DTM) or term frequency-inverse document freqency (TF-IDF) matrix. A DTM contains counts of occurence of terms within documents. A TF-IDF matrix reweights these raw counts according to a vector inversely proportional to the number of documents in which a term appears. Using the notation used in LDA, above, we can write LSI as

[1]The frequentist equivalent of this model is probabilistic latent semantic analysis (pLSA), usually estimated with the EM algorithm.

$$\mathbf{W} = \mathbf{\Theta\Sigma\Phi} \tag{2}$$

where $\mathbf{W}$ is a DTM or TF-IDF matrix and $\mathbf{\Sigma}$ is a diagonal matrix of singular values. The rows of $\mathbf{\Theta}$ contain a non-probabilistic distribution of topics over documents. The rows of $\mathbf{\Phi}$ contain a non-probabilistic distribution of words over topics.

2.3 ProdLDA

In a 2017 ICLR conference paper, Srivastava and Sutton introduce the use of an autoencoder for VI to estimate LDA. They also propose a new topic model—ProdLDA—computed in a similar manner.

2.3.1 Autoencoding Variational Inference for Topic Models (AVITM)

Autoencoders are neural networks where the input and output layers are the same. Typically, an autoencoder consists of two stages. The encoding stage transforms the input, $\mathbf{W}$ into a hidden, latent, and usually lower-dimensional representation, $\mathbf{\Theta}$. The decoding stage transforms $\mathbf{\Theta}$ into an approximation of $\mathbf{W}$.

Variational autoencoders introduce randomness into the mix. Instead of being encoded to $\mathbf{\Theta}$ directly, $\mathbf{W}$ is encoded into vectors of parameters to a probability distribution. Then the rows of $\mathbf{\Theta}$ are sampled from that distribution and decoded, as above. The objective function is conceptually the same as in VI. Thus, this method is VI using variational autoencoders, hence the name "autoencoding variational inference for topic models."

Srivastava and Sutton find that they must overcome several practical challenges to apply AVITM for LDA. First, they use a Laplace approximation to the Dirichlet distribution because of the difficulty parameterizing it in the context of variational autoencoders. They use the ADAM optimizer with high momentum and learning rate to avoid getting trapped in bad local optima. To offset the risk of divergence caused by the high learning rate, they use several batch normalization layers. Finally, they use a 50% dropout layer when computing $\mathbf{\Theta}$ to get the network to "use more of its capacity."

2.3.2 ProdLDA

ProdLDA losens a constraint in the formation of $\mathbf{\Phi}$. Under LDA, the rows of $\mathbf{\Phi}$—denoted $\boldsymbol{\phi}_k$ for the k-th topic—are constrained to parametarize a multinomial symplex during training. ProdLDA lifts this constraint. Instead of words being drawn from the multinomial two-step described in Section 2.1, instead they are drawn subject to

$$\mathbf{W} \sim \text{Multinomial}_V(\mathbf{n}, \sigma(\mathbf{\Theta} \cdot \mathbf{\Phi})) \tag{3}$$

where $\mathbf{n}$ is a vector whose d-th entry is the number of tokens in the d-th document and $\sigma(\cdot)$ denotes the softmax function. ProdLDA is trained using AVITM described above.

As published, two challenges arise with ProdLDA. First, the calculations of $\mathbf{\Theta}$ and $\mathbf{\Phi}$ take place within layers of a neural network that are not constrained to create probability distributions. This is mitigated by applying a softmax function to the rows of each after extracting the appropriate hidden layer (for $\mathbf{\Theta}$) and weights (for $\mathbf{\Phi}$). Yet it is still not clear that this makes ProdLDA's outputs directly comparable to LDA. Second—and as pointed out by the AVITM paper's reviewers[2]—the authors do not explore the effects of hyperparameter selection on their results and on the topic models to which they compare. I correct for this latter issue in my experiments, below.

[2]ICLR's reviews are available publicly. See here: https://openreview.net/forum?id=BybtVK9lg

3 Related Work

Most research on topic model evaluation has focused on presenting ordered lists of words that meet human judgement about words that belong together. For each topic, words are ordered from the highest value of $\boldsymbol{\phi}_k$ to the lowest. In 2009 Chang et al. introduced the "intruder test." Judges are shown a few high-probability words in a topic, with one low-probability word mixed in. Judges must find the low-probability word, the intruder. They then repeat the procedure with documents instead of words. A good topic model should allow judges to easily detect the intruders.

Coherence metrics attempt to approximate the results of intruder tests in an automated fashion. Researchers have put forward several coherence measures, mostly these compare pairs of highly-ranked words within topics. In 2013 Rosner et al. evaluate several of these. They have human evaluators rank topics by quality and then compare rankings based on various coherence measures to the ranking of the evaluators. They express skepticism that existant coherence measures are sufficient to assess topic quality. In a 2014 ACL paper, Lau et al. find that normalized pointwise mutual information (NPMI) is a choerence metric that closely resembles human judgement. Srivastava and Sutton use NPMI as a coherence measure in their AVITM work.

Measuring goodness-of-fit in topic models is less explored.The primary goodness of fit measures in topic modeling are likelihood methods. Likelihoods, generally the log likelihood, are naturally obtained from probabilistic topic models. A popular likelihood method for evaluating out-of-sample fit is called perplexity. Perplexity measures a transformation of the likelihood of the held-out words conditional on the trained model. However, since Chang et al.'s 2009 *Reading Tea Leaves*, researchers have eschewed goodness of fit. I believe this is a mistake and have offered my own goodness-of-fit measure in a working paper (Jones 2015) described in section 4.3.1.

A recent work on word embeddings[3] provides inspiration for this research paper. OpenAI's researchers (Radford et al., 2019) develop a method for training text embeddings based on "generalized pre-training" or GPT. The core idea of GPT is that researchers use embeddings to aid some final task—such as document classification, question answering, calculating sentence similarity, et cetera. They use performance of the final model on these tasks to demonstrate the efficacy of their embedding model and training process.

4 Methods and Techniques

4.1 Experimental Setup

This research rates the efficacy of topic models on a classification task. For this I use the 20 Newsgroups data set (Mitchell 1996), described in more detail later in the paper. This experiment proceeds in four steps: data preparation, topic model training, classifier training, and evaluation. I perform two experiments during topic model training. In the *fixed experiment*, I set all hyperparameters to the same values (where applicable) across models. In the *optimized experiement*, I use a hyperparameter optimization service to specify each topic model.

4.1.1 Data Preparation

Data cleaning follows standard practice for bag-of-word models. All tokens are converted to lowercase. Punctuation and numbers are replaced with spaces. I tokenize on spaces, creating a corpus of unigrams. I remove stop words using two standard stopword lists: 175 English stop words from the snowball stemmer project and 571 "SMART" stop words.[4] Finally, I exclude words appearing in fewer than 5 documents.

[3]Word embeddings are similar to topic models. However, instead of modeling the distribution of words in an entire document, they model distributions of words related to other words in local contexts.

[4]For more information see https://rdrr.io/rforge/tm/man/stop words.html

I divide the data into two training sets and one test set. These sets contain 6,665; 6,665; and 6,667 observations respectively. This division accounts for uncertainty at each step. The first training set is used to train the topic models. Then $\hat{\boldsymbol{\Theta}}$ is predicted under the topic model for the second training set. The predicted $\hat{\boldsymbol{\Theta}}$ is training input to the classifier.

4.1.2 Topic Model Training

4.1.2.1 Training LDA

The hyperparameters for LDA are as follows: k is the number of topics; $\boldsymbol{\alpha}$ is the parameter for the Dirichlet prior of topics over documents; is the parameter for the Dirichlet prior of words over topics; and *iter* is the number of iterations of the Gibbs sampler.

For the fixed experiment I set k to 200. I set $\boldsymbol{\alpha}$ to a vector where each entry is 0.1. This is known as a *symmetric* prior. I set $\boldsymbol{\beta}$ to an *asymmetric* prior where the entries are proportional to the frequencies of each term in the training data.[5] For the fixed experiment I set the magnitude of $\boldsymbol{\beta}$ to 1,668.9.[6] I set the number of iterations to 300. The first 250 iterations are discarded and $\boldsymbol{\Theta}$ and $\boldsymbol{\Phi}$ are averaged over the remaining 50 iterations. For the fixed experiment I use all 6,665 records in training.

For the optimized experiement, I use the optimization service provided by SigOpt[7] to tune the symmetric entries of $\boldsymbol{\alpha}$, the magnitude (though not shape) of $\boldsymbol{\beta}$, and k. (Iterations are handeled the same as in the fixed experiment.) I use 2,000 randomly-sampled records from the first training set to train initial models and the remaining 4,665 records as a validation set. I use the the average of R^2 and mean coherence of the model applied to the validation set as the optimization objective. (Both coherence and R^2 are described in more detail below.) After 20 optimization iterations through the SigOpt service, I train a final model using "optimal" hyperparameters on all 6,665 records in the first training set.

4.1.2.2 Training LSI

LSI has only a single hyperparameter—k, as above. For the fixed experiment, I set k to 200 and train on training set 1. For the optimized experiment, I use the same training/validation split in the SigOpt service. The objective is only mean coherence as calculating R^2 for LSI is undefined. For LSI, I use TF-IDF rather than a DTM of word counts as in LDA and ProdLDA. I calculate the IDF term based on the first training set and then carry it forward when predicting topic distributions on the second training set and test set.

4.1.2.3 Training ProdLDA

ProdLDA has 4 hyperparameters: the number of topics, k; the parameter for the prior of topics over documents, $\boldsymbol{\alpha}$; the number of hidden nodes at each layer of the encoder network; and the number of training epochs.

In the fixed experiment, I set k and $\boldsymbol{\alpha}$ to the same values as in LDA. I use 100 hidden nodes for each of the two layers in the encoder network, the same value used in the AVITM paper. I set the number of training epochs to a maximum of 100, but allow for early termination using default Keras settings.

In the optimized experiment, SigOpt finds optimal settings for k, $\boldsymbol{\alpha}$, and the number of hidden nodes using the same objective and training/validation splits as LDA.

[5]This prior does not necessarily conform to conventional wisdom. The choice and its justification will be a significant point in my dissertation. In the meantime a proof implicating this choice may be found in an appendix here: https://drive.google.com/file/d/0Bz2enPyUvnKIQjRzZU9TWFJNcVhobWlWV180TmdmendzZ2JJ/view?usp=sharing

[6]This is the equivalent magnitude of a symmetric $\boldsymbol{\beta}$ where each entry is 0.05. This is a more conventional setting.

[7]https://sigopt.com/

4.1.3 Classifier Training

For each topic model, I use the same specifications on the classifier. It is a simple 5-layer feed-forward neural network. I use 30% dropout between layers, the ADAM optimizer, and the categorical crossentropy loss function. Each of the hidden layers use ReLU activation and the output layer has softmax activation. For the fixed experiment, I use 200 hidden nodes at each layer. For the optimized experiment, I use 862 nodes at each layer. In the latter case, I perform a SigOpt optimization run for the classifier to select the number of hidden nodes for each topic model. I then averaged the number of hidden nodes suggested by the optimizer across the three topic models.

4.2 Computational Methods

For both LDA and LSI, I use the *textmineR* package for the R language[8]. For ProdLDA, I rewrote the *Keras* implementation[9] of ProdLDA[10] in R using the *keras* package for R[11] with a Tensorflow backend. *textmineR* does much of the heavy lifting elsewhere. Data preparation, calculation of coherence, and calculation of R^2 are all done in *textmineR* too. The classifiers are trained using the *keras* package for R with a Tensorflow backend.

4.3 Evaluation Metrics

I use four evaluation metrics. I use F1 and accuracy for classification. The secondary metrics are probabilistic coherence—a coherence metric for the topic model—and R^2—a goodness-of-fit metric for the topic model. In all cases, the metrics are calculated on the test set. Accuracy and F1 are well-understood. Below is a brief description of R^2 and probabilistic coherence.

4.3.1 R^2 for Topic Models

R^2 for topic models relies on a geometric interpretation of the standard definition—$R^2 \equiv 1 - \frac{SSR}{SST}$. For a scalar outcome of each observation, y_i, and prediction, f_i, SSR is $\sum_i (y_i - f_i)^2$. Similarly, SST is $\sum_i (y_i - \frac{1}{n}\sum_i y_i)^2$. The sums of squares in SSR and SST may be thought of as a sum of squared euclidean distances in a single dimension. R^2 for topic models extends this definition to multiple dimensions, specifically the dimension of the vocabulary. Within the context of topic modeling, R^2 has the same interpretation that it does when used in a broader class of statistical models, as a proportion of variablity in the data explained by the model. However, generalizing to multiple dimensions causes the loss of a lower bound of zero. This is not problematic, however. Since an R^2 less than zero means that your model is a worse fit than just guessing that each document's word frequencies are equivalent to the mean across all documents. A full derivation and analysis are available in a working paper (Jones 2015).

4.3.2 Probabilistic Coherence

Probabilistic coherence is available in the *textmineR* package for R. It is a cohenrence measure based on the average difference between probabilities. For each pair of words $\{a, b\}$ in the top M words in a topic, probabilistic coherence calculates $P(b|a) - P(b)$, where a is more probable than b in the topic. $P(b|a)$ measures how probable b is only in documents containing a. $P(b)$ measures how probable b is in the corpus as a whole. If b is not more probable in documents containing b, then the difference $P(b|a) - P(b)$ is close to zero. For example, suppose the top 4 words in a topic are $\{a, b, c, d\}$. Then calculate (1) $P(a|b) - P(b)$, $P(a|c) - P(c)$, $P(a|d) - P(d)$, (2) $P(b|c) - P(c)$, $P(b|d) - P(d)$ and (3) $P(c|d) - P(d)$.

[8]https://cran.r-project.org/package=textmineR

[9]https://github.com/nzw0301/keras-examples/blob/master/prodLDA.ipynb

[10]Original source code for the AVITM paper here: https://github.com/akashgit/autoencoding_vi_for_topic_models

[11]https://cran.r-project.org/package=keras

And all 6 differences are averaged together, giving the probabilistic coherence measure. Probabilistic coherence is bound between 1 and -1, though in practice negative values are very close to zero. Values close to 0 indicate that words in a topic are statistically independent of each other, indicating a junk topic. Positive values indicate words in a topic are positively correlated and *not* independent.[12]

5 Discussion and Results

5.1 Data Set

For expiriments, I used the 20 Newsgroups data set (Mitchell 1996). This data set contains 19,997 posts across 20 different newsgroups. Each of the 20 newsgroups has 1,000 documents in the corpus with the exception of *soc.religion.christian* which has 997. After converting all words to lowercase and removing numbers and punctuation, the corpus has a vocabulary of 120,024 terms and a total of 7,383,852 unique tokens. The shortest document is 47 tokens and the longest is 38,944 with the median being 251. After removing stop words and infrequent terms, the vocabulary has 33,378 terms and a total of 3,759,122 unique tokens. The shortest document is now 34 tokens long and the longest is 12,719 with the median bieng 141.

5.2 Experimental Results

Table 1, below, shows hyperparameter settings and results for the fixed experiment. Focusing on F1 and accuracy, the core metrics, LDA is the clear winner and LSI comes in second. ProdLDA has a relatively high coherence, as advertised. However, it is not significantly higher than LDA's. R^2 for ProdLDA is poor.

Table 1: Parameters and evaluation metrics of the fixed experiment

	k	alpha	Sum(beta)	Nodes	coherence	R-sq.	Accuracy	F1
LDA	200	0.1	1668.9	-	0.16	0.80	0.82	0.82
LSI	200	-	-	-	0.10	-	0.63	0.69
ProdLDA	200	0.1	-	200	0.17	-0.01	0.41	0.41

Table 2, below, shows hyperparameter settings based on SigOpt optimization and evaluation metrics for the optimized experiment. Optimization resulted in more topics for all three models. In both the case of LDA and ProdLDA, the entries of $\boldsymbol{\alpha}$ are larger, indicating a flatter average distribution of topics within each document. The magnitude of $\boldsymbol{\beta}$ is much smaller, indicating that the terms in the average topic have a highly-skewed distribution; topics are very heterogeneous. The optimum number of nodes found is 900, the maximum checked.[13] We see the same ranking of the three models based on accuracy and F1, though ProdLDA curiously got less accurate. Coherence did not change much for either model. Though R^2 increased for LDA it did not change for ProdLDA.

Table 2: Parameters and evaluation metrics of the optimized experiment

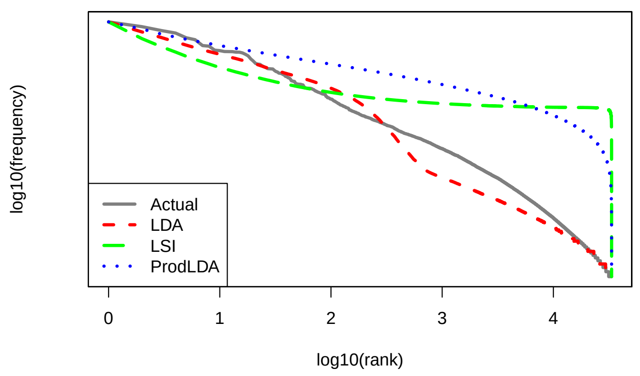
	k	alpha	Sum(beta)	Nodes	coherence	R-sq.	Accuracy	F1
LDA	317	0.14	233.3	-	0.18	0.88	0.87	0.87
LSI	233	-	-	-	0.80	-	0.65	0.71
ProdLDA	380	0.91	-	900	0.18	-0.01	0.28	0.27

[12]For an example of words that are positively correlated consider the words "the" and "this". Where you see "the" frequently, you will also find "this" (positive correlation). However, the relative frequency of the word "this" in documents containing "the" is the same (likely identical) to the relative frequency of "this" across the whole corpus (statistical independence).

[13]This indicates that I should increase the maximum number of nodes in the search space SigOpt uses.

Word frequencies may explain why LDA works well on the classification task, but not why ProdLDA works poorly. Figure 1, below, depicts Zipf's law (Zipf 1949) in the 20 Newsgroups corpus against each of the models. Zipf's law is an empirical law of language. Term frequencies follow a power law distribution against rank (e.g. 1 is the most frequent, 2 is second-most, etc.) and is log-linear.

Figure 1: Comparison of Zipf's law



The gray line in Figure 1 is the data—the "true" Zipf's law of the corpus. For the topic models, I sorted each $\boldsymbol{\phi}_k$ from largest to smallest value and averaged across topics to get a single line for each model. In the case of LSI, I subtracted the smallest value to avoid taking a log of negative numbers. LDA (red, dashed) clearly follows Zipf's law the closest. ProdLda and LSI clearly do not. This observation also explains why LDA fits the data better than ProdLDA as measured by R^2.

6 Conclusion

The big take away from this research is that LDA—in spite of being "old news"—is a solid method for vectorizing text. It fits the data well, performs on-par with ProdLDA in terms of coherence[14], and is the clear winner in the classification task.

6.1 Directions for Future Work

Large questions remain. Would these results hold against a broader set of topic models and embedding models? Would these results hold against a wider range of tasks? I would like to extend this project to include more topic models and popular embedding models like word2vec, GPT, and BERT. LDA can be a word embedding model too. Instead of a DTM input, one can feed it a term co-occurrence matrix (TCM). This TCM may be based on skip-grams as in word2vec. I would also like to extend this project to a wider range of tasks such as in Radford et al. (2019). I hypothesize—based on this research and some of my other work—that LDA would perform well.

A limiting factor for LDA is its speed. While the typical method for Gibbs sampling—as implemented in *textmineR*—is purely sequential, it need not be. The Gibbs algorithm may be run in parallel over documents or tokens without loss of convergence guarantees. And there is some work performing MCMC on GPUs. (Terenin et al., 2018) Taken together, this may bring LDA Gibbs sampling up to par with respect to speed. This project has given me additional motivation top upgrade *textmineR*'s Gibbs sampler to take advantage of parallelism.

[14]Srivastava and Sutton report ProdLDA significantly outperforms LDA in terms of coherence. I did not find the same. This could be due to several things: They used a more limited set of the 20 Newsgroups data than me. They report coherence based on NPMI whereas I use probabilistic coherence. Yet, the underlying mechanics are similar enough that differences between models should be the same. Also, they measured coherence in-sample while I report out-of-sample coherence, my in-sample results were no better.

7 References

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. Journal of machine Learning research, 3(Jan), 993-1022.

Chang, J., Gerrish, S., Wang, C., Boyd-Graber, J. L., & Blei, D. M. (2009). Reading tea leaves: How humans interpret topic models. In Advances in neural information processing systems (pp. 288-296).

Deerwester, S., et al, Improving Information Retrieval with Latent Semantic Indexing, Proceedings of the 51st Annual Meeting of the American Society for Information Science 25, 1988, pp. 36–40.

Griffiths, T. L., & Steyvers, M. (2004). Finding scientific topics. Proceedings of the National academy of Sciences, 101(suppl 1), 5228-5235.

Jones, T. (2015). A Coefficient of Determination for Topic Models. Available at: https://drive.google.com/file/d/0Bz2enPyUvnKIQmtDTEswbzdUMU0/view?usp=sharing

Lau, J. H., Newman, D., & Baldwin, T. (2014). Machine reading tea leaves: Automatically evaluating topic coherence and topic model quality. In Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (pp. 530-539).

Mitchell, Tom. (1996). 20 Newsgroups Data Set. Available from the University of California, Irvine Machine Learning Repository Web site: https://archive.ics.uci.edu/ml/datasets/Twenty+Newsgroups

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018) Improving language understanding by generative pre-training. Available at: https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf

Rosner, F., Hinneburg, A., Röder, M., Nettling, M., & Both, A. (2014). Evaluating topic coherence measures. arXiv preprint arXiv:1403.6397.

Srivastava, A., & Sutton, C. (2017). Autoencoding variational inference for topic models. arXiv preprint arXiv:1703.01488.

Terenin, A., Dong, S., & Draper, D. (2018) GPU-accelerated Gibbs sampling: a case study of the Horseshoe Probit model. Statistics and Computing.

Wang, Y., Bai, H., Stanton, M., Chen, W. Y., & Chang, E. Y. (2009). Plda: Parallel latent dirichlet allocation for large-scale applications. In International Conference on Algorithmic Applications in Management (pp. 301-314). Springer, Berlin, Heidelberg.

Zipf, G. K. (1949). Human behavior and the principle of least effort: An introduction to human ecology. Ravenio Books.