Topic Mining with LDA (and LARAM)

CS 410 Text Information Systems Course Project

December 2020
Authors: Dan Qian (dansiq2), Joe Everton (everton2), Ed Pureza (epureza2)
CMT ID: 85

Introduction

Our team originally selected the research paper titled “Latent Aspect Rating Analysis without Aspect Keyword Supervision” by Honging Wang, Yue Lu, and ChengXiang Zhai. After spending a considerable amount of time on attempting to understand the implementation details, we decided there were too many obstacles to continue recreating the research paper’s results. Taking into consideration Professor Zhai’s counsel, we decided to switch our topic to Latent Dirichlet Allocation. Our work is based on “Latent Dirichlet Allocation” by David Blei, Andrew Ng, and Michael Jordan [3].

Latent Aspect Rating Analysis without Aspect Keyword Supervision

When a review is submitted for a product, it can be assumed that the reviewer had different aspects of the product in mind and a priority on those aspects that led to a given overall rating. The objective of latent aspect rating analysis without aspect keyword supervision is to identify the following in a set of reviews:

- Main aspects that the reviewers considered
- Weights of the identified aspects in each review and overall
- Ratings given to each aspect identified

The generative model involved inferring the latent aspect assignments (i.e., probability of a word drawn from a topic) and the aspect weights for each document and estimating the following corpus-level parameters:

- Distribution of vocabulary words in each aspect
- Prior distribution of aspects for the whole corpus
- Sentiment polarity of each vocabulary word for each aspect
- Average weight of each aspect in the ratings
- Variance of weights for each aspect
- Variance of the ratings

Our main challenges included implementing the log-likelihood of each review that required an understanding of Jensen's inequality for convex functions and finding the maximums of the latent aspect assignments and prior distribution of aspects that involved implementing gradient-based optimization solutions.

After an office hour session with Professor Zhai, correspondence with Dr. Hongning Wang and evaluating the effort required to bring this type of project to completion, we decided to switch our focus to Latent Dirichlet Allocation, which we found the implementation to be more feasible given the timeline and amount of available information on the subject.

Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) is a generative probabilistic model for identifying topics in a collection of documents using Bayesian modeling. LDA draws the topic assignment of a word from a distribution of topics and the word from the word distribution of topics for a document. Unlike PLSA where the documents in the corpus define the probabilities, LDA can be used to generalize new documents by sampling from the distributions found by the model (i.e., the dataset used to generate the model serves as training data for new data). This is possible thanks to hyperparameters to Dirichlet distributions that generate the multinomial distributions, rather than basing them on training documents.

Objective

Our first objective was to have LDA find k topics from a corpus of documents. This would give us a sense at how well our implementation is doing at learning the topics from the corpus. The model will also be able to predict topic weights for new documents.

Our objectives also include applying document topic weights predicted by the model to a text classification task. One way of evaluating the utility of the model discussed in the paper is to see how well the topic models can be used to to train a document classifier, compared to classifying with a plain bag of words feature set based on the same documents. Using the topic distribution inferred by the model and labeled data, a classification model can be generated. The two classification applications covered in this paper include the identification of fake news and spam messages.

It is interesting to compare and contrast this approach of classification with LDA to Principal Component Analysis. We explore this relationship in the appendix.

Data

The original data set could not be found. We decided to use two other publicly available datasets from Kaggle.

Dataset	Source	# of Docs	# of Terms	Classes
Spam	https://www.kaggle.com/team-ai/spam-text-message-classification	5,157	8,741	Spam: 13% Not Spam: 87%
Fake News	https://www.kaggle.com/mohamadalhasan/a-fake-news-dataset-around-the-syrian-war	804	10,157	Fake: 47% Not Fake: 53%

Evaluation criteria

We will use Support Vector Machine (SVM) as our classification model for both LDA outputs and document terms. Using document terms to measure a baseline, the following results were obtained:

Dataset	Number of Features	Training Data Size	Precision	Recall	Average F1
Fake News	2,020	200	0.562	0.694	0.621

Spam	892	2,000	0.992	0.855	0.919

Our success criteria is to achieve better or equal performance metrics with LDA. Doing so would prove that similar or better classification accuracy can be achieved using significantly less features.

Tools

Python was the programming language for this project. The following packages were used:

numpy	re
scipy	random
math	sklearn
pandas	time

Methods

We attempted two different methods for creating the LDA models. The first method uses variational inference as explained in the original Latent Dirichlet Allocation paper [3] and further explained by Chase Geigle [4] to calculate the word assignment to a topic and the topic distribution for each document (posterior inference). The general steps are outlined in the figure below (from [3]).

(1) initialize $\phi_{ni}^0 := 1/k$ for all i and n
(2) initialize $\gamma_i := \alpha_i + N/k$ for all i
(3) **repeat**
(4) **for** $n = 1$ **to** N
(5) **for** $i = 1$ **to** k
(6) $\phi_{ni}^{t+1} := \beta_{iw_n} \exp(\Psi(\gamma_i^t))$
(7) normalize ϕ_n^{t+1} to sum to 1.
(8) $\gamma^{t+1} := \alpha + \sum_{n=1}^{N} \phi_n^{t+1}$
(9) **until** convergence

Figure 6: A variational inference algorithm for LDA.

The second method uses Collapsed Gibbs Sampling for the posterior inference which resembles a Markov-chain Monte Carlo method. The method is described in Geigle's paper and the implementation is detailed in Sterbak's' post [7].

Our implementation differed from the original LDA paper in the following ways:

1. We used random initialization for word topic assignments and document topic distributions
2. We borrowed the method implemented in gensim for updating alpha which claims [8] to use the Newton-Raphson method described in [3]. This method uses an additional

learning rate parameter ρ. In some cases this alpha updating code did not work well; to get around this, we used a simpler linear combination to calculate alpha.

3. We replaced the log likelihood estimation with a distance measurement for the document topic distribution.

The decision to deviate from the papers was based on our level of understanding, the level of complexity involved in original papers, and the amount of time allocated for the project.

Variational Inference

This method uses an EM algorithm similar to what is used in PLSA with the exception of the variables updated and parameters estimated. The corpus level parameters include the following priors:

- α: the parameter to the dirichlet distribution which forms the topic multinomial distribution in the training data
- β: the parameter to the dirichlet distribution which forms the word multinomial distribution for each topic

These parameters are estimated in the M step.

The following document level variables are inferred the E step:

- Φ: word topic assignment
- □: topic assignment

Gibbs Sampling

The difference in this method is the use of random sampling given the conditional distributions calculated in iterative steps. The word topic assignment is the main parameter updated with each iteration.

```
for iteration in range(iterations):
    for d, doc in enumerate(docs):
        for n, w in enumerate(doc):
            topic = z[d][n]

            # Remove the topic assignment for the word
            theta[d][topic] -= 1

            # Recalculate the topic
            p_topic_given_doc = normalize(theta[d]) + self.alpha
            p_word_given_topic = (self.phi[:,w] + self.beta) / (self.topic_sampling_count + self.vocabulary_size * self.beta + 0.0000001)
            p_topic = normalize(p_topic_given_doc * p_word_given_topic)
            new_topic = np.random.multinomial(1, p_topic).argmax()

            # Assign the new topic to the word
            z[d][n] = new_topic
            theta[d][new_topic] += 1
```

Snippet from [lda_var_inf_without_smoothing.py]

Experiment Results

For our experiments, we set the number of topics (k) to 15. The max number of EM iterations was set to 10 as well as the maximum number of E step iterations.

Precision, Recall, and F1 Scores

Feature set	Data set	Training Set Size	Vocabulary Size	Num Topics	Precision	Recall	F1
Baseline (bag of words)	**Spam**	**2000**	**892**	**N/A**	**0.992**	**0.855**	**0.919**
Variational Inference	Spam	2000	892	5	0.889	0.526	0.661
Variational Inference	Spam	2000	892	10	N/A	0	N/A
Variational Inference	Spam	2000	892	15	N/A	0	N/A
Gibbs Sampling Training + Variational Inference	Spam	2000	892	5	0.899	0.763	0.826
Gibbs Sampling Training + Variational Inference	Spam	2000	892	10	0.912	0.822	0.865
Gibbs Sampling Training + Variational Inference	Spam	2000	892	15	0.930	0.789	0.854
Gibbs Sampling	Spam	2000	892	5	0.886	0.770	0.824
Gibbs Sampling	Spam	2000	892	10	0.919	0.895	0.907
Gibbs Sampling	Spam	2000	892	15	0.941	0.842	0.889

Feature set	Data set	Training Set Size	Vocabulary Size	Num Topics	Precision	Recall	F1
Baseline (bag of words)	**Fake News**	**200**	**2020**	**N/A**	**0.562**	**0.694**	**0.621**
Variational Inference	Fake News	200	2020	5	0.557	0.874	0.681
Variational Inference	Fake News	200	2020	10	0.605	0.802	0.690
Variational Inference	Fake News	200	2020	15	0.570	0.694	0.626
Gibbs Sampling Training + Variational Inference	Fake News	200	2020	5	0.555	0.640	0.594
Gibbs Sampling Training + Variational Inference	Fake News	200	2020	10	0.576	0.784	0.664
Gibbs Sampling Training + Variational Inference	Fake News	200	2020	15	0.619	0.586	0.602
Gibbs Sampling	Fake News	200	2020	5	0.563	0.721	0.632
Gibbs Sampling	Fake News	200	2020	10	0.570	0.766	0.650
Gibbs Sampling	Fake News	200	2020	15	0.587	0.550	0.567

Our setup for evaluation is different from the original paper in two ways:

- Different datasets were used as shown in the figure below (extracted from [3])
- Size of dataset and proportion of data used for training

Blei, Ng, and Jordan's paper produced good accuracy results with SVM classification as described above using topic weights as parameters. Their classifier outperformed the complete bag-of-words SVM classifier most of the time. These plots use proportion of training data as

their x axis, which we did not analyze, but they give some good references for relative accuracy between bag-of-words SVM and LDA-SVM.

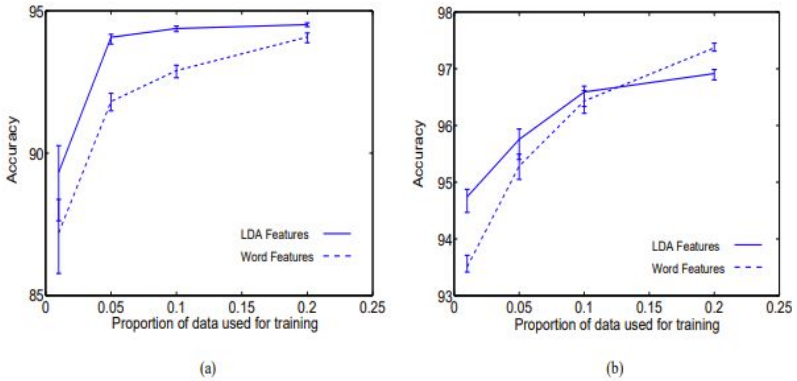

Figure 10: Classification results on two binary classification problems from the Reuters-21578 dataset for different proportions of training data. Graph (a) is EARN vs. NOT EARN. Graph (b) is GRAIN vs. NOT GRAIN.

For our reproduction of their paper, we used two other data sets because we could not find the one cited in the paper. Our results are not directly comparable to Blei et al's. However, our LDA/SVM classifier did outperform the bag-of-words/SVM classifier baseline on the fake news dataset we used, which approaches Blei's findings. With more tuning, we may nudge our results even closer. Our results are in the table above. Improvements are in green. These results show that we have likely reproduced the paper reasonably well.

We note that for the Spam/Ham dataset, the results are consistently worse than the baseline. This indicates that this LDA/SVM classifier does not perform as well on shorter documents with smaller vocabularies. The topics found are probably less useful for classification. This is not inconsistent with Blei's findings; their dataset was based on newspaper articles, and probably closer to the fake news dataset we used that had better results.

Conclusion

1. Aspect weights outperform the baseline (bags of words) as features for text classification, when the documents are long and the vocabulary is large (news articles in our experiments).
2. Aspect weights underperform the baseline (bags of words) as features for text classification when the documents are short and the vocabulary is small (text messages in our experiments).
3. Using more aspects does not necessarily improve the quality of the aspect weights as features.

Successes

We successfully implemented Latent Dirichlet Allocation, following as much as possible Blei, Ng, and Jordan's paper [3], and using two different approaches: collapsed Gibbs sampling, and variational inference. We drew conclusions based on experiments with each approach across two data sets. Interestingly, cutting the features down from thousands (bag of words) to around 10 (topics) gave comparable, and for longer documents, better results.

Opportunities

If we had more time, it would be nice to try the following:

- Complete the implementation of LARAM, our original intended paper. This paper required more understanding of variational inference, and left the exercise of calculating a complicated log likelihood and some parts of the E and M steps to the reader. With more time, we might figure these details out.
- Complete or borrow the log-likelihood calculation for LDA for proper termination of the EM algorithm.
- Compare our results against library implementations of LDA, including Gensim and NLTK.

References

1. H. Wang, Y. Lu, C Zhai. Latent aspect rating analysis without aspect keyword supervision https://dl-acm-org.proxy2.library.illinois.edu/doi/10.1145/2020408.2020505

2. S. Kapadia. Evaluate Topic Models: Latent Dirichlet Allocation (LDA). A step-by-step guide to building interpretable topic models https://towardsdatascience.com/evaluate-topic-model-in-python-latent-dirichlet-allocation-lda-7d57484bb5d0

3. D. Blei, A. Ng, M. Jordan. Latent Dirichlet Allocation https://dl.acm.org/doi/pdf/10.5555/944919.944937

4. C. Geigle. Inference Methods for Latent Dirichlet Allocation. http://times.cs.uiuc.edu/course/510f18/notes/lda-survey.pdf

5. Various. StackExchange https://stats.stackexchange.com/questions/126268/how-do-you-estimate-alpha-parameter-of-a-latent-dirichlet-allocation-model

6. Various. Github: RaRe Technologies - gensim. https://github.com/RaRe-Technologies/gensim/blob/6c80294ad8df16a878cb6df586c797184b39564a/gensim/models/ldamodel.py#L434

7. T. Sterbak. Latent Dirichlet Allocation from Scratch https://www.depends-on-the-definition.com/lda-from-scratch/

8. How do you estimate α parameter of a latent dirichlet allocation model? https://stats.stackexchange.com/questions/126268/how-do-you-estimate-alpha-parameter-of-a-latent-dirichlet-allocation-model

Appendix

LDA VS PCA

It is interesting how LDA can be viewed in this setting as a substitute for Principal Component Analysis (PCA). PCA is a more general technique that can be used to reduce the dimensionality of data. A bag of words implementation may work with some classification algorithms and some data sets, but the matrices can become prohibitively large, sometimes making it difficult to train a model. PCA can help by reducing the features from |V| to some smaller k, without losing very much of the variation of the data, or its prediction power, even with surprisingly low values of k. This can work fairly well even when k is 5% or less of |V|.

The downside of PCA is that the compressed feature set is not as interpretable. The k new features will usually map to several words that may not be related, or form any topic meaningful to humans. In spite of this, the compressed features are often still useful for prediction.

Compared to using LDA as outlined above to generate topic distributions of documents, and then classify those distributions rather than the entire bag of words, LDA is also reducing the dimensionality from |V| to k, but it is designed to result in k *topic* distributions that should be more explainable in terms of topics than the K dimensions PCA created. To put this in a concrete example, a spam classifier might take as inputs, the weights of 10 topics in each new document, as predicted by the LDA model. These 10 weights could then be used as features in the spam classifier. PCA might do just as well at this compression of features, but its features will not be explainable as topics.