Topic Modeling and Language Modeling for Gutenberg Project

Tan Nguyen
Department of Computer Science
Johns Hopkins University
Baltimore, MD 21218
tnguy188@jhu.edu

Abstract

Topic modeling and language modeling have been the center of natural language processing. Both abstractions heavily use a variation of graphical models to represent their inherent architectures. Here, we analyze these structures, specifically Latent Dirichlet Allocation and Correlation Explanation for topic modeling and Restricted Boltzmann Machine and Log Bilinear Language Model for language modeling. To analyze and test these models, we will use parts of Gutenberg text corpus dataset.

1 Introduction

In recent years, seeing the potential of machine learning, applications in natural language processing (NLP) have grabbed the attention of both the public and researchers. Many models have been proposed to solve tasks that seem only possible to human in the past. Here, we look at two specific tasks, topic modeling and language modeling. Topic modeling provides the abstract summary of topics given the set of documents while language modeling provides a model that can predict the next word given the preceding texts. Since both tasks can only be achieved with probabilistic result, we use many variation of graphical models. For our implementation, we will use Python and its corresponding packages like nltk, gensim... For our dataset, we will use parts of Gutenberg dataset, which has more than 3,000 books in English.

2 Topic Modeling

While books and documents are consisted of a lot of words, the actual contents lie within only. Topic modeling aims to "summarize" a set of documents into a much smaller abstract set of topics, which can hopefully explain the context of these documents. Here, we consider two specific models, Latent Dirichlet Allocation[2] , a graphical model using a generative process, and Correlation Explanation[3], a discriminative model.

2.1 Preprocessing and Evaluation Measures

Preprocessing: Topic modeling does not need to consider all words as well as their orders. This relaxation allows for a more aggressive preprocessing step where we can remove "stop" words (words added for grammatical purposes) and very short words (less than 3 characters). Since topics are also not needed to be grammatically correct, topic modeling allows lemmatization and stemming of words, which allows us to consider only the word's root. This preprocessing step also allows the model to scale down significantly. In most documents, we only need to consider $\sim 10\%$ of the words in the documents as "important".

Evaluation: The state-of-the-art to evaluate whether the topic represents the set of documents well is topic coherence [4], which test the quality of learned topic. Words within each topic are scored pairwise and then coherence score is computed by averaging over all the scores of all topics.

There are two standard measures, UMass[5], representing intrinsic measures, and UCI [4], representing extrinsic measures. In particular, UMass Score is defined as

$$Score_{UMass}(w_i, w_j) = \log \frac{D(w_i, w_j) + 1}{D(w_i)}$$

where $D(w_i, w_j)$ is the number of documents both w_i and w_j appear together. Therefore, the score is higher if two words appear together a lot relatively to when each appears alone.

UCI Score is defined as

$$Score_{UCI}(w_i, w_j) = \log \frac{p(w_i, w_j)}{p(w_i)p(w_j)}$$

This is similar measurement with UMass Score where it shows the likelihood of two words appear together in the same document over when it appears in a random document.

2.2 Latent Dirichlet Allocation

For topic modeling, Latent Dirichlet Allocation (LDA) [2] has been considered "the state of the art". LDA is a generative unsupervised probabilistic model that views texts as "bags of words", or a collection of words where the order within the text is not important. LDA views each document as a set of topics, which consists of words. LDA accepts a matrix of document-word as input and learn the set of topics through a generative process.

2.2.1 Method

LDA's plate notation is described in Figure 1.

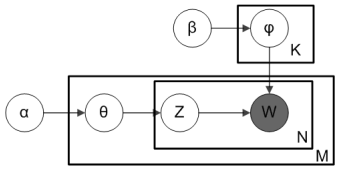

Figure 1: LDA Plate Notation

Since w is the only observable variable, we can use a generative process to find the original distribution of all the latent variables. We present the LDA algorithm in Algorithm 1.

Algorithm 1 LDA Generative Process

1: **for** each document with index i in M **do**
2: Choose $\theta_i \sim Dir(\alpha)$
3: **for** each topic with index k in K **do**
4: Choose $\phi_k \sim Dir(\beta)$
5: **for** each word in w at position (i, j) **do**
6: Choose a topic $z_{ij} \sim Multinomial(\theta_i)$
7: Choose a word $w_{ij} \sim Multinomial(\phi_{z_{ij}})$

As shown through the algorithm, this process will not scale well when the models train with high number of documents and words. Therefore, in order to speed up such process, many choose to use Gibbs sampling rather than variation Bayes approximation to take advantage of the natural sparsity of the input matrix (since not every word will appear in every document, thus the majority of the matrix will be 0).

For testing purpose, we will use the Python package *gensim*'s implementation of LDA. For tuning parameters, we build topic models for Nathaniel Hawthorne, who has 86 books in Gutenberg project.

2.2.2 Tuning Parameters

Dictionary Filter: Not all words after preprocessing steps are equal, so we should consider "extreme" filtering of the dictionary. We consider the minimum number of documents a word should be in to be considered as "topic" that represents the whole set of documents. We show this in Figure 2. We can see that the UCI coherence score peaked at 3 with the second max around 16. Though the model seems more "coherent", filtering too little at 3 will leave these topics generic, applicable to other set of authors, rather than just Nathaniel Hawthorne. Therefore, it is best to use a larger number, i.e. 16.

Number of Topics: Number of topics, K, is the only parameters that needs to be defined. For each set of documents, we will need a different value of K. We choose K at the end of a sharp increase of topic coherence as it offers the most meaningful topics. However, picking too high value K will allow more "sub-topics" or repeated words over many set of topics. We show this in Figure 2 where the peak is at 8 and 17. Since our set of documents have a relatively small dictionary ($\sim$ 3400 words), choosing 8 topics will be better fit to our purpose.

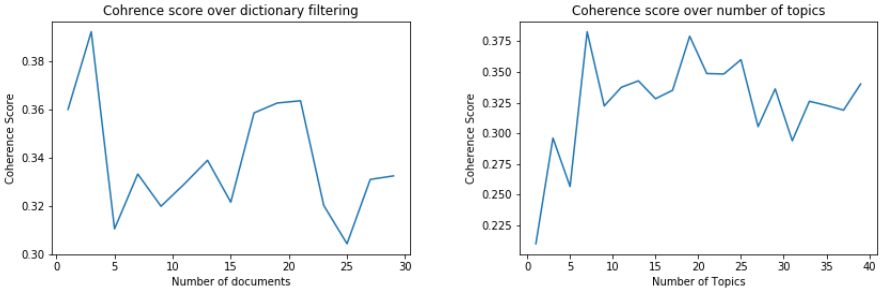

Figure 2: Dictionary Filter and Number of Topics

Term Frequency-Inverse Document Frequency (TF-IDF): We can fine tune LDA by using a different interpretation of document-word matrix. TF-IDF [6] builds the matrix with weighting factor reflecting the frequency that words appear in each document and the number of documents that words appear. However, TF-IDF does not perform well in LDA, which shows in Figure 3. In particular, TF-IDF fails to get the coherence score to be higher than 0.32, which is the lower bound of the original implementation. LDA uses "bag-of-words" representation, which does not consider the word orders; therefore, adding weights to word frequency will not help, if not worsen, the performance.

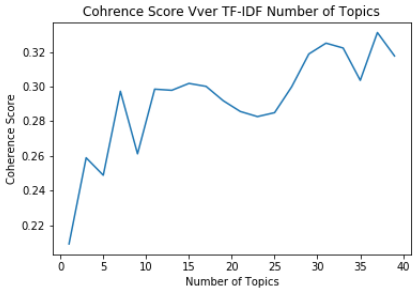

Figure 3: TF IDF Coherence Score

2.3 Correlation Explanation

While LDA has yielded great results, many cast doubts on LDA's generative process for assuming the underlying generative model. Correlation Explanation (CoRex) [3] introduces a different approach by using a disciminative process to learn the "maximally informative topics through information-theoretic framework". Similar to LDA, CoRex accepts a document-word matrix and considers topics as latent variables. It seeks to explain the dependencies between words in documents and latent topics through discriminative process.

2.3.1 Method

We consider X_G as a group of word and Y as a set of topics. We have entropy of X as $H(X)$. CoRex uses total correlation, defined as multivariate mutual information, as: $TC(X_G) = \sum_{i \in G} H(X_i) - H(X_G)$. Total correlation describes correlation as the measure of total dependence. Since we need to condition on the latent variable Y, we also have: $TC(X_G|Y) = \sum_{i \in G} H(X_i|Y) - H(X_G|Y)$. Therefore, to maximally explain the dependencies between words in documents and latent variables, we want to maximize

$$TC(X_G; Y) = TC(X_G) - TC(X_G|Y) = \sum_{i \in G} H(X_i) - H(X_G) + H(X_i|Y) - H(X_G|Y)$$

For the purpose of fine tuning, we will use Python package *corextopic*'s implementation of CoRex.

2.3.2 Tuning Parameters

Number of Topics: Similar to LDA, CoRex needs human input for the number of latent variables. As each topic explains only a portion of the text, we want to maximize the total correlation and/or coherence score until the additional topic yields diminishing return. As shown in Figure 4, since our vocabulary is quite small in CoRex, we see a lot of fluctuation, but the peak is around 13 and 17. At 17 topics, the value is approaching 0; therefore, the training may have been overfitting, thus not representing the whole set well.

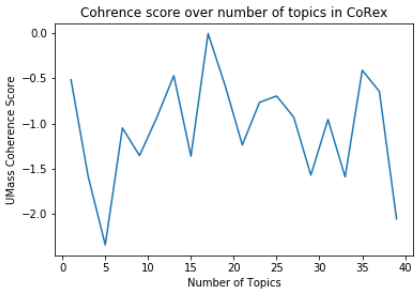

Figure 4: Coherence Score for CoRex

Initialization: Similar to other discriminative methods, CoRex depends on the weight initialization to find the best total correlation. Some initialization can only reach low local maximization, while others come closer to the global maximization. From experiences, any fixed initialization will yield non-ideal result; therefore, we use random initialization for our CoRex testing.

2.4 Experiments and Results

2.4.1 Dataset

Given the Gutenberg dataset provided, we want to find the topics/themes of some famous authors through their books. Therefore, we will run different topic modeling models on these set of books and see the result. We will consider three famous authors: Nathaniel Hawthorne, Sir Arthur Conan Doyle, and Mark Twain.

2.4.2 Results

Numeric Result: We show the numeric results of both models in Table 1. For LDA, we consider both intrinsic and extrinsic measures. We see that there are correlation between UMass and UCI score measurement since both methods use similar pairwise score functions. Therefore, we can use either measurement to check for the coherence of our models.

For the second comparison, we want to see the scores of LDA and CoRex. We can see that for all authors, LDA outperforms CoRex with great margin. This shows that the underlying generative model that LDA uses represents a good estimation, while discriminative process of CoRex does not perform as well. This is also due to the size of dataset, where the number of words considered are still

relatively small, thus discriminative process will get worse result due to overfitting, while generative process provides for better result as it depends more on the distribution rather than the values within the input.

Different Coherence Score for LDA		
Author	UMass	UCI
Nathaniel Hawthorne	-0.4998	0.3229
Sir Arthur Conan Doyle	-0.7064	0.3319
Mark Twain	-0.4419	0.3155

UMass Coherence Scores Between Models		
Author	LDA	CoRex
Nathaniel Hawthorne	-0.4998	-2.1909
Sir Arthur Conan Doyle	-0.7064	-2.3887
Mark Twain	-0.4419	-1.8865

Table 1: Coherence Score Results

Literary Result: We want to provide some literary results to see how well these models perform in relative to human interpretation. In the interest of literary interpretation, we provide what critics and historians talk about these authors. Nathaniel Hawthorne is known for his anti-Puritan inspirations and dark Romanticism, with themes centering around humanity sin and evil, in the context of New England. Sir Arthur Conan Doyle is known for his Sherlock Holmes works, with the theme centering around crime fiction, fantasy, and science fiction. Mark Twain is known for his humorist and satire style of writing where he discussed a lot about civil rights, religion, and anti-imperalism.

From the literary standpoint described, we can see from Table 2 that LDA performs better than CoRex with words more specific to each author. This also shows that the coherence score seems to match well with human interpretation of the words, showing a strong testing measures. For both cases, words in each topic also describe a larger context. LDA also contains more nouns than verbs or adjectives, which the topics should usually be, thus show LDA understands the context of the text better. The main downside that LDA has is the words within topics overlapped, which shows that each topic does not clearly separate the texts nor describe each portion of documents well. CoRex shows the separation more clearly while suffering the cost that the topic becomes more vague.

Author	Rank	LDA	CoRex
Nathaniel Hawthorne	1	english, tower, architecture, sculptor, statue, american, palace, artist, castle, cathedral	alarm, committed, readily, contrary, prepared, directly, fought, assistance, gleaming, skill
Nathaniel Hawthorne	5	grandfather, governor, boston, english, william, soldier, massachusett, captain, british, province	stars, headless, outspread, displaying, deeds, despair, monument, leap, summits, briskly
Sir Arthur Conan Doyle	1	holmes, watson, smith, sherlock, inspector, robert, baker, client, moore, lawn	betwixt, destination, crashed, curls, studded, blackened, wars, assembled, fires, hardened
Sir Arthur Conan Doyle	4	emperor, infantries, cavalries, colonies, regiment, brigade, artillery, battering, military, robert	metallic, largest, pencil, flashed, machine, horrified, ingenuity, trembling, informed, sympathy
Mark Twain	1	clemency, hartford, boston, juli, roger, pilot, francisco, type, magazine, mississippi	brand, deposit, clergyman, repair, spar, display, prayer, twilight, uplift, righteous
Mark Twain	8	nigger, raft, hain, aunt, bout, shove, salli, steamboat, chamber, bust	linen, shelter, candlestick, callow, hire, mild, francisco, whisper, shirt, obtain

Table 2: Literary Result for both models

2.4.3 Analysis

While these two graphical models have worked great for topic modeling, both models require a really large overhead of preprocessing. Besides the fact that both models need to go through all the texts to filter out important words, both need to convert set of documents to document-word matrix form which is O(MN) running time (with M be the number of words per document and N be the number of documents). The matrix turns out to be sparse in general, thus easier to work with in other steps.

Both models also require human input, number of topics, to run. This poses as a challenge especially if we want to run topic modeling on multiple dataset as this requires fine-tuning specifically for each one. In addition, the difference between generative and discriminative process leads to the difference in running time where CoRex tends to take 3 to 5 times longer. This is because CoRex needs to go through the complete array for each step as it maximizes the total correlation.

3 Language Modeling

Human languages, unlike programming language, are quite fluid and evolving over time. However, computers needs rules to learn efficiently, thus many applications of NLP like speech recognition or machine translation require statistical language modeling, where the model predicts the next word given preceding texts. Graphical models become a great fit for such models, since they help answer these conditional probability questions with ease.

Here, we consider two graphical models proposed by (Hinton et al., 2007), Restricted Boltzmann Machines [7] and Log Bilinear Language Model[7].

3.1 Preprocessing and Evaluation Measure

Preprocessing: Unlike topic modeling, each word and its order matter in language modeling as the model learns how to predict the next word. Therefore, for preprocessing, we can only split the sentences within each document and tokenize each word in the sentence to compute a very sparse document-word matrix.

Evaluation Measure: To evaluate whether our language model performs well, we use perplexity score, which is the "state-of-the-art". Perplexity measures how likely a language model will predict the test data. Perplexity is defined as:

$$PP(p) = 2^{H(p)}$$

where $H(p)$ is defined as the entropy of the distribution. Therefore, by definition, language model is preferred to have lower entropy as want the predicted result to do better than randomly guessing. For example, given a vocabulary of of n words, we would hope that the number of words, m, that our model needs to consider as the next word to be less than n. m is the result of our perplexity score.

3.2 Restricted Boltzmann Model

In order to design a probabilistic model to predict the next words, the model needs to learn the dependencies between words in a sequence. Restricted Boltzmann Machines (RBM) [7] achieved this by assuming a layer of stochastic hidden variables and try to learn the weights between the two layers for maximum likelihood learning. Though maximal likelihood learning is intractable, RBMs can be trained efficiently with contrastive divergence, an extension of MCMC sampling methods to approximate the gradient efficiently.

3.2.1 Method

First, we need to define the energy function for joint configuration of visible nodes and hidden units. We represent each word as feature vectors, which are distributed representations for words, to avoid a really large vocabulary size and allow less nodes to represent words in the network. So, energy function is defined as:

$$E(w_n, h; w_{1:n-1} = -(\sum_{i=1}^{n} v_i^T R W_i)h - b_h^T h - b_r^T R^T v_n - b^T v_n$$

where R is word-feature matrix, v is feature vector, w is word, and b is the bias term. From this energy function, we can deduce the conditional distribution of the next word given a text to be:

$$P(w_n|w_{1:n-1}) = \frac{1}{Z_c}\exp(-E(w_n, h; w_{1:n-1}))$$

where Z_c is the context-dependent normalization term. For our learning task, we need to maximize the log-likelihood for the whole dataset or $L(D) = \sum \log P(w_n|w_{1:n-1})$.

3.2.2 Tuning Parameters

Epochs: We need to run RBM over multiple epochs until it reaches the optimal point. In particular, when running RBM over multiple epochs, we get the result as shown in Figure 5. The L1 loss function converges very fast to 0 after just couple of epochs given our learning rate. This is partly due to the small dataset that we considered. Therefore, the learning process is quite efficient but susceptible to overfit very easily.

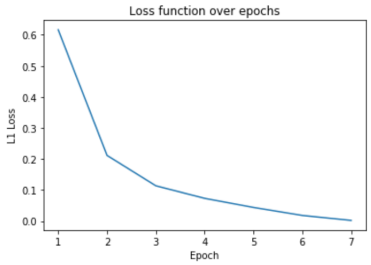

Figure 5: Running RBMs with Multiple Epochs

Number of Hidden Units: For RBM, we also need to specify the number of hidden units. For many testings, surprisingly, the best number of hidden units is the number of visible nodes in the network. This one-to-one correspondence seems to work well because our network is relatively small ($\sim$ 1000 nodes). Therefore, this parameter needs more extensive fine-tuning for larger network.

3.3 Log Bilinear Language Model

While RBMs work relatively well, the introduction of stochastic hidden variables adds a lot of complexity. Log Bilinear Language Model (LBL) alternatively avoids hidden variables completely and directly parameterize the distribution from the energy function.

3.3.1 Method

Similar to RBM's energy function but without the hidden variables, we define the energy function as:

$$E(w_n; w_{1:n-1}) = -(\sum_{i=1}^{n-1} v_i^T R C_i) R^T v_n - b_r^T R^T v_n - b_v^T v_n$$

where C_i explains the interaction between the feature vector of w_i and that of w_n. From this energy function, we get the predictive distribution as:

$$P(w_n|w_{1:n-1}) = \frac{1}{Z_c}\exp(-E(w_n; w_{1:n-1}))$$

where Z_c is the normalizing term.

3.3.2 Tuning Parameters

End of Sentence Tag: Since the model accepts sentence-based input, many adjust the input with "end of sentence" tag to avoid overfitting by learning the next word from the following sentences. Using both methods for Gutenberg dataset, we find the difference in perplexity score to be less than 0.5%. Therefore, we tend not to consider this as a fine-tuning parameter for our models. There is a possibility that larger dataset can gain benefit from adding this end of sentence tag.

Learning Rate: We can also adjust the learning rate of the model. By default, the learning rate is 0.001. When we run our model with the larger learning rate, the perplexity score increases minutely, until around 0.05, where the model completely diverges and perplexity score goes to infinity. With a smaller learning rate, the perplexity score decreases as it has not reach the optimal point. With a larger dataset, the manipulation of learning rate will affect the model's performance more.

3.4 Experiments and Results

3.4.1 Dataset

Given the Gutenberg dataset, we want to build language models where it can predict the next words given a context. To achieve this, we train the model on couple of books from an author and then test with another book from the same author. Here, we consider three famous authors: Nathaniel Hawthorne (4 documents), Mark Twain (1 document), and Abraham Lincoln (3 documents).

3.4.2 Results

From the comparison described in Table 3, we can see that RBM tends to have a slightly higher result from LBL. However, since our training dataset is small, the perplexity score for the test data is relatively bad. For example, learning Abraham Lincoln's writing of around 659 words cannot help the model much to predict the next word in the test. Therefore, it makes sense that RBM and LBL will need around 620 words to predict the following words.

From our model, we can also see that the stochastic hidden variables tend to complicate our models, and in our case, the data will underfit such model, yielding the result to be worse than we hope for.

Authors	Vocab. Size	RBM	LBL
Nathaniel Hawthorne	447	401.29	396.77
Mark Twain	900	880.54	875.91
Abraham Lincoln	659	623.42	617.36

Table 3: Perplexity Score of RBM and LBL

3.4.3 Analysis

Both models run very slowly as each word matters and modeling a very large text will cost a lot of memory and processing power. For a larger set of texts, it may take hours. For RBM implementation, we optimize with PyTorch implementation of Tensor and for the LBL implementation, we optimize with vectorization; however, these tools still did not provide the speed that we have hoped for.

While training for both are slow, it is clear that LBL takes less time and memory while maintaining a good enough perplexity score. This is due to the simplicity of the model where it completely avoids the hidden layer and its normalizing term much less computationally intensive.

4 Conclusion

Using Gutenberg's text corpus, we have evaluated various graphical models performing topic modeling, providing a set of topics given a set of texts, and language modeling, predicting the next word given preceding texts. In topic modeling, we evaluate Latent Dirichlet Allocation and Correlation Explanation where given a set of books from three famous authors, LDA outperforms CoRex based on topic coherence score. In language modeling, we evaluate Restricted Boltzmann Model and Log Bilinear Language Model, where given a set of books as training data, LBL has slightly better results than RBM based on perplexity score.

While the results are promising, there are many extensions we hope to make. For topic modeling, we want to implement the semi-supervised versions, Guided LDA and Anchored CoRex [3]. For language modeling, we want to consider other graphical models like n-gram [8] and temporal RBM [7]. Regarding the Gutenberg, we hope to learn more from other authors or a group of authors for topic modeling and a larger set of books from the same author for language modeling. We hope that these extensions can show us how tuning different parameters will affect more significantly the result of each model.

References

[1] Lahiri, Shibamouli. "Complexity of word collocation networks: A preliminary structural analysis." arXiv preprint arXiv:1310.5111 (2013).

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

[3] Gallagher, Ryan J., et al. "Anchored correlation explanation: Topic modeling with minimal domain knowledge." Transactions of the Association for Computational Linguistics 5 (2017): 529-542.

[4] Mimno, David, et al. "Optimizing semantic coherence in topic models." Proceedings of the conference on empirical methods in natural language processing. Association for Computational Linguistics, 2011.

[5] Topic Model Diagnostics, mallet.cs.umass.edu/diagnostics.php (2018).

[6] Aizawa, Akiko. "An information-theoretic perspective of tf–idf measures." Information Processing & Management 39.1 (2003): 45-65.

[7] Mnih, Andriy, and Geoffrey Hinton. "Three new graphical models for statistical language modelling." Proceedings of the 24th international conference on Machine learning. ACM, 2007.

[8] Brown, Peter F., et al. "Class-based n-gram models of natural language." Computational linguistics 18.4 (1992): 467-479.

[9] Wenjieguan. "Wenjieguan/Log-Bilinear-Language-Models." GitHub, 25 July 2014, github.com/wenjieguan/Log-bilinear-language-models.