DUKE UNIVERSITY

INDEPENDENT STUDY FINAL REPORT

Lin Xiao

Contents

1 Introduction

Overview

Scheduling optimization is concerned with the optimal allocation of events to time slots. In this paper, we look at one particular example of scheduling problems - 2015 Joint Statistical Meeting. We want to assign each session which is a mix of topics to time slots to reduce scheduling conflicts. First we will talk about the motivation for this example as well as the constraints and the optimality criterion, then propose using Latent Dirichlet Allocation (LDA) to identify the topic proportion of each session and talk about the fitting and the extensions of the model. Later we will introduce a Greedy Algorithm to minimize conflicts.

2 Data Scraping

The data is downloaded on JSM website session search. Each abstract is associated with a unique abstract key number, a non-unique session number, author name, title, and datestime of that session. In 2015, JSM has more than 3000 abstracts and 700 sessions(each session has 1 or more talks). We scraped the data with python's built-in method of "request" and "HTML". We created some function in Python to remove unnecessary characters, such as backslash n, backslash r and backslash t.

3 Data Scraping and Processing

Each step we create new "abstract" column.

3.1 Stop words

Remove all stop words. David Blei has a good list of stop words; it is part of David Blei's python implementation of turbotopics on his website and is the last download on this page of links: `http://www.cs.columbia.edu/~blei/topicmodeling_software.html`.) Word_tokenize from **nltk** package is used.

3.2 Stemming

Stem the words so that sing/sang/sung/song are all mapped to the same token. SnowballStemmer from **nltk**.stem.snowball is used.

3.3 Tokenization

We remove tokens that only appear once in all abstracts, and tokens only appear once in all abstracts for each session. Word_tokenize from **nltk** is used here.

3.4 N gram

Create N grams. In this case we create bi-grams and tri-grams, so that two words and three words could be mapped to one token. We iterate through the entire corpus and add each pair of bigrams/trigrams to the list.

We found out there are still some n-grams that are not helpful in identifying the topics in the corpus, and we proposed using some books that are almost completely irrelevant to statistics to remove these unhelpful n-grams. They are downloaded from Project *Gutenberg*. The first one is called *LittleBusybodies* by Jeanette Augustus Marks and Julia Moodyand. It is about learning the life of crickets, bees, beetles and other busy bodies for pre-school children. The second one is called *HereinisLove* by Reuel L. Howe. It is about a study of the biblical doctrine of love in its bearing on personality, parenthood, teaching, and all other human relationships.

Formerly we formed a list of words from these books, joined all the abstract in to one big word list, took out ones appeared more than 4 times, and eliminated the bigrams/trigrams that are in the book word list. This leaves us with a list of eligible bigrams and trigrams. If the bigrams and trigrams in each abstract are also in that list, we add them into the abstracts. However, the chosen 4 times seems ad hoc. We can further investigate how to select the threshold, but here I propose an alternative way for choosing bigrams and trigrams:

1. Bigramming the book word list and leave out bigrams only appear once, create book word bigram list(same for trigram).
2. For each abstract:
   (a) Remove punctuations and other non words. Remove bigrams that are also in book word bigram list(do the same step for trigrams).
   (b) Remove once bigrams and trigrams [ad hoc].
   (c) Compare the frequencies of bigrams and trigrams, if a bigram appears the same number of times as a similar trigram, we keep the trigram and leave out the bigram; if the bigram appears less than the similar trigram, we do the opposite. For "similar" it means whether the bigram is a subset of trigram. For example, "time seri" is similar to "Bayesian time seri". If "time seri" shows up n times while "Bayesian time seri" also shows up n times, we keep 'Bayesian time seri" cause "Bayesian" is the key word here. If "time seri" shows up n times, 'Bayesian time seri" appears [n/2] times and 'Frequent time seri" appears n-[n/2] times, we keep "time seri" and leave out the other two.
   (d) Join bigrams and trigrams back to each abstract.

This is one idea I have, trying to deal with the case where we are not sure whether a topic or its sub topics are more representative. An improvement can be added to 2(b) is to compare conditional probabilities of bigrams/trigrams with unigrams to decide if we should keep these once bigrams/trigrams.

4 Latent Dirichlet Allocation

4.1 LDA

LDA is a generative model with the assumption that the number of unknown latent topics is fixed[1] and the process could be visually presented as:

- For each topic k:
  - Draw a vector of word proportions $\phi_k \sim Dir_V(\alpha)$. This determines the relative weights of each team in topic k.

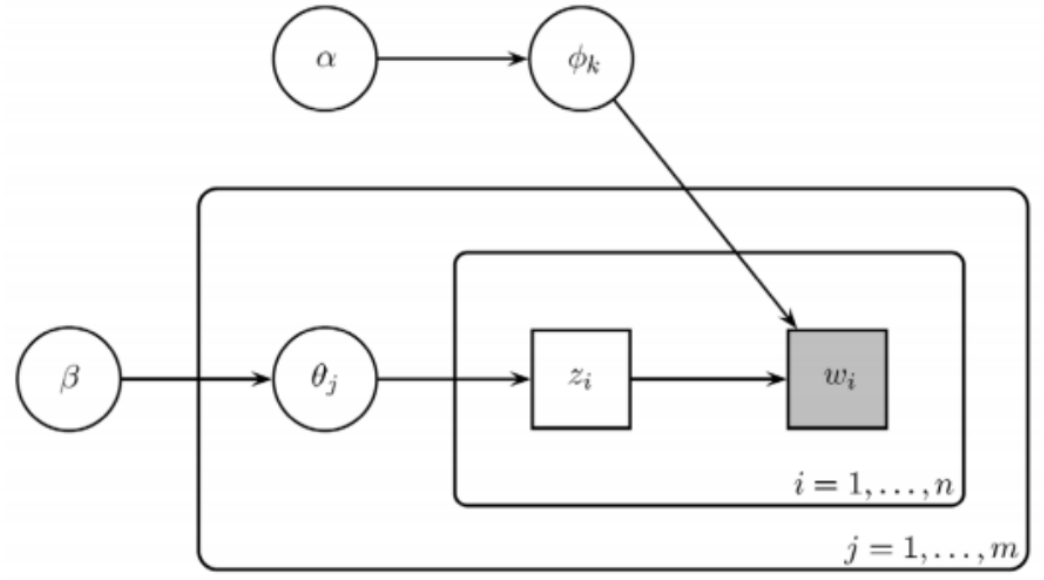

Figure 1: Latent Dirichlet Allocation

- For each document D_j:
  - Draw a vector of topic proportions $\theta_j \sim Dir_j(\beta)$. This determines the extent to which document D_j is composed of each of the K topics.
  - For each word w_i in document j: Draw a topic assignment $z_i \sim$ Mult(θ_j). Draw a word from the topic $w_i \sim$ Mult(ϕ_{z_i})

The goal of LDA is to invert the generative model and find the posterior distribution of the latent variables conditional on the documents.

4.2 Correlated Topic Models(CTM)

The LDA model assumes that the words of each document arise from a mixture of topics, each of which is a distribution over the vocabulary. A limitation of LDA is the inability to model topic correlation even though, for example, a document about genetics is more likely to also be about disease than x-ray astronomy[2]. This limitation stems from the independence assumptions implicit in the Dirichlet distribution on the topic proportions. Under a Dirichlet, the components of the proportions vector are nearly independent; this leads to the strong and unrealistic modeling assumption that the presence of one topic is not correlated with the presence of another. Topic proportions exhibit correlation via the logistic normal distribution[2].

Let $\{\mu, \Sigma\}$be a K-dimensional mean and covariance matrix, and let topics $\beta_{1:K}$ be K multinomials over a fixed word vocabulary, as above. The CTM assumes that an N-word document arises from the following generative process:

- Draw $\eta|\{\mu, \Sigma\} \sim N(\mu, \Sigma)$

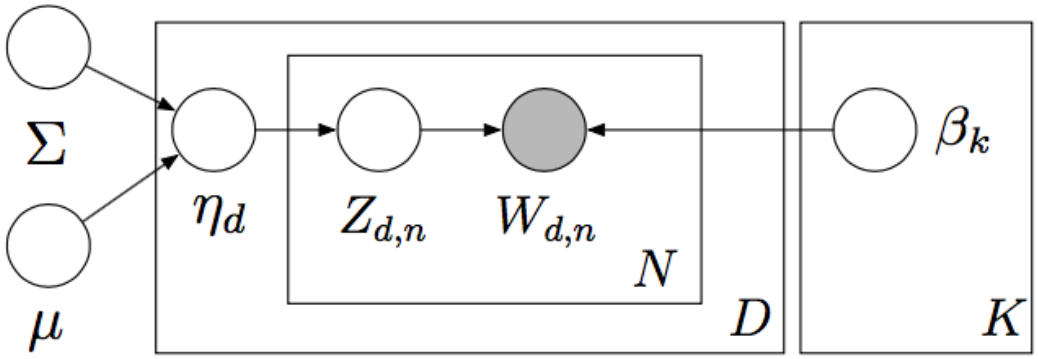

Figure 2: Correlated topic models

- For $n \in \{1, ..., N\}$
  - Draw topic assignment $Z_n|\eta$ from Mult(f(η)).
  - Draw word $W_n|\{z_n, \beta_{1:K}\}$ from Mult(β_{z_n}).

The function that maps the real-vector η to the simplex is

$$f(\eta_i) = \frac{\exp(\eta_i)}{\sum_j \exp(\eta_j)}$$

In r, topicmodels package has function CTM[3].

4.3 Model evaluations

To compare between models, including comparing same LDA models with different number of topics, and comparing LDA and its extensions, we can use the following criterions which frequently appear in other situations.

4.3.1 Log likelihood

An intuitive way to measure the models is to compare the log likelihood of the data across models. If we have got enough computation efficiency, we can use ten-fold cross validation and compute the log probability $\log P(\mathbf{w}|\mathbf{z}, T)$ of the held-out data given a model estimated from the remaining data. Higher log probability indicates a better model.

4.3.2 Perplexity

The perplexity is often used to evaluate the models on held-out data and is equivalent to the geometric mean per-word likelihood. In topicmodels package in r, the following definition is used:

$$\text{Perplexity}(w) = \exp\left\{-\frac{\log p(w)}{\sum_{d=1}^{D}\sum_{j=1}^{V} n^{(jd)}}\right\}$$

where $n^{(jd)}$ denotes how often the jth term occurred in the dth document[3]. We need log probability to calculate perplexity, but perplexity takes into account the length of document to better compare between models.

Another way of using perplexity is described in Blei's paper on correlated topic models[2]. Perplexity here is used to measure how well the models predict the remaining words of a document after observing a portion of it. Suppose we observe P words from a document, perplexity is defined as:

$$\text{Perp}(\Phi) = \left(\prod_{d=1}^{D} \prod_{i=P+1}^{N_d} p(w_i|\Phi, w_{1:P}) \right)^{-1/(\sum_{d=1}^{D}(N_d - P))}$$

4.3.3 Results on JSM data

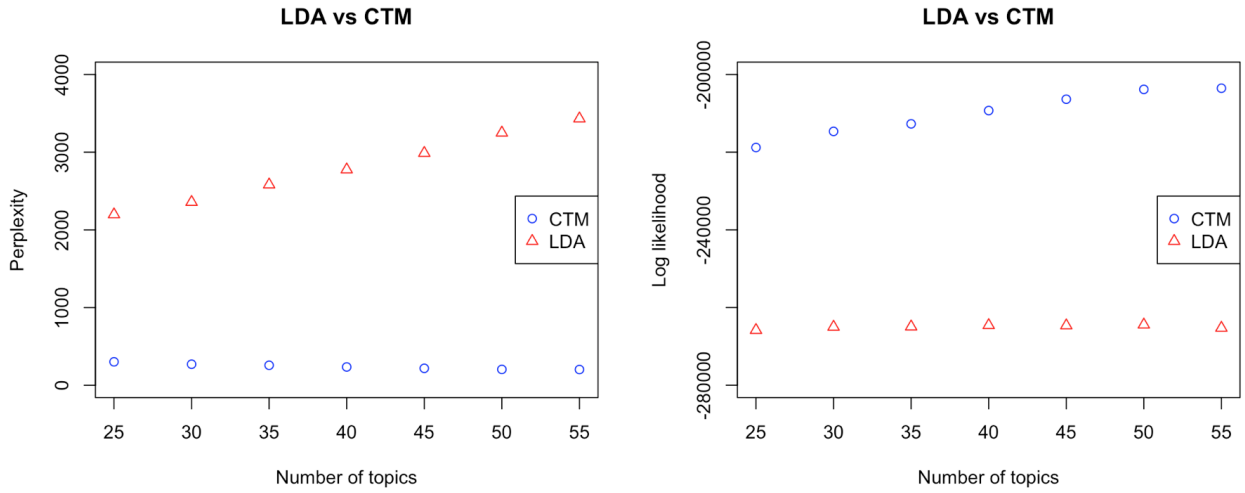

Figure 3: Comparisons of log likelihood and perplexity between CTM and LDA

A note here is that in "topicmodels" package, only variational expectation maximization is implemented for CTM[3], and for LDA I used Gibbs method. Reason for this is that VEM for LDA is not in the same scale as VEM for CTM. CTM is able to capture more topics through topic correlations, we can verify this empirically by looking into the graphs that with K being larger, for CTM perplexity is decreasing and log likelihood is increasing, while for LDA the effects are opposite.

5 Metrics for finding the natural number of topics

Above we have demonstrated how Latent Dirichlet Allocation and its extensions are able to identify topics and capture word correlations. However, it is important but difficult to select an optimal number or range of topics to fit a topic model. Here I will describe some of the metrics devised in recent years and in the end of the section I will run them on our JSM dataset and compare the results.

5.1 Griffiths2004

Thomas Griffiths and Mark Steyvers[4] proposed using Markov chain Monte Carlo algorithm for inference. As described above in the LDA section, we have prior parameters α and β. Griffiths fixed α and β and explore the consequences of varying T. This is a model selection problem using Gibbs sampling and log likelihood.

We have words $\mathbf{w}$ as the data in the corpus and the number of topics T specifying the model together to compute the likelihood P($\mathbf{w}|T$). With $\mathbf{z}$ being the assignments of words to topics, the complication here is that we need to sum over all possible assignments of $\mathbf{z}$. One approach is to calculate the harmonic mean of a set of values of P($\mathbf{w}|\mathbf{z}, T$) when $\mathbf{z}$ is sample from Gibbs sampler P($\mathbf{z}|\mathbf{w}, T$)[4]. The value of P($\mathbf{w}|\mathbf{z}, T$) can be computed as follows:

$$P(\mathbf{w}|\mathbf{z}, T) = \left(\frac{\Gamma(W\beta)}{\Gamma(\beta)^W}\right)^T \prod_{j=1}^{T} \frac{\prod_w \Gamma(n_j^{(w)} + \beta)}{\Gamma(n_j^{(\cdot)} + W\beta)}$$

in which $n_j^{(w)}$ is the number of times word $\mathbf{w}$ has been assigned to topic j in vector assignments $\mathbf{z}$, W is the total number of words, and $\Gamma(\cdot)$ is the standard gamma function[4].

Now we compute estimate of log P($\mathbf{w}|T$) for different T values. For all values of T, we run MCMC, burn the iteration samples that haven't converge, and take converged samples to calculate log P($\mathbf{w}|T$). Now we can pick the summit(or the range) that has the highest log P($\mathbf{w}|T$). The corresponding number of topics T is what we are looking for. Normally we don't select a number as optimal but instead use a small range, this general solution is considered more robust.

5.2 CaoJuan2009

Cao and her co-workers proposed a method of adaptively selecting the best LDA model based on density[5]. The motivation behind this is that the words representing several topics are likely to generate new topics. After computing the density of each topic we find out unstable topics and iteratively update number of topic T until the model is stable.

5.2.1 Best number of topics K and topic correlations

Before illustrating the definitions of topic density and the concrete algorithm, let's take a look at the relation between number of topic and topic correlations. Here I am reproducing the example from Cao. Suppose we have a corpus as following[5]:

Doc1	drug clinical patients
Doc2	drug disease
Doc3	HIV virus aids
Doc4	aids HIV disease

Table 1: Reproducible example documents

Now we fit LDA models with topic number K = 2,3,4. From the figure we can see that, when K = 2, two topics overlay on the word "drug" and the proportions are close. This reflects relatively strong correlations between two topics and "drug" is considered an unstable word. All words seem to be stable when K = 3, and there are again significant overlaps on couple of words when K = 4. This indicates that new topic could be generated from unstable words.

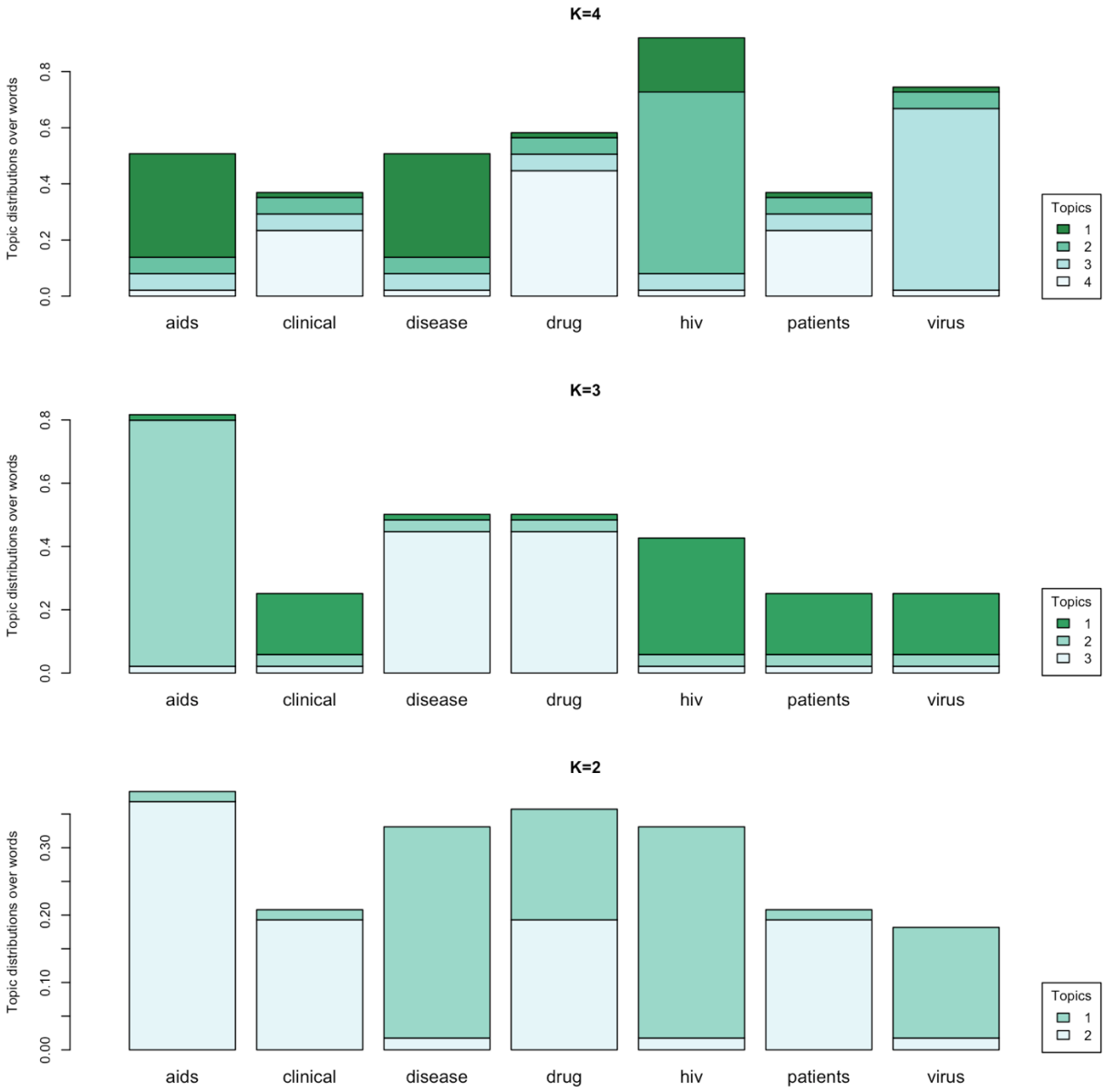

Figure 4: Topic distributions over words

We can quantify the finding by introducing cosine distance to measure the correlations of the topics:

$$\text{corre}(T_i, T_j) = \frac{\sum_{v=1}^{V} T_{iv} T_{jv}}{\sqrt{\sum_{v=1}^{V} (T_{iv})^2} \sqrt{\sum_{v=1}^{V} (T_{jv})^2}}$$

and use the average cosine distance between every pair of topics to measure the stability of topic structure:

$$\text{avg\_dis(structure)} = \frac{\sum_{i=0}^{K} \sum_{j=i+1}^{K} \text{corre}(T_i, T_j)}{K \times (K-1)/2}$$

smaller average cosine distance indicates more independent topics. In the above case we have the measures 0.2282496, 0.1164192, 0.2275417 for K = 4,3,2, showing that K = 3 is more stable among them.

5.2.2 Density based algorithm

From above we have a intuition of the relations between K and topic distances. Cao introduced clustering based idea into the method, trying to make the similarities in each (topic)cluster as large as possible, while making the distances cross (topic)clusters farthest. Here are some definitions[5] from Cao's paper before moving on to the algorithm:

Topic density: Given a topic $\mathbf{Z}$ and the distance $\mathbf{r}$, by computing the average cosine distance between $\mathbf{Z}$ and the other topics, the number of topics within the radius of $\mathbf{r}$ from $\mathbf{Z}$ is the density of $\mathbf{Z}$, called Density($\mathbf{Z}$, $\mathbf{r}$).

Model cardinality: Given a topic model $\mathbf{M}$ and a positive integer n, the number of topics whose topic densities are less than $\mathbf{n}$ is the cardinality of $\mathbf{M}$, called Cardinality($\mathbf{M}$, $\mathbf{n}$).

Reference sample: Given a topic $\mathbf{Z}$, radius $\mathbf{r}$ and threshold $\mathbf{n}$, if Density($\mathbf{Z}, \mathbf{r}$) $\leq \mathbf{n}$, then call the word distribution vector of $\mathbf{Z}$ as a reference sample of topic $\mathbf{Z}$.

Algorithm: While average cosine distance and cardinality of LDA both haven't converged:

1. Given an arbitrary number of topics K_0, randomly initialize the sufficient statistics and use the variational EM algorithm to estimate the model parameters, we obtain the initial model LDA (α, β);

2. We sequentially compute the model's average cosine distance $\mathbf{r1}$ = ave_dis(β), the densities of all the topics Density($\mathbf{Z}, \mathbf{r1}$), and the cardinality of the old model C = Cardinality(LDA,0);

3. Re-estimate the number of topic K based on C

$$K_{n+1} = K_n + f(r) \times (K_n - C_n)$$

   f($\mathbf{r}$) is the changing direction of $\mathbf{r}$. If the direction is negative (opposite to the former), then $f_{n+1}(\mathbf{r}) = -1 * f_n(\mathbf{r})$, else $f_{n+1}(\mathbf{r}) = f_n(\mathbf{r})$. When the convergence direction is negative, we ascending sort the topics by the densities, and extract the former K topics as the reference samples to initialize the sufficient statistics. When the convergence direction is positive, we randomly initialize the sufficient statistics with seed.

4. Go through step 2 and 3 until convergence.

To sum up, the whole idea of this algorithm is started from identifying a relation between number of topic K and the topic correlations, as suggested by the previous subsection. Then the author uses clustering idea and define distance and density for iterating the process until convergence.

5.3 Arun2010

Arun in 2010 questioned Cao's approach, since she only considered topic-word matrix and ignore document-topic matrix[6]. Desiring to come up with a more robust approach, Arun views LDA as a matrix factorization method factorizing document-word frequency matrix M into T×W matrix M1 and D×T matrix M2. And he proposed a new measure that computes the symmetric Kullback-Leibler divergence of the Singluar value distributions of matrix M1 and the distribution of the vector L×M2 where L is a 1×D vector containing the lengths of each document in the corpus. The proposed measure is the following:

$$\text{Proposed Measure}(M1, M2) = KL(C_{M1}||C_{M2}) + KL(C_{M2}||C_{M1})$$

where C_{M1} is the distribution of singular values of Topic-Word matrix M1, C_{M2} is the distribution obtained by normalizing the vector L$\times$M2 (where L is 1$\times$D vector of lengths of each document in the corpus and M2 is the Document-Topic matrix). Notice that both the distributions C_{M1} and C_{M2} are in sorted order so that the corresponding topic components are expected to match[6].

5.4 Results on JSM data

Here is the comparison of all three metrics described above:

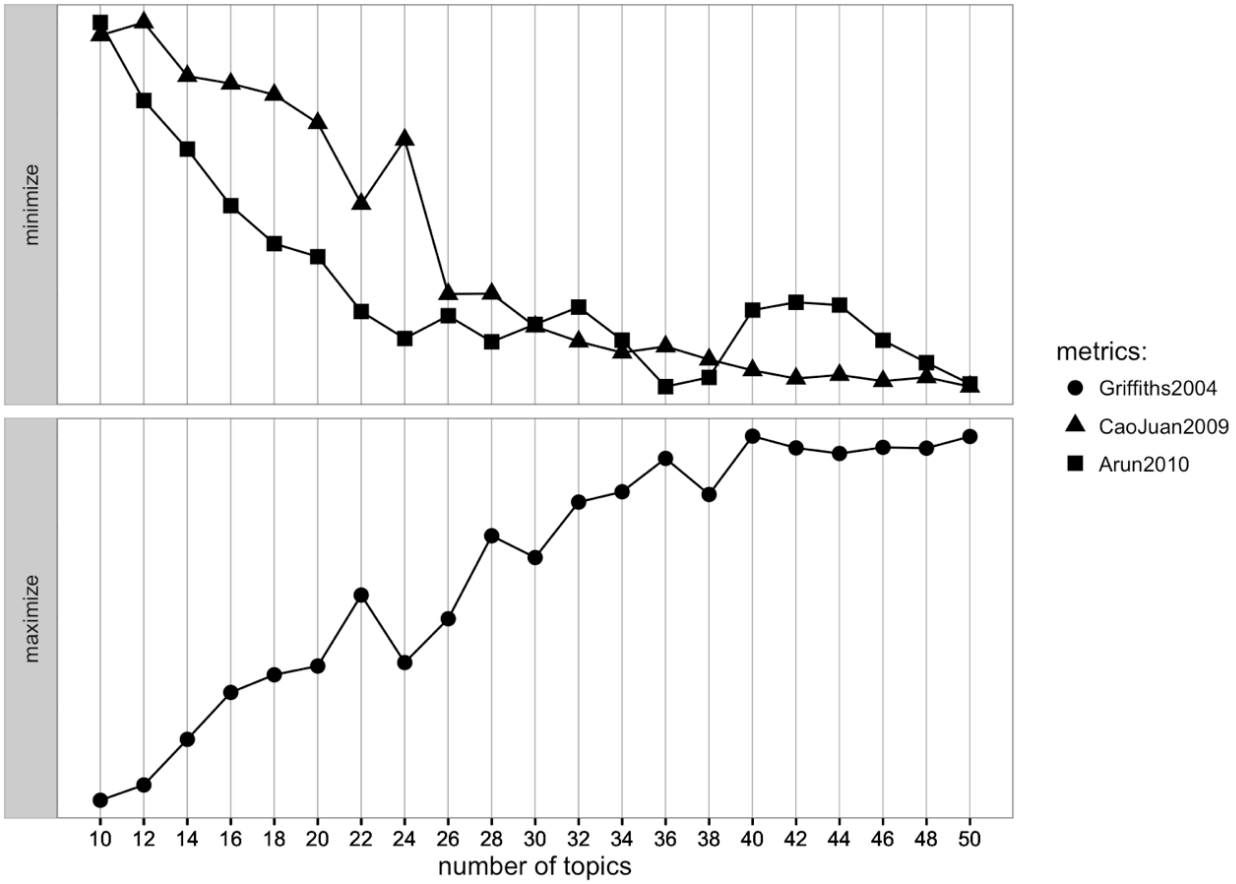

Figure 5: Metrics to determine the optimal range of topic number

As mentioned above, an optimal range of topic number is more robust, and in JSM data metrics return a range between 36 and 42. The metrics are measured under Gibbs method in LDA, so we could expect a slightly higher K for correlated topic models.

6 Mathematical Optimization

Optimization is the hardest part of the whole project. After obtaining topic proportions, how can we formulate the optimization problem determine whether we have reach to a point that we are satisfied with? Following is one way to formulate the problem.

6.1 Problem formulation

$$
\begin{aligned}
\underset{X}{\text{Minimize}} \quad & \sum_{i \neq j} X_i \cdot X_j * I(S_i, S_j) \\
\text{subject to} \quad & \forall i, \sum_j X_i \cdot X_j \leq N \\
& \forall i, X_i \cdot X_i = 1 \\
& I(S_i, S_j) = f(\max_{\sigma} \min(p_{i\sigma}, p_{j\sigma})) \\
& f(x) = \begin{cases} 3x, & \text{if } 0.5 < x \\ x, & \text{otherwise} \end{cases}
\end{aligned}
$$

σ represents each topic, and p_σ is the topic σ vector. f is a penalty function, if two sessions are tending towards the same topic, which in this case both are having a higher than 0.5 probability belonging to the same topic, we will penalize this condition leading a higher objective function.

$I(S_i, S_j)$ is an interface function between any two sessions, or in this case, i and j. Suppose we have three topics, and the topic vector for one session is [0.6, 0.3, 0.1], another vector might be [0.3, 0.6, 0.1]. Then the interference between these two sessions if they are within the same time slot is f(0.3). Similarly, for two sessions with topic vectors as [0.7, 0,2, 0.1] and [0.6, 0.3, 0.1], the interference is f(0.6). One can change the function f itself according to the context of her optimization problem.

Each X_i is a m dimensional vector where m indicates the number of time slots. For example, X_i (0,1,0,0,0,0) means that session i takes place during the second time slot of six time slots. if $X_i \neq X_j$where $i \neq j$, then $X_i \cdot X_j = 0$.

The objective function we define above is also used as a quantifier, defined as total interference score. The lower the score, the better the arrangement is.

6.2 Greedy Algorithm

Here we propose a greedy algorithm. Greedy algorithm is being used in many cases where we want to reach local optimum every time and hope we can reach global optimum when the algorithm stops. However, this is never the guarantee, but we can integrate some other techniques to give us enough confidence with our results.

This greedy algorithm is describe as follows. For iteration l from 1 through k:

1. Assign each session randomly to a time slot with the constraints mentioned at the beginning of this chapter, calculate $TF_l = \sum_{i \neq j} X_i \cdot X_j * I(S_i, S_j)$.

2. For each session i:

   (a) If session i interchange time slots with session $j \neq i$, we extract sessions that are in the same time slot either with i or j, and calculate the total interference score TF_{ij}.

   (b) Among all TF_{ij}, find the j^* that has the minimum TF_{ij}, make session i interchange time slots with session j^* .

   (c) If $TF_l > TF_{ij^*}$, assign TF_{ij^*} to TF_l, do the above steps with $i + 1$. if not, stop the iteration, store TF_l and X_{il} for all session i.

This seems to be a good start, but we can run into local optimum without knowing so. One way to start with is to have 100 random session assignment and run the algorithm. But in terms of speed, "eachcomp" function would take around 7 minutes to finish, and we have to run it for 598 sessions and at least 3 iterations, a roughly total of 12500 minutes(more than 8 days). If we want to check whether we get the same optimum with different starts, it will take weeks.

7 Reflection and future work

7.1 Summary of work and limitation

For the report I have made some reasonable improvements based on Yongjian's work:

- Improve on some of the data cleaning steps, including leaving out tf-idf criterion.
- Introduce correlated topic models, and perform model evaluations between LDA and CTM.
- Look into papers and packages that could help determining the optimal range of topic number.

I also tried

- Improving the greedy algorithm by re-formulating the problem, but it doesn't seem to work out nicely.
- Speed up the ctm function in "topicmodel" by using ctm-c in Blei's lab, but some issues came up with environment configuration.
- Dive into Hierarchical Dirichlet Process and its online version for LDA[7].

Currently the biggest issue lies in that both CTM function in r and greedy algorithm are time consuming. Each run of CTM would take around 2 hours to finish without running into unconvergence problem. And as mentioned above, greedy algorithm is not well-implemented and would take weeks to run.

Other issue I am concerned about is the method of computing the posterior for both LDA and CTM, since variational EM is the only method implemented in "ctm" function, to make two models comparable we need to take a deeper look into how these methods are used and what cautions we need to care for.

7.2 Further investigation of Hierarchical Dirichlet Process LDA

HDP is an extension of LDA, designed to address the case where the number of mixture components is not known a priori[8]. As far as pros and cons, HDP has the advantage that the maximum number of topics can be unbounded and learned from the data rather than specified in advance. It is more complicated to implement, and unnecessary in the case where a bounded number of topics is acceptable. Currently it is partly implemented in gensim package in Python as "hdpmodel". I ran the function to the data, but there are no options returning the posteriors. The posterior over the number of topics obtained under the hierarchical DP mixture model can be used to determine the optimal number of topic. It has been empirically proved in Teh's paper that the optimal obtained by HDP is consistent with the result return by using perplexity in LDA[8].

A separate and deeper work could be done from here.

8 Acknowledgements

This is my first time looking into natural language processing and the materials seemed overwhelming at the beginning. I want to thank Professor David Banks for the constant help, the independent study group for the sharing and comments, the former researcher Yongjian for his initial idea.

References

[1] David M Blei and John D Lafferty. Topic models. *Text mining: classification, clustering, and applications*, 10(71):34, 2009.

[2] David Blei and John Lafferty. Correlated topic models. *Advances in neural information processing systems*, 18:147, 2006.

[3] Kurt Hornik and Bettina Grün. topicmodels: An r package for fitting topic models. *Journal of Statistical Software*, 40(13):1–30, 2011.

[4] Thomas L Griffiths and Mark Steyvers. Finding scientific topics. *Proceedings of the National academy of Sciences*, 101(suppl 1):5228–5235, 2004.

[5] Juan Cao, Tian Xia, Jintao Li, Yongdong Zhang, and Sheng Tang. A density-based method for adaptive lda model selection. *Neurocomputing*, 72(7):1775–1781, 2009.

[6] R Arun, Venkatasubramaniyan Suresh, CE Veni Madhavan, and MN Narasimha Murthy. On finding the natural number of topics with latent dirichlet allocation: Some observations. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 391–402. Springer, 2010.

[7] Chong Wang, John William Paisley, and David M Blei. Online variational inference for the hierarchical dirichlet process. In *AISTATS*, volume 2, page 4, 2011.

[8] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.