Latent Dirichlet Allocation on Reddit Text Study

Tom Tran

February 12, 2016

Abstract

In this paper, I examine **Latent Dirichlet Allocation** (LDA), a topic modeling technique. I summarize the literature on methodology including the generative process, model estimation, and model section. I report the use of software for LDA in R environment. I demonstrate the applications of LDA to the chosen data set. I show that collapsed Gibbs sampling is the better method of estimation for this dataset than Variational Expectation-Maximization.

1 Introduction

1.1 Background

With the rise of social networking sites in recent years, text data has been of particular interest to both industry and academia. By analyzing these user-generated sources, we can get some insight into what people discuss on social media. Additionally, the massive volumes of these data sets require a computational method for extracting information without manual efforts. The method being used is **Latent Dirichlet Allocation** (Blei, Ng, and Jordan, 2003), one of the most popular tools for topic modeling. With LDA, we can computationally identify the topics in a text dataset, monitor a particular topic of interest over time, and create a model for prediction of new data. The data set being used for this project is collected from Reddit.com.

1.2 Data

Reddit is a social networking website where registered members can submit contents, such as text posts, pictures, direct links, etc. making it an online bulletin board system. Registered members can comment on each post to voice their opinions and discuss with other members. The data set obtained has the following variables: title, score, selftext, num_comments. All of these variables' values are created by users. When a user creates a post, they create a title as the required field. Selftext (usually null) is an optional field that provides additonal information about the title. The score is calculated on the number of upvotes (i.e. likes) and downvotes (i.e. dislikes). Variable num_comments is the number of comments on the

post. The primary variable for this analysis is the title of the post. The data set being analyzed contains the posts from Technology, a subset of Reddit community, where users discuss technology products, events, and news from January to August of 2015. The size of the monthly data with about 12,000 posts each is 1 MB. Table **??** shows examples of posts in reddit.com/r/technology.

title	selftext	score	num_comments
My college needs some better security on their news monitor. This was too easy.		1	1
FCC Chair Tom Wheeler: "4Mbps connection isnt exactly whats necessary in the 21st century."		1191	250
Tim Cook's Philosophy at Apple, in His Own Words		1	0

Table 1: Examples of Technology Posts

2 Methodology

This section discusses the generative process, model estimation, and model selection in LDA.

2.1 Generative Process

LDA, which is a Bayesian hierarchical model, assumes that the characteristics of topics and documents are drawn from the Dirichlet distribution. The Dirichlet distribution is the multivariate generalization of the Beta distribution. It is the conjugate prior for the multinomial distribution. The LDA's generative process can be summarized as (Blei and Lafferty, 2009):

1. Draw a distribution of words ϕ_k for each topic k: $\phi_k \sim Dirichlet\,(\beta)$
2. Draw topic distribution for each document d: $\theta_d \sim Dirichlet\,(\alpha)$

For each word i of N words in a document:

(a) Choose a topic assignment for each word: $z_{d,i} \sim Multinomial(\theta_d), z_{d,i} \in \{1, ..., K\}$

(b) Choose a word $w_{d,i} \sim Multinomial(\phi_{z_{d,i}}), w_{d,i} \in \{1, ..., V\}$ where V is the number of words in the corpus (collection of all documents).

The graphical model can be generated as in Figure 1. Note that the only observed variable is word, which is represented by a shaded node. The others are latent variables.

The joint distribution resulting from the graphical model is:

$$p(w, z, \theta, \phi|\alpha, \beta) = p(w|z, \phi)p(z|\theta)p(\phi|\beta)p(\theta|\alpha)$$

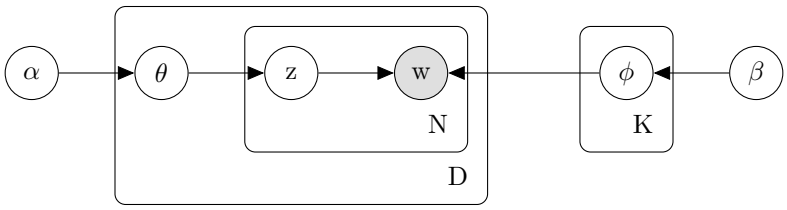

Figure 1: Directed Acyclical Graph of LDA (adapted from Blei and Lafferty, 2009)

The formula for each component is as follows. Topic distribution for all documents:

$$p(\theta|\alpha) = \prod_{d=1}^{D} \left(\frac{\Gamma(\alpha.)}{\prod_{k=1}^{K} \Gamma(\alpha_k)} \prod_{k=1}^{K} \theta_{d,k}^{\alpha_k - 1} \right)$$

Word distribution for all topics:

$$p(\phi|\beta) = \prod_{k=1}^{K} \left(\frac{\Gamma(\beta_{k.})}{\prod_{v=1}^{V} \Gamma(\beta_{k,v})} \prod_{v=1}^{V} \phi_{k,v}^{\beta_{k,v} - 1} \right)$$

Topic assignment to words:

$$p(z|\theta) = \prod_{d=1}^{D} \prod_{k=1}^{K} \theta_{d,k}^{n_{d,k,\cdot}}$$

$n_{d,k,\cdot}$ is the count of times topic k is assigned to any word in document d.

The likelihood for the corpus:

$$p(w|z,\phi) = \prod_{k=1}^{K} \prod_{v=1}^{V} \phi_{k,v}^{n_{\cdot,k,v}}$$

$n_{\cdot,k,v}$ is the count of times topic k is assigned to word v in the whole corpus.

The joint distribution is therefore:

$$p(w,z,\theta,\phi|\alpha,\beta) = p(w|z,\phi)p(z|\theta)p(\phi|\beta)p(\theta|\alpha)$$

$$= \prod_{k=1}^{K} \prod_{v=1}^{V} \phi_{k,v}^{n_{\cdot,k,v}} \times \prod_{d=1}^{D} \prod_{k=1}^{K} \theta_{d,k}^{n_{d,k,\cdot}} \times \prod_{k=1}^{K} \left(\frac{\Gamma(\beta_{k.})}{\prod_{v=1}^{V} \Gamma(\beta_{k,v})} \prod_{v=1}^{V} \phi_{k,v}^{\beta_{k,v} - 1} \right) \times \prod_{d=1}^{D} \left(\frac{\Gamma(\alpha.)}{\prod_{k=1}^{K} \Gamma(\alpha_k)} \prod_{k=1}^{K} \theta_{d,k}^{\alpha_k - 1} \right)$$

$$= \left(\prod_{d=1}^{D} \frac{\Gamma(\alpha.)}{\prod_{k=1}^{K} \Gamma(\alpha_k)} \prod_{k=1}^{K} \theta_{d,k}^{\alpha_k + n_{d,k,\cdot} - 1} \right) \times \left(\prod_{k=1}^{K} \frac{\Gamma(\beta_{k.})}{\prod_{v=1}^{V} \Gamma(\beta_{k,v})} \prod_{v=1}^{V} \phi_{k,v}^{\beta_{k,v} + n_{\cdot,k,v} - 1} \right)$$

2.2 Model Estimation

The original paper details the Variational Expectation-Maximization (VEM) method for estimating the parameters. Model estimation can also be done by other methods including Maximum Likelihood, Maximum a Posterior, Variational Bayesian Inference, Collapsed Variational Bayesian Inference, Collapsed Gibbs Sampling, and Expectation Propagation. Gibbs

sampling is briefly introduced in the course Bayesian Statistics (STAT4520); therefore, it is educationally valuable to look into this method. For Collapsed Gibbs Sampling, instead of explicitly representing ϕ or θ as parameters to be estimated, we consider the posterior distribution of the assignments of words to topics $P(\mathbf{z}|\mathbf{w})$. We need the full conditional distribution of $P(z_i|\mathbf{z}_{-i}, \mathbf{w})$. This distribution can be calculated after expansions and transformations (Griffiths and Steyvers, 2004) as:

$$P(z_i = j|\mathbf{z}_{-i}, \mathbf{w}) \propto \frac{n_{-i,j}^{(d_i)} + \alpha}{n_{-i,\cdot}^{(d_i)} + K\alpha} \times \frac{n_{-i,j}^{(w_i)} + \beta}{n_{-i,j}^{(\cdot)} + V\beta}$$

$n_{-i,j}^{(d_i)}$ is the number of times topic j is assigned to some word token in document d, not including the current instance i. $n_{-i,j}^{(w_i)}$ is the number of times word w is assigned to topic j, not including the current instance i. The equation is then intuitive. The first ratio is the probability of topic j in document d_i. The second ratio is the probability of w_i under topic j.

The Gibbs sampling procedure is then carried out to determine the posterior $P(\mathbf{z}|\mathbf{w})$ (Steyvers and Griffiths, 2007). For any sample, we can then estimate θ and ϕ by the value of $\mathbf{z}$:

$$\hat{\phi}_j^{(w)} = \frac{n_j^{(w)} + \beta}{n_j^{(\cdot)} + V\beta}$$

$$\hat{\theta}_j^{(d)} = \frac{n_j^{(d)} + \alpha}{n_{\cdot}^{(d)} + K\alpha}$$

Compared to VEM, collapsed Gibbs sampling has an important advantage. Since EM algorithm is susceptible to problems involving local maxima and is slow to converge, Gibbs sampling has been proven to be a more stable method for model estimation (Griffiths and Steyvers, 2004).

2.3 Model Evaluation and Selection

Perplexity is conventionally used in language modeling, which is equivalent to the inverse of geometric mean per-word likelihood. A lower perplexity on the held-out data indicates a model with better generalization.

$$perplexity(D_{test}) = \exp\left\{ -\frac{\sum_{d=1}^{D} \log p(\mathbf{w}_d)}{\sum_{d=1}^{D} N_d} \right\}$$

3 Software

Package **topicmodels** is used for the application of LDA in R environment. The outputs in this paper were produced in R. **topicmodels** in R has a comprehensive functionality. In addition, R has a variety of strong preprocessing packages such as **tm**, **RTextTools**, and **quanteda**.

RTextTools and **tm** are very popular tools in the natural language processing community. However, **quanteda** shows advantages when a large number of documents need to be processed that **tm** and **RTextTools** fail to deliver. The preprocessing for the data set before applying LDA therefore is done with **quanteda**.

The main function used in **topicmodels** package is `LDA()`. It requires two parameters: `x` - the document-term matrix and `k` - the number of topics. An optional argument is `method`, which users can specify whether `Gibbs` or `VEM` will be used. The default is `VEM`. This function returns an object of class `LDA`.

4 Application

This section demonstrates the application of LDA on Technology data for 2015.

4.1 Preprocessing

Function `dfm()` from quanteda is used to remove punctuation, tokenize, stem words, remove stop-words, and remove sparse terms. In addition, words that could be considered stop-words such as 'reddit' and 'redditor' and common spelling errors are also removed. This function also creates a document-feature matrix (equivalent to document-term matrix for topicmodels package) with the columns for terms (words) and rows for documents. This matrix contains the frequency of each word for all the documents.

4.2 Model Selection

In order to determine the optimal number of topics, function `perplexity()` from package **topicmodels** is performed for different models using January data with both Gibbs and VEM methods. By first fitting the model with $k \in \{5, 15, 25, 35, 45\}$ and the required parameters, we can see in Figure 2 that for Gibbs models, the lowest perplexity seems to be models with k between 20 and 30. VEM models, on the other hand, seem to have some problems, since the lower perplexity is attained with smaller k's. In fact, lowest perplexity attained is when $k = 2$, which does not reflect the true number of topics for a collections of a lot of documents. Also, Figure 2 and 3 show that Gibbs models attain much lower perplexity than VEM's, which further indicates the problem with VEM's. Since we look for a model with lowest perplexity, upon increasing the resolution of perplexity, the graph of Gibbs models shows that the optimal topic number might be around 25.

4.3 Model Fitting and Interpretation

1 **Comparison of Gibbs and VEM Outputs for January**

LDA model is fitted using function `LDA()` from **topicmodels** as mentioned above. Fitting the model for January document-term matrix with k = 25, method = 'Gibbs', and using function `terms()` provided by **topicmodels** on the returned `LDA` object

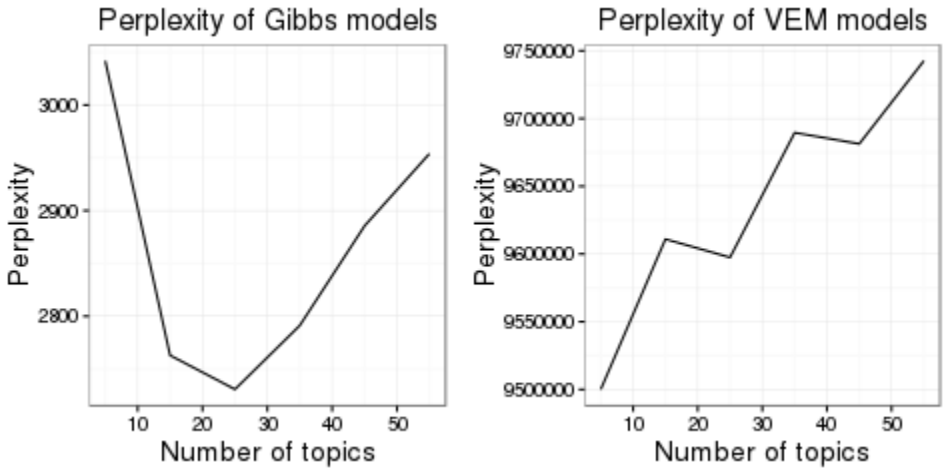

Figure 2: k = 5, 15, 25, 35, 45

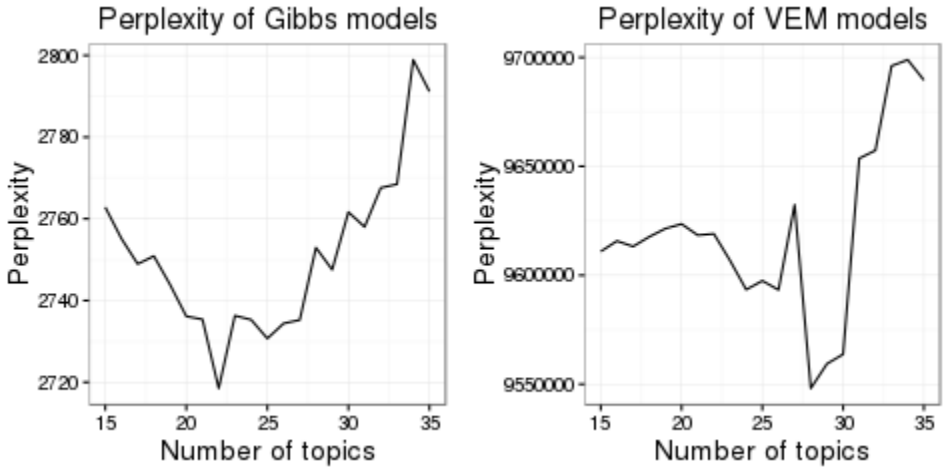

Figure 3: k = 15, 16, ..., 34, 35

named `LDAModel1` to get the top 10 words for each topic gives the following output. The words are ordered by decreasing probability in each topic ($\hat{\phi}$). We can identify the topics by their top words. For example, topic 21 is about the Sony hack incident in early 2015 with keywords such as 'hack', 'attack', 'soni', etc. All of these words consistently represent the language revolving the incident. Table 2 shows examples of four topics from the output that consists of 25 topics for our Gibbs model.

```
> terms(LDAModel1,10)
      Topic 1  Topic 2    Topic 3     Topic 4   Topic 5    Topic 6  Topic 7
```

```
 [1,] "servic" "googl"    "smartphon" "new"     "user"     "best"   "comcast"
 [2,] "web"    "come"     "samsung"   "car"     "one"      "buy"    "time"
 [3,] "busi"   "project"  "launch"    "ces"     "wireless" "tv"     "chang"
 [4,] "solut"  "glass"    "review"    "sale"    "million"  "get"    "cabl"
 [5,] "design" "search"   "galaxi"    "electr"  "go"       "smart"  "get"
 [6,] "system" "fiber"    "price"     "batteri" "allow"    "websit" "custom"
 [7,] "manag"  "next"     "india"     "machin"  "verizon"  "offer"  "block"
 [8,] "cloud"  "translat" "note"      "drive"   "turn"     "game"   "say"
 [9,] "lead"   "citi"     "ces"       "tesla"   "let"      "case"   "wi-fi"
[10,] "save"   "becom"    "specif"    "unveil"  "sell"     "led"    "realli"
      Topic 8      Topic 9  Topic 10  Topic 11    Topic 12     Topic 13
 [1,] "app"        "new"    "year"    "technolog" "android"    "use"
 [2,] "mobil"      "releas" "first"   "market"    "phone"      "comput"
 [3,] "develop"    "chrome" "world"   "futur"     "now"        "whatsapp"
 [4,] "compani"    "featur" "power"   "digit"     "devic"      "work"
 [5,] "blackberri" "see"    "solar"   "social"    "io"         "intel"
 [6,] "applic"     "take"   "top"     "big"       "updat"      "pc"
 [7,] "india"      "site"   "energi"  "media"     "avail"      "ani"
 [8,] "platform"   "tool"   "better"  "share"     "game"       "messag"
 [9,] "hire"       "scienc" "startup" "trend"     "app"        "report"
[10,] "network"    "extens" "look"    "top"       "gotowebsit" "post"
      Topic 14  Topic 15  Topic 16   Topic 17    Topic 18   Topic 19
 [1,] "whi"     "data"    "tech"     "onlin"     "appl"     "video"
 [2,] "like"    "call"    "support"  "technolog" "iphon"    "free"
 [3,] "need"    "obama"   "help"     "3d"        "watch"    "download"
 [4,] "look"    "encrypt" "number"   "back"      "ipad"     "crack"
 [5,] "peopl"   "privaci" "gmail"    "say"       "camera"   "key"
 [6,] "know"    "law"     "custom"   "real"      "set"      "youtub"
 [7,] "think"   "want"    "news"     "print"     "os"       "pro"
 [8,] "control" "state"   "technic"  "laptop"    "patent"   "full"
 [9,] "take"    "us"      "password" "pirat"     "china"    "plus"
[10,] "doe"     "uk"      "firm"     "bay"       "electron" "file"
      Topic 20    Topic 21   Topic 22    Topic 23  Topic 24  Topic 25
 [1,] "internet"  "facebook" "microsoft" "secur"   "build"   "make"
 [2,] "fcc"       "hack"     "window"    "network" "musk"    "bill"
 [3,] "net"       "twitter"  "new"       "open"    "elon"    "robot"
 [4,] "neutral"   "soni"     "announc"   "email"   "drone"   "live"
 [5,] "broadband" "attack"   "browser"   "program" "billion" "way"
 [6,] "plan"      "hacker"   "hololen"   "spi"     "test"    "creat"
 [7,] "rule"      "us"       "offic"     "govern"  "spacex"  "thing"
 [8,] "access"    "claim"    "realiti"   "secret"  "track"   "water"
 [9,] "speed"     "north"    "virtual"   "sourc"   "space"   "intellig"
[10,] "industri"  "korea"    "upgrad"    "nsa"     "want"    "made"
```

Topic 2	Topic 3	Topic 18	Topic 20
GOOGLE	SAMSUNG	APPLE	NET NEUTRALITY
googl	smartphon	appl	internet
come	samsung	iphone	fcc
project	launch	watch	net
glass	review	ipad	neutral
search	galaxi	camera	broadband
fiber	price	set	plan
next	india	os	rule
translate	note	patent	access
citi	ces	china	speed
become	specif	electron	industri

Table 2: Four out of 25 Topics From January Gibbs Output

With method = 'VEM', the output is as follows. This output contains some problems as we expected. For example, the collection of words of topic Sony hack is "broken up" and "put" into different topics. The output shows 'soni', 'attack' and 'hack' are put into topic 16, 13 and 8 respectively, which makes the results very unreliable. Note that the topics in Gibbs and VEM outputs are not identical. For example, topic 1 in Gibbs is about Web Services, but it is not in VEM. The topic order in both outputs is random.

```
> terms(LDAModel1,10)
      Topic 1   Topic 2     Topic 3     Topic 4  Topic 5    Topic 6
 [1,] "window"  "app"       "broadband" "launch" "new"      "year"
 [2,] "develop" "best"      "internet"  "note"   "fiber"    "new"
 [3,] "india"   "devic"     "fcc"       "window" "s"        "road"
 [4,] "user"    "now"       "new"       "video"  "come"     "technolog"
 [5,] "free"    "mobil"     "free"      "servic" "power"    "call"
 [6,] "one"     "custom"    "thing"     "app"    "north"    "ceo"
 [7,] "certif"  "launch"    "appl"      "vehicl" "trick"    "say"
 [8,] "compani" "smartphon" "definit"   "car"    "intern"   "show"
 [9,] "world"   "get"       "get"       "cost"   "internet" "silk"
[10,] "time"    "comput"    "tech"      "tech"   "vpn"      "app"
      Topic 7     Topic 8    Topic 9     Topic 10   Topic 11  Topic 12
 [1,] "real"      "facebook" "new"       "offer"    "price"   "3d"
 [2,] "whi"       "servic"   "googl"     "new"      "specif"  "io"
 [3,] "facebook"  "appl"     "use"       "open"     "india"   "intel"
 [4,] "phone"     "hack"     "app"       "servic"   "detail"  "comput"
 [5,] "today"     "data"     "android"   "sourc"    "get"     "internet"
 [6,] "appl"      "keyboard" "free"      "internet" "januari" "storag"
 [7,] "valley"    "now"      "mobil"     "money"    "lead"    "googl"
```

```
 [8,] "twitter"   "batteri"  "download"  "mobil"    "iphon"   "make"
 [9,] "silicon"   "video"    "iphon"     "deal"     "youtub"  "launch"
[10,] "microsoft" "ces"      "technolog" "appl"     "help"    "whi"
      Topic 13 Topic 14    Topic 15    Topic 16    Topic 17    Topic 18
 [1,] "app"    "appl"      "servic"    "technolog" "musk"      "facebook"
 [2,] "iphon"  "microsoft" "digit"     "new"       "elon"      "tech"
 [3,] "data"   "best"      "encrypt"   "app"       "work"      "privaci"
 [4,] "mobil"  "googl"     "compani"   "soni"      "new"       "review"
 [5,] "photo"  "smartphon" "email"     "onlin"     "googl"     "whi"
 [6,] "phone"  "realli"    "technolog" "net"       "data"      "copyright"
 [7,] "attack" "phone"     "new"       "nsa"       "appl"      "hous"
 [8,] "make"   "built"     "give"      "world"     "press"     "canadian"
 [9,] "may"    "iphon"     "us"        "futur"     "date"      "forc"
[10,] "top"    "announc"   "custom"    "neutral"   "technolog" "manag"
      Topic 19  Topic 20   Topic 21    Topic 22   Topic 23  Topic 24
 [1,] "robot"   "internet" "googl"     "electron" "network" "mobil"
 [2,] "hire"    "googl"    "energi"    "china"    "new"     "internet"
 [3,] "develop" "use"      "app"       "servic"   "samsung" "must"
 [4,] "live"    "communic" "android"   "screen"   "galaxi"  "step"
 [5,] "mobil"   "govern"   "technolog" "consum"   "tv"      "de"
 [6,] "mani"    "full"     "free"      "iphon"    "use"     "onlin"
 [7,] "android" "year"     "internet"  "lizard"   "s6"      "save"
 [8,] "perform" "access"   "job"       "launch"   "app"     "googl"
 [9,] "free"    "time"     "support"   "squad"    "android" "comput"
[10,] "app"     "secur"    "camera"    "gadget"   "googl"   "develop"
      Topic 25
 [1,] "googl"
 [2,] "window"
 [3,] "control"
 [4,] "microsoft"
 [5,] "net"
 [6,] "fcc"
 [7,] "remov"
 [8,] "support"
 [9,] "onlin"
[10,] "wireless"
```

Based on the outputs and the previously discussed model evaluation, Gibbs has been shown to be a better model than VEM. Not only did Gibbs attain a much lower perplexity, but it also produced outputs that are more accurate and reliable. The topics observed in Gibbs are also more identifiable. I was able to identify almost all topics in the Gibbs output. The same statement cannot be made about the VEM

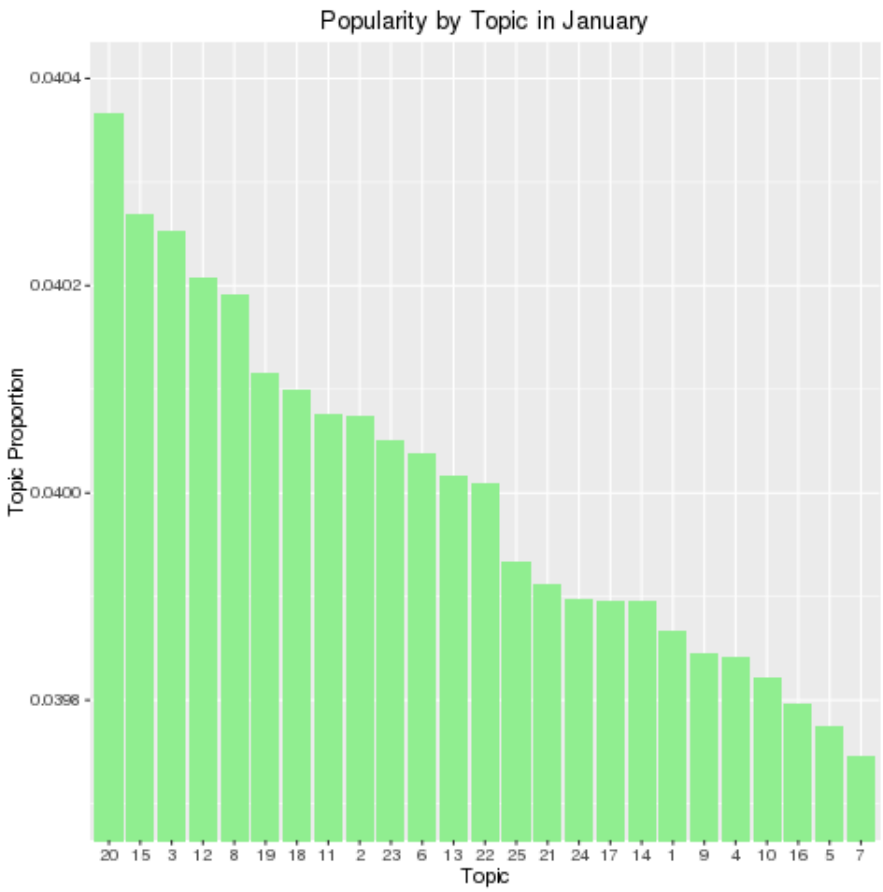

Figure 4: Popularity of Topics in January

output. Therefore, Gibbs is our chosen method moving forward.

2 Topic Proportions in January

We can estimate topic proportions for each document and calculate the average proportions for all the documents to determine the popularity by topic for the January Gibbs model. Topic 20 seems to be most talked about in January. That makes sense, since it is about the net neutrality bill introduced by the Republicans in January of 2015 that attracted a lot of attention. Figure 4 shows the popularity of each topic.

3 Topic Visualization

We can also visualize each topic with function `wordcloud()` from **wordcloud** package in order to see the importance of each word compared to the others. The words with higher probability will appear larger. The wordclouds help us in the process of understanding and determining the subject of a topic. Figure 5 and 6 show topic Net Neutrality and topic Samsung, respectively.

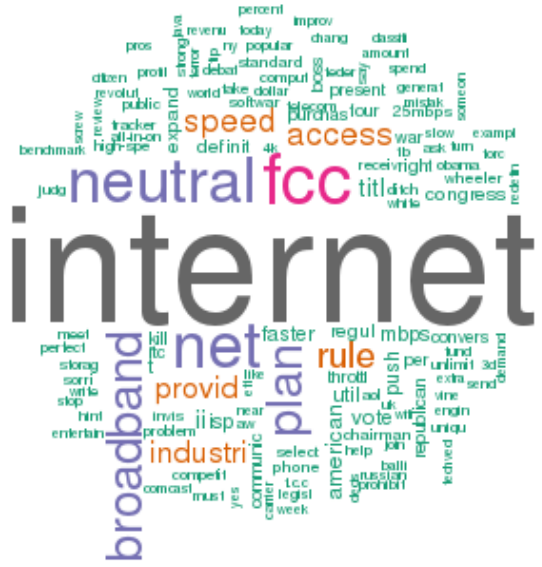

Figure 5: Topic Net Neutrality

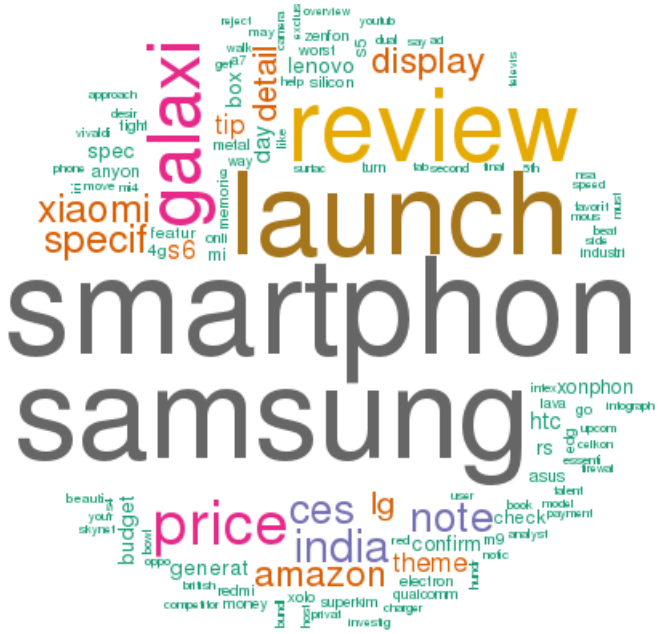

Figure 6: Topic Samsung

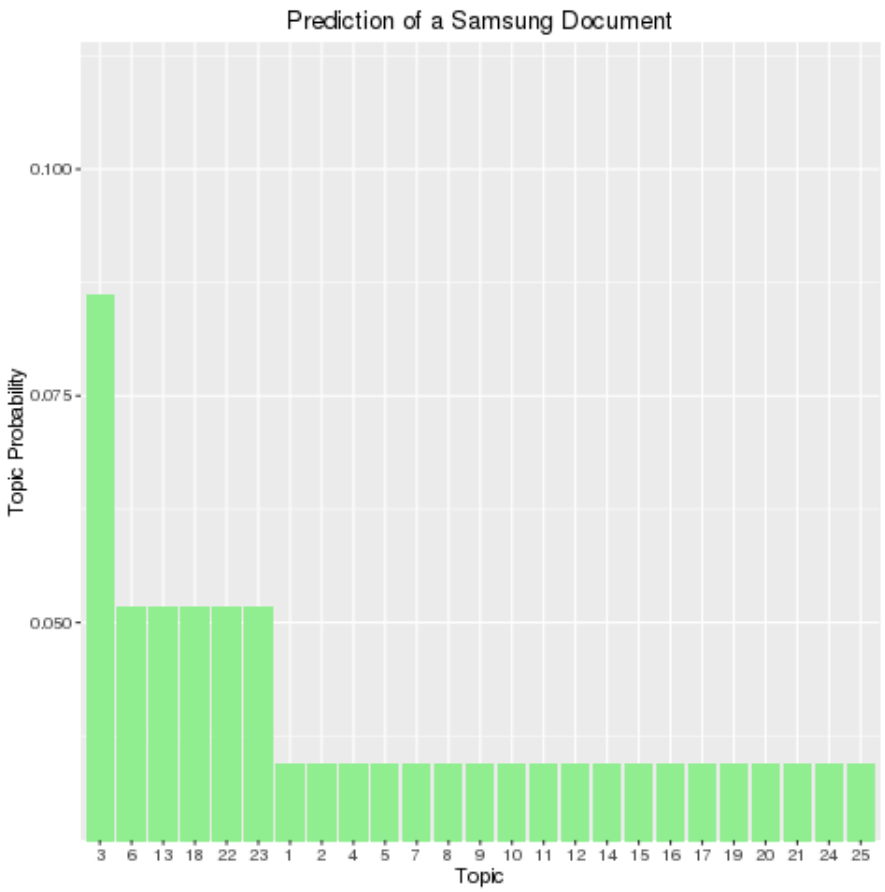

Figure 7: Topic Prediction for a Random Document

4 Prediction

One of the advantages that LDA has over its precedents is the prediction capability. We can use the fitted model to predict the topic of a new document using `posterior()` funtion. If we fit a model with data from January and use it for our prediction of a new random document below (document 50 of March), we can obtain the probabilities of this document belonging to different topics. The highest probability is for topic 3 in the Gibbs output, which is the 'Samsung' topic. Figure 7 shows the probability predicted for topic 3 compared to the others.

```
> t201503[50,1]
[1] "How Samsung may have trumped Apple to get exposure for the Galaxy S6"
```

5 Topic Monitoring

By fitting models for each month we could monitor the popularity of a particular topic over time. Figure 8 shows the popularity of topic Samsung over eight months. It makes

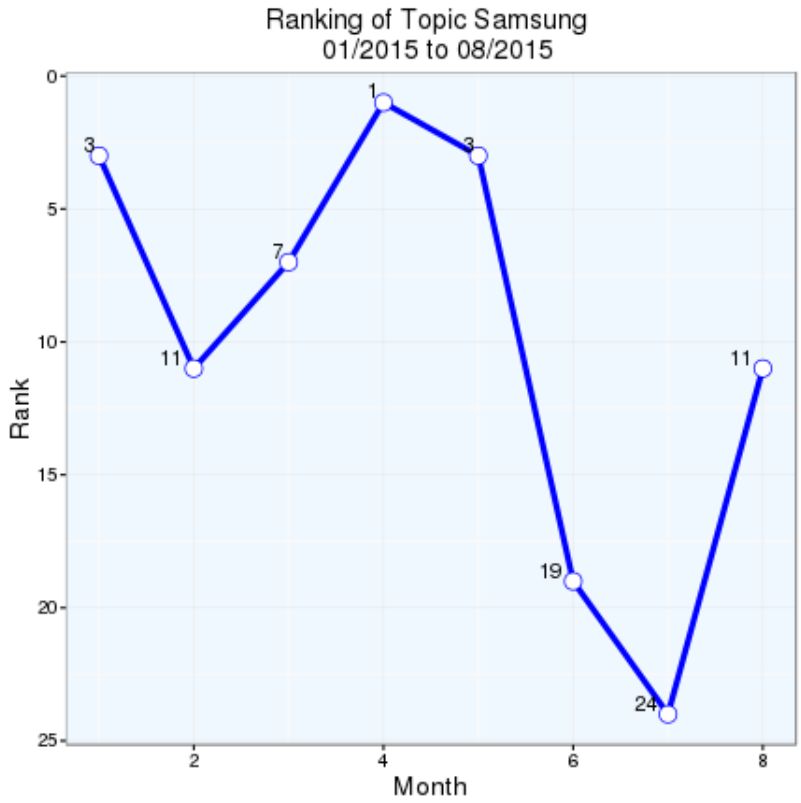

Figure 8: Topic Samsung in 2015

sense that 'Samsung' was the most discussed topic of April since Samsung Galaxy S6 was released on April 10. The popularity of topic Samsung declined after that.

5 Conclusion and Future Work

I have summarized the methodology of LDA. I have demonstrated the applications of LDA to Reddit Technology text data in R environment including topic discovery, visualization, monitoring, and prediction. For my analysis, collapsed Gibbs sampling has shown better result than VEM. R has a comprehensive collection of packages along with topicmodels that can perform these smoothly. The most valuable application of LDA is perhaps automatically extracting information from a large collection of documents without human efforts.

I have identified several questions for further analysis: How do the methods of model estimation perform under specific conditions such as different length and quantity of the documents? How do other methods of model selection perform compared to Perplexity? How does Correlated Topic Model, a method that does not require the number of topics for

model estimation, perform compared to LDA?

References

Blei DM, Lafferty JD (2009). "Topic Models." In A Srivastava, M Sahami (eds.), *Text Mining: Classification, Clustering, and Applications.* Chapman Hall/CRC Press.

Blei DM, Ng AY, Jordan MI (2003). "Latent Dirichlet Allocation." *Journal of Machine Learning Research,* **3**, 993-1022.

Griffiths TL, Steyvers M (2004). "Finding Scientific Topics." *Proceedings of the National Academy of Sciences of the United States of America,* **101**, 5228-5235.

R Core Development Team (2015). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Steyvers M, Griffiths TL (2007). "Probabilistic topic models." In TK Landauer, DS McNamara, S Dennis, W Kintsch (eds.) *Handbook of Latent Semantic Analysis: A Road to Meaning.* Lawrence Erlbaum Associates.

Appendix: Sample Source Code Listings

```
library(topicmodels)
library(tm)
library(ggplot2)
library(quanteda)
library(wordcloud)
library(RColorBrewer)

# Pre-processing
myCorpus <- corpus(textfile("technology201501.csv",
    textField = "title"))
myDfm <- dfm(myCorpus, ignoredFeatures = stopwords("english"),
    stem = TRUE, removeNumbers = TRUE, removePunct = TRUE,
    removeSeparators = TRUE)
myDfm <- removeFeatures(myDfm, c("reddit", "redditors",
    "redditor", "nsfw", "hey", "vs", "versus",
    "ur", "they'r", "u'll", "u.", "u", "r", "can",
    "anyone", "will", "amp", "http", "just"))
sparsityThreshold <- round(ndoc(myDfm) * (1 -
    0.99999))
myDfm2 <- trim(myDfm, minDoc = sparsityThreshold)
myDtm <- quantedaformat2dtm(myDfm2)
save(myDfm2, file = "dfm1.RData")
save(myDtm, file = "dtm1.RData")
nfeature(myDfm2)
dim(myDfm2)

# Create training set and test set
set.seed(42)
samp <- sample(ndoc(myDfm2), 9000, replace = FALSE)
myDfm.train <- myDfm2[samp, ]
myDfm.test <- myDfm2[-samp, ]
save(myDfm.train, file = "dfmtrain.RData")
save(myDfm.test, file = "dfmtest.RData")

# Gibbs with different k's
best.model0.Gibbs <- lapply(seq(5, 55, by = 10),
    function(k) {
        LDA(quantedaformat2dtm(myDfm.train), k,
            method = "Gibbs", list(seed = 123))
    })
save(best.model0.Gibbs, file = "LDAGibbs-by10.RData")
```

```
best.model.perp0.Gibbs <- as.data.frame(as.matrix(lapply(best.model0,
    perplexity, quantedaformat2dtm(myDfm.test))))
best.model.perp.df0.Gibbs <- data.frame(topics = c(seq(5,
    55, by = 10)), P = as.numeric(as.matrix(best.model.perp0.Gibbs)))
ggplot(best.model.perp.df0.Gibbs, aes(x = topics,
    y = P)) + xlab("Number of topics") + ylab("Perplexity") +
    geom_line() + theme_bw() + theme(axis.title.x = element_text(vjust = -0.25,
    size = 14)) + theme(axis.title.y = element_text(size = 14,
    angle = 90)) + ggtitle("Perplexity of Gibbs models")

# VEM with different k's
best.model0.vem <- lapply(seq(5, 55, by = 10),
    function(k) {
        LDA(quantedaformat2dtm(myDfm.train), k,
            method = "VEM", list(seed = 123))
    })
save(best.model0.vem, file = "LDAVEM-by10.RData")
best.model.perp0.vem <- as.data.frame(as.matrix(lapply(best.model0.vem,
    perplexity, quantedaformat2dtm(myDfm.test))))
best.model.perp.df0.vem <- data.frame(topics = c(seq(5,
    55, by = 10)), P = as.numeric(as.matrix(best.model.perp0.vem)))
ggplot(best.model.perp.df0.vem, aes(x = topics,
    y = P)) + xlab("Number of topics") + ylab("Perplexity") +
    geom_line() + theme_bw() + theme(axis.title.x = element_text(vjust = -0.25,
    size = 14)) + theme(axis.title.y = element_text(size = 14,
    angle = 90)) + ggtitle("Perplexity of VEM models")

# Model Selection Plots
png(filename = "ldaperpGibbsVEM0.png")
require(gridExtra)
plot1 <- ggplot(best.model.perp.df0.Gibbs, aes(x = topics,
    y = P)) + xlab("Number of topics") + ylab("Perplexity") +
    geom_line() + theme_bw() + theme(axis.title.x = element_text(vjust = -0.25,
    size = 14)) + theme(axis.title.y = element_text(size = 14,
    angle = 90)) + ggtitle("Perplexity of Gibbs models")
plot2 <- ggplot(best.model.perp.df0.vem, aes(x = topics,
    y = P)) + xlab("Number of topics") + ylab("Perplexity") +
    geom_line() + theme_bw() + theme(axis.title.x = element_text(vjust = -0.25,
    size = 14)) + theme(axis.title.y = element_text(size = 14,
    angle = 90)) + ggtitle("Perplexity of VEM models")
grid.arrange(plot1, plot2, ncol = 2, heights = c(0.5,
    0.5), widths = c(0.485, 0.515))
dev.off()
```

```
# Model Fitting with Gibbs for January
LDAModel1 <- LDA(quantedaformat2dtm(myDfm2), 25,
    "Gibbs", list(iter = 4000, burnin = 2000,
        seed = 123))
save(LDAModel1, file = "LDAGibbs-01-25topics.RData")
terms(LDAModel1, 10)

# Topic Proportion
load("LDAGibbs-01-25topics.RData")
theta.df <- posterior(LDAModel1)$topics
theta <- as.data.frame(cbind(1:25, apply(theta.df,
    2, mean)))
theta <- theta[order(-theta$V2), ]
df = data.frame(table(topics(LDAModel1)))
df = df[order(-df$Freq), ]
theta$V1 <- as.character(theta$V1)
theta$V1 <- factor(theta$V1, levels = theta$V1)
png(filename = "topicrank1bar.png")
ggplot(theta, aes(x = theta[, 1], y = theta[,
    2])) + geom_bar(stat = "identity", fill = "lightgreen") +
    coord_cartesian(ylim = c(0.0397, 0.0404)) +
    xlab("Topic") + ylab("Topic Proportion") +
    ggtitle("Popularity by Topic in January")
dev.off()

# Wordcloud
terms <- as.data.frame(t(posterior(LDAModel1)$terms))
df <- data.frame(freq = terms[, 21] * 10000, word = rownames(terms))
pal <- brewer.pal(8, "Dark2")
png("wordcloud_topic21.png", width = 800, height = 800)
wordcloud(df$word, df$freq, scale = c(7, 0.5),
    min.freq = 8, max.words = Inf, random.order = FALSE,
    rot.per = 0.15, colors = pal)
dev.off()

# Prediction
newCorpus <- corpus(textfile("technology201503.csv",
    textField = "title"))
newDfm <- dfm(newCorpus, ignoredFeatures = stopwords("english"),
    stem = TRUE, removeNumbers = TRUE, removePunct = TRUE,
    removeSeparators = TRUE)
newDfm <- removeFeatures(newDfm, c("reddit", "redditors",
```

```
    "redditor", "nsfw", "hey", "vs", "versus",
    "ur", "they'r", "u'll", "u.", "u", "r", "can",
    "anyone", "will", "amp", "http", "just"))
sparsityThreshold <- round(ndoc(newDfm) * (1 -
    0.99999))
newDfm2 <- trim(newDfm, minDoc = sparsityThreshold)
newDfm.pred <- newDfm2[50, ]
pred <- posterior(LDAModel1, newDfm.pred)$topics
pred.t <- as.data.frame(t(pred))
pred.t <- cbind(topic = factor(rownames(pred.t)),
    pred.t)
pred.o <- pred.t[order(-pred.t[, 2]), ]
pred.o[, 1] <- as.character(pred.o[, 1])
pred.o[, 1] <- factor(pred.o[, 1], levels = pred.o[,
    1])
png(filename = "predict.png")
ggplot(pred.o, aes(x = pred.o[, 1], y = pred.o[,
    2])) + geom_bar(stat = "identity", fill = "lightgreen") +
    coord_cartesian(ylim = c(0.03, 0.11)) + xlab("Topic") +
    ylab("Topic Probability") + ggtitle("Prediction of a Samsung Document")
dev.off()

t201503 <- read.csv("technology201503.csv", stringsAsFactors = F)
t201503[50, 1]

# Monitoring of topic Samsung
library(grid)
df <- data.frame(rank = c(3, 11, 7, 1, 3, 19,
    24, 11), month = c(1:8))
png("samsung.png")
ggplot(df, aes(x = month, y = rank)) + xlab("Month") +
    ylab("Rank") + geom_line(colour = "blue",
    linetype = "solid", size = 1.5) + theme_bw() +
    theme(axis.title.x = element_text(vjust = -0.25,
        size = 14)) + theme(axis.title.y = element_text(size = 14,
    angle = 90)) + scale_y_continuous(trans = "reverse") +
    geom_point(colour = "blue", size = 4, shape = 21,
        fill = "white") + theme(panel.background = element_rect(fill = "aliceblue")) +
    ggtitle("Ranking of Topic Samsung \n  01/2015 to 08/2015") +
    theme(plot.margin = unit(c(1, 1, 1, 1), "cm")) +
    geom_text(aes(label = rank), hjust = 1.5,
        vjust = -0.25)
dev.off()
```