Creative Component

Taikgun Song

Abstract

Write abstract. It is known that the performance of Laten Dirichlet Allocation based topic models over short texts. In this paper, we would like to compare LDA methods with different 'parameter' under experimental setting.

1 Introduction

Then the following `R` packages were utilized to conduct `Latent Dirichlet Allocation`(Blei, Ng, and Jordan 2003) method: `openNLP`(Hornik 2016a), `NLP` (Hornik 2016b), `topicmodels` (Bettina Grün 2016).

2 Data Gathering (Need revising)

In order to compare different LDA processes, a collection of short user-generated online reviews from a popular website, the Trip Adviser, was used for evaluation. Among all restaurants in Honolulu, Hawaii registered on the Trip Adviser, the latest 10 reviews were scraped and read into `R` (R Core Team 2014) using the `rvest` (Hadley Wickham [aut 2016) package. This dataset includes the following information of the 7700 Honolulu restaurants: restaurant name, number total reviews, average star rating, individual review title, individual review entry, individual star rating, and the date visited. In this paper, we are particularly interested in individual review entry.

2.1 Procedure 1: Obtaining the page numbers for all restaurants

The following URL: "https://www.tripadvisor.com/Restaurants-g60982-Honolulu_Oahu_Hawaii.html" in Figure 1 leads to all 1745 restaurants registered on TripAdvisor (As of October 27th, 2016).

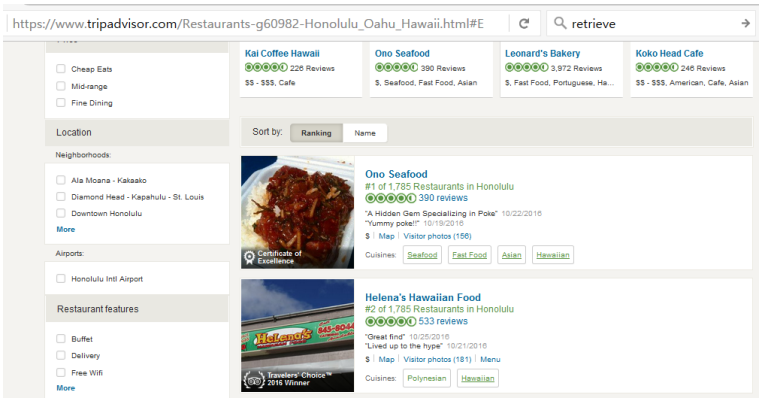

Figure 1: Example image of the initial TripAdvisor webpage for all restaurants in Honolulu, Hawaii

Figure 2: Example pages number image of the restaurants in Honolulu, Hawaii

These restaurants are listed into 59 different pages as it is shown in Figure 2. Therefore, obtaining URL of all 59 pages would be the first step to scrape each individual restaurant reviews. Note that there are 30 restaurants in each page except for the last page. The following image of Figure 3 portraits how the html code is written for this web page.

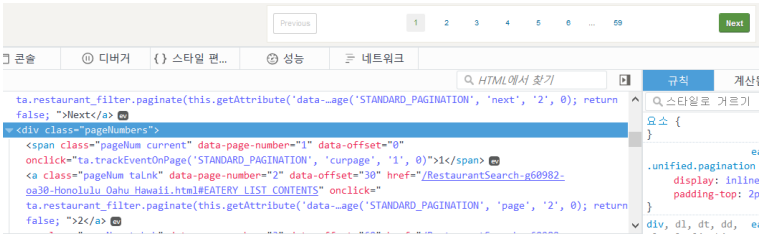

Figure 3: Image of the source code

2.1.1 Step 1: Read html using the 'rvest' package

The following steps R code using the `rvest` (Hadley Wickham [aut 2016) package will scrape 59 URL address for all restaurants in Honolulu, Hawaii.

```
initial.url="https://www.tripadvisor.com/Restaurants-g60982-Honolulu_Oahu_Hawaii.html"
doc=read_html(initial.url)
```

2.1.2 Step 2: Scrape the last page number of the review pages.

Note that, from Figure 3, the page numbers are under the node `//div [\@class="pageNumbers"]`. Use the `html_nodes` function in conjunction with specific node using the `xpath` in the `revst` and `XML` package respectively to obtain the information under the specific node. Then, apply the `html_text` function to obtain text information. Lastly, since the page numbers are in between `\\n` patterns, employ gsub function to retrieve the last page number: 59.

```
pg.num.path=c(pg.url = '//div [@class="pageNumbers"]')
check=lapply(pg.num.path, function(x) html_nodes(doc,xpath=x))
f.page=check$pg.url %>% html_text() %>% gsub(".*\n(.*)\n", "\\1", .)
```

2.1.3 Step 3: Generate urls for each review page.

The general URL for these pages are as the following: "https://www.tripadvisor.com/Restaurants-g60982-oa SUB -Honolulu_Oahu_Hawaii.html#EATERY_LIST_CONTENTS". There are 30 restaurants in each page, and the 1^{st} restaurant that appears in n^{th} page is actually the $(30 * (n-1)+1)^{th}$ restaurant. Therefore, changing the SUB in the general URL with gsub function will allow us to create a list of all page links.

```
complete.initial.url=rbind.data.frame(rest.url=initial.url, data.frame(rest.url=sapply(c(1:(as.numeric(f.page)
row.names(complete.initial.url)=NULL
```

2.1.4 Step 4: Combine the all the above functions

The previous R codes were combined into a single function named `get.pg.num`.

```
get.pg.num=function(intial.url){
  doc1=read_html(initial.url)
  pg.num.path=c(pg.url = '//div [@class="pageNumbers"]')
  check=lapply(pg.num.path, function(x) html_nodes(doc1,xpath=x))
  f.page=check$pg.url %>% html_text() %>% gsub(".*\n(.*)\n", "\\1", .)
  complete.initial.url=rbind.data.frame(rest.url=initial.url, data.frame(rest.url=sapply(c(1:(as.numeric(f.pag
  row.names(complete.initial.url)=NULL
  return(complete.initial.url[[1]])
}
```

2.2 Procedure 2: Obtaining url for individual restaurants.

Once the URL for each pages are retrieved, the next step is to get URL for all restaurants within each pages. As shown in Figure 4, the title of the restaurant is embedded in the html node `//div [\@id="EATERY_SEARCH_RESULTS"]//h3 [\@class="title"]` and its corresponding URL is embedded in `//div [\@id="EATERY_SEARCH_RESULTS"]//h3 [\@class="title"] //a [\@class="property_title"]`.

```
doc=read_html(pg.num.url)
rest.path=c(ea.url = '//div [@id="EATERY_SEARCH_RESULTS"]//h3 [@class="title"] //a [@class="property_title"]',
apply_rest.path=lapply(rest.path, function(x) html_nodes(doc,xpath=x))
```

Thus, the title of a restaurant and its URL is obtained by running the above R code. Simple cleaning Code below would extract the title and the URL of each restaurant.

```
r_url.r_name=cbind.data.frame(rest.url=gsub(".*?href=\"(.*?)\".*", "https://www.tripadvisor.com\\1",apply_rest
```

As a result, the above two codes were made into a function `rest.url`

Figure 4: Example source code image of each restaurant URL

```
rest.url=function(pg.num.url){
  doc=read_html(pg.num.url)
  rest.path=c(ea.url = '//div [@id="EATERY_SEARCH_RESULTS"]//h3 [@class="title"] //a [@class="property_title"]
  apply_rest.path=lapply(rest.path, function(x) html_nodes(doc,xpath=x))
  r_url.r_name=cbind.data.frame(rest.url=gsub(".*?href=\"(.*?)\".*", "https://www.tripadvisor.com\\1",apply_re
  return(r_url.r_name)
}
```

2.3 Procedure 3: Obtaining individual review information for all restaurants.

After the second procedure, URL for each restaurants in Honolulu Hawaii that is registered in TripAdvisor was scraped. These URL provide access to a list of individual review entries of that particular given restaurant. For long individual reviews, however, full entry is not directly available from the URL generated in Procedure 2. For example, individual review illustrated in Figure 5, ellipsis symbol ... was used to omit reviews after `Black Sand`. Therefore, further scraping is required to retrieve full review entry. The following steps were taken.

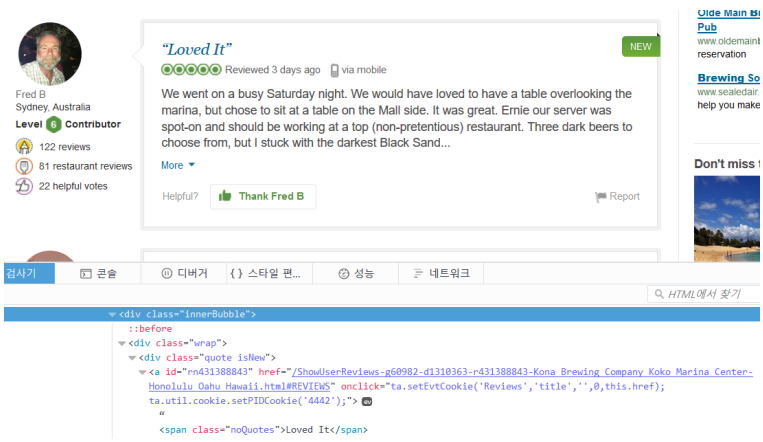

Figure 5: Example source code image of individual review URL

2.3.1 Step 1: Reading a given restaurant url.

As the default behavior, the `read_html` function does not properly identify itself to the server that it is retrieving the contents from. Hence, for a long process such as Procedure 3, the `useragent` should be used to the `handle` argument of `read_html` function using `curl` in order for the scraper to identify itself. Otherwise, an error message such as the following will be generated after some iteration: `Error in open.connection(x, "rb")`

```
doc=curl(ea.rest.url, handle = new_handle("useragent" = "Mozilla/5.0")) %>%  read_html()
```

2.3.2 Step 2: Scraping the title of an individual review and its url.

Figure 5 illustrates that the title of the restaurant is embedded in the html node `//div [\@id="REVIEWS"]//div [\@class="innerBubble"]//span [\@class="noQuotes"]` and its corresponding URL is embedded in `//div [\@id="REVIEWS"]//di`

[\@class="innerBubble"]. The code below will generate URL for individual reviews after further cleaning with gsub function.

```
u.n.t.path=c(ea.url = '//div [@id="REVIEWS"]//div [@class="innerBubble"]',ea.title= '//div [@id="REVIEWS"]//di
apply_u.n.t.path=lapply(u.n.t.path, function(x) html_nodes(doc,xpath=x))
u.n.t=cbind.data.frame(ea.url=gsub(".*?<a href=\"(.*?)\".*", "https://www.tripadvisor.com\\1",apply_u.n.t.path
```

2.3.3 Step 3: Scraping the review entry of an individual review.

Figure 6: Example source code image of individual review URL

The source code and its output is presented in Figure 6. It is clear that the review entry is embedded in the html node (//div [@id="REVIEWS"]//div [@class="innerBubble"]//p)[1]. Thus, the code below was used to capture the full review entry.

```
r.path <- c(ea.review = '(//div [@id="REVIEWS"]//div [@class="innerBubble"]//p)[1]')
apply_r.path=lapply(r.path, function(x) sapply(u.n.t[,1], function(y) html_nodes(y %>% read_html(),xpath=x) %>'
u.t.r.out.put=cbind.data.frame(u.n.t, apply_r.path, stringsAsFactors=F)
```

2.3.4 Step 4: Dealing with restaurants with no review.

There are multiple restaurants without any review. These restaurants will generate error messages and should be dealt by using a simple if loop. If there is no review entry, then the u.t.r.out.put from the previous code will have zero number of rows. Therefore, re-define u.t.r.out.put with all column entry having No review posted with the number of rows equal to zero as the code provided below.

```
if (nrow(u.t.r.out.put)!=0){
  row.names(u.t.r.out.put)=c(1:nrow(u.n.t))
} else {u.t.r.out.put=cbind.data.frame(ea.url="No review posted", ea.title="No review posted", ea.review ="No
```

2.3.5 Step 5: Combine all steps in procedure 3.

The R codes in Procedure 3 were combined into a single function named review.data. The function Sys.sleep(2) allow computer to rest for 2 seconds. Some unstable servers crash when the number of requests exceed the server's capability. Therefore, Sys.sleep was introduced to avoid crashing the server.

```
review.data=function(ea.rest.url){
  doc=curl(ea.rest.url, handle = new_handle("useragent" = "Mozilla/5.0")) %>%  read_html()
  u.n.t.path=c(ea.url = '//div [@id="REVIEWS"]//div [@class="innerBubble"]',ea.title= '//div [@id="REVIEWS"]//
  apply_u.n.t.path=lapply(u.n.t.path, function(x) html_nodes(doc,xpath=x))
  u.n.t=cbind.data.frame(ea.url=gsub(".*?<a href=\"(.*?)\".*", "https://www.tripadvisor.com\\1",apply_u.n.t.pa
  r.path <- c(ea.review = '(//div [@id="REVIEWS"]//div [@class="innerBubble"]//p)[1]')
  apply_r.path=lapply(r.path, function(x) sapply(u.n.t[,1], function(y) html_nodes(y %>% read_html(),xpath=x) '
  u.t.r.out.put=cbind.data.frame(u.n.t, apply_r.path, stringsAsFactors=F)
  if (nrow(u.t.r.out.put)!=0){
    row.names(u.t.r.out.put)=c(1:nrow(u.n.t))
```

```
  } else {u.t.r.out.put=cbind.data.frame(ea.url="No review posted", ea.title="No review posted", ea.review ="N
  return(u.t.r.out.put)
  Sys.sleep(2)
}
```

2.4 Procedure 4: Combining the previous procedures to construct full dataset.

```
# Procedure 1
initial.url="https://www.tripadvisor.com/Restaurants-g60982-Honolulu_Oahu_Hawaii.html"
source("Code/pg_num_url.r")
all.url=get.pg.num(intial.url)

# Procedure 2
source("Code/ind_rest_url.r")
ea.url=lapply(all.url, function(x) rest.url(x)) %>% do.call("rbind",.)

# Procedure 3
source("Code/ind_url_title_review.r")
fin=nrow(ea.url)

# Final Product
source("Code/review_data_w_pb.r")
final.data=lapply(1:nrow(ea.url), function(x) review.data.w.pb(x) %>% cbind(rep(ea.url[x,2], nrow(.)), .)) %>%
colnames(final.data)[1]="rest.name"
#View(final.data)
```

The above code combines all the previous procedures and generates the complete dataset of all restaurants in Honolulu, Hawaii with following information: Restaurant Name, URL, title, and entry of individual reviews. The combined procedure takes time, thus simple progression bar was introduced to monitor the progress. The code for the progression bar is shown below.

```
pb <- winProgressBar(title = "progress bar", min = 0, max = fin, width = 300)
review.data.w.pb=function(x){
  setWinProgressBar(pb, x, title=paste(round(x/fin*100, 0), "% done"))
  return(review.data(ea.url[x,1]))
}
```

	rest.name	ea.url	ea.title	ea.review
1	Ono Seafood	https://www.tripadvisor.com/ShowUserReviews-g6…	A Hidden Gem Specializing in Poke	This tiny, laid back place is a favorite of the locals. In…
2	Ono Seafood	https://www.tripadvisor.com/ShowUserReviews-g6…	Yummy poke!!	Visiting Hawaii for the first time and based on review…
3	Ono Seafood	https://www.tripadvisor.com/ShowUserReviews-g6…	First time to taste that kind of sushi rice!	I love sushi and have been eating many kinds of sus…
4	Ono Seafood	https://www.tripadvisor.com/ShowUserReviews-g6…	Close. Cheap. Delicious.	If you've got yourself a rental, it's not far at all from W…
5	Ono Seafood	https://www.tripadvisor.com/ShowUserReviews-g6…	A-MAZ-ING · On par or better with 40-70 dollars dishes	My wife and I walked to Ono Seafood fron Honolulu. I…
6	Ono Seafood	https://www.tripadvisor.com/ShowUserReviews-g6…	Good authentic Hawaiian food.	Being away from the islands, I wanted good food I ha…
10068	Bon Bon Cafe	No review posted	No review posted	No review posted
10069	Orange Grove Frozen Yogurt	No review posted	No review posted	No review posted
10070	Tokiwaya	No review posted	No review posted	No review posted
10071	The Kona Bean Cafe	No review posted	No review posted	No review posted
10072	Tokyo Hotdogs	No review posted	No review posted	No review posted
10073	Morning Hill Foods LLC	No review posted	No review posted	No review posted

Figure 7: First 10 and the last 10 entries of the final dataset

The head and the tail of the constructed dataset is given in Figure 7.

2.5 Procedure 5: Clean data entries for further process

The raw data scraped from the web are noisy and preparation process is necessary to minimize this noise. The initial step for this cleaning process is to remove all non-Latin characters, and any other unnecessary words or characters present in the data set collected in Procedure 4.

2.5.1 Step 1: Removing reviews with non-latin characters

	rest.name	ea.url	ea.title	ea.review
1168	Ramen Nakamura	https://www.tripadvisor.com/ShowUserReviews-g6...	불란궁, 돼지 냄새에 찐 음식	일본 라멘의 팬이라서 라멘을 즐겨 먹는다. 하지만 여기 미...
7858	Joe Hyung Restaurant	https://www.tripadvisor.com/ShowUserReviews-g6...	チップのばったくり注意 - Overcharged Tip Without An...	日本語をしゃべれる韓<U+56FD>人のおばさんがいます...
3471	Kuhio Beach Grill	https://www.tripadvisor.com/ShowUserReviews-g6...	なかなか良い	午後8時、<U+5E30><U+56FD>前後なので移動や待つの...

Figure 8: Example of reviews written in non-Latin characters

TripAdvisor is a globally well known website and there are reviews that are written in non-Latin characters as shown in Figure 8. These non-Latin characters could be deleted by the following code.

	rest.name	ea.url	ea.title	ea.review
1168	Ramen Nakamura	https://www.tripadvisor.com/ShowUserReviews-g6...	,	. . I am a fan of japanese style ramen. But this is too s...
7858	Joe Hyung Restaurant	https://www.tripadvisor.com/ShowUserReviews-g6...	- Overcharged Tip Without Any Note	<U+56FD><U+30FC><U+30FC>2015<U+6570>6118...
3471	Kuhio Beach Grill	https://www.tripadvisor.com/ShowUserReviews-g6...		8<U+5E30><U+56FD><U+5185><U+6765>4 <U+4F1...

Figure 9: Example of deleting reviews written in non-Latin characters

The result of deleting non-Latin characters could be found in Figure 9.

2.5.2 Step 2: Removing other unnecessary words and symbols

Although non-Latin characters are deleted, there are remaining words and symbols remaining in the data set. Unicode such as in Figure 9 is an example. These words and symbols should be treated using gsub functions,

2.5.3 Step 3: Removing restaurants with no review from the data set

As illustrated in Figure 7, Some of the restaurants on the TripAdvisor webpage has no reviews. Moreover, cleaning processes such as step 1 and 2 may remove the entire review entry for certain reviews blank. Therefore, these reviews should be removed from the data set by running a simple function as the following.

	rest.name	ea.url	ea.title	ea.review
1	Ono Seafood	https://www.tripadvisor.com/ShowUserReviews-g6...	A Hidden Gem Specializing in Poke	This tiny, laid back place is a favorite of the locals. In...
2	Ono Seafood	https://www.tripadvisor.com/ShowUserReviews-g6...	Yummy poke!!	Visiting Hawaii for the first time and based on review...
3	Ono Seafood	https://www.tripadvisor.com/ShowUserReviews-g6...	First time to taste that kind of sushi rice!	I love sushi and have been eating many kinds of sus...
4	Ono Seafood	https://www.tripadvisor.com/ShowUserReviews-g6...	Close. Cheap. Delicious.	If you've got yourself a rental, it's not far at all from W...
5	Ono Seafood	https://www.tripadvisor.com/ShowUserReviews-g6...	A-MAZ-ING - On par or better with 40-70 dollars dishes	My wife and I walked to Ono Seafood fron Honolulu. I...
6	Ono Seafood	https://www.tripadvisor.com/ShowUserReviews-g6...	Good authentic Hawaiian food.	Being away from the islands, I wanted good food I ha...
9798	Pizza Hut	https://www.tripadvisor.com/ShowUserReviews-g6...	This restaurant was closed over 15 years ago	If I remember this correctly, this location was a Pizza ...
9799	Konos	https://www.tripadvisor.com/ShowUserReviews-g6...	Fantastic food!	This place specializes in kalua pig, which is roasted f...
9800	Giovanni's Shrimp Truck	https://www.tripadvisor.com/ShowUserReviews-g6...	OVERHYPED!	All my friends and Yelp recommended this place and ...
9801	Giovanni's Shrimp Truck	https://www.tripadvisor.com/ShowUserReviews-g6...	Terrible -Horrible food truck	This food truck been recommended by Yelp and few ...
9802	Paradise Seafood Market & Restaurant	https://www.tripadvisor.com/ShowUserReviews-g6...	Live Dungeness Crab, Lobsters and Oysters	My 1st time to this seafood market/restaurant I was s...
9803	Jamba Juice	https://www.tripadvisor.com/ShowUserReviews-g6...	What refreshment!	Shopping at Ala Moana is tough to hang around unde...

Figure 10: Head and tail part of the data set after under going the cleaning procedure

As the result of Procedure 5, final data set was obtained as illustrated in Figure 10.

3 Data Processing

Inspite of applying preparation processes such as the cleaning procedure during the Data gathering phase, the following techniques could be put to use in order to enhance the accuracy of the text-mining analysis.

3.1 Removing Stop Words

Stop words are a group of natural language words that are essential in construction of grammatical structure of English. For example, words such as “a”, “an”, “the”, “that”, “these”, “my”, “his”, “most”, “with”, and *et cetera* is an example of stop words. Although these words are crucial elements of English structure, however, these stop words are less important when it comes to the purpose of delivering the meanings. Also, high frequency and abundance nature of these stop words makes little value in helping text mining techniques, and thus be filtered out. In this paper, stop words were deleted by utilizing the `tm` (Ingo Feinerer 2015) and the `RTextTools` (Timothy P. Jurka 2014) packages in R using the following code.

```
vdc=VectorSource(text)
my.corpus = Corpus(vdc)
my.corpus_copy = my.corpus
my.corpus = tm_map(my.corpus, removeWords, c("the", stopwords("english")))
```

An example result of using the above code is as follows.

```
eg="This is an example of removing the stop words."
eg_vdc=VectorSource(eg)
eg_my.corpus = Corpus(eg_vdc)
eg_my.corpus = tm_map(eg_my.corpus, removeWords, c("the", stopwords("english")))
strwrap(eg_my.corpus[[1]])
```

```
## [1] "This example removing stop words."
```

Note that stop words such as “is”, “an”, “the”, “of” are removed from the given sentence.

3.2 Stemming

Due to structural and grammatical reasons of English, a family of words that are driven from a single root word is used in different forms. For example, words such as “stems”, “stemmer”, “stemming”, and “stemmed” are all based on a root word “stem”. These different form of words with the same root word may cause difficulties applying some text mining techniques. Therefore, stemming procedure should be introduced to reduce inflectional forms of a word to a word in its root form. The `tm` package is again used for the stemming process and its code is given as the following.

```
vdc=VectorSource(text)
my.corpus = Corpus(vdc)
my.corpus = tm_map(my.corpus, stemDocument, language="english")
```

```
eg="This is an example of stemming process where words are stemmed into its root form."
eg_vdc=VectorSource(eg)
eg_my.corpus = Corpus(eg_vdc)
eg_my.corpus = tm_map(eg_my.corpus, stemDocument, language="english")
strwrap(eg_my.corpus[[1]])
```

```
## [1] "This is an exampl of stem process where word are stem into it root"
## [2] "form."
```

Note that words “exmple”, “stemming”, “stemmed”, “words”, and “its” were reduced to “exampl”, “stem”“,”word“, and”it“.

3.3 Stop words and Stemming procedure combined

A simple function as below could combine the two processes above together.

```
StopNStem=function(text){
  vdc=VectorSource(text)
  my.corpus = Corpus(vdc)
```

```
  my.corpus_copy = my.corpus
  my.corpus = tm_map(my.corpus, removeWords, c("the", stopwords("english")))
  my.corpus = tm_map(my.corpus, stemDocument, language="english")
  return(paste(strwrap(my.corpus[[1]]), sep="", collapse=""))
}
```

```
StopNStem("This is an example of removing the stop words. This is an example of stemming process where words a
```

```
## [1] "This exampl remov stop words. This exampl stem process word stemroot form."
```

```
setwd("C:/Users/Taikgun Song/Desktop/CC Meeting/CC")
Final=read.csv("data/Final.csv", stringsAsFactors = F)
ea.review.SWS=lapply(Final$ea.review, StopNStem) %>% do.call("rbind",.) %>% rbind.data.frame(stringsAsFactors=
View(ea.review.SWS)
write.csv(ea.review.SWS, "ea_review_SWS.csv", row.names = F)
```

- LDA, documents are represent as random mixtures over latent topics, where each topic is characterized by a distribution over words. (Need paraphrasing)

LDA assumes that words are generated by topics. That is, word distribution

$$p(w|\theta, \beta)$$

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

Therefore, retrieving information by reducing the inflected words to its original word stem

may increase the probability of the joint distribution of topic mixture leading (thus increasing the probability of a document and a corpus).

	Topic.1	Topic.2	Topic.3	Topic.4
1	food	food	good	food
2	service	good	place	great
3	good	place	food	good
4	restaurant	like	great	place
5	just	just	also	service
6	one	one	chicken	get
7	ordered	service	just	one
8	back	get	ordered	restaurant
9	like	restaurant	really	can
10	really	will	get	best

(a) Table LDA output before stemming

	Topic.1	Topic.2	Topic.3	Topic.4
1	good	food	good	food
2	food	place	food	good
3	place	good	order	great
4	great	restaur	servic	restaur
5	get	great	place	place
6	breakfast	servic	one	delici
7	servic	order	just	tri
8	time	price	great	like
9	price	eat	time	one
10	wait	time	like	love

(b) Table LDA output after stemming

Table 1: Difference of LDA output between with and without stop words

4 Sequential bigram

Wallach

References

Bettina Grün, Kurt Hornik. 2016. *Topic Models*. https://cran.r-project.org/web/packages/topicmodels/topicmodels.pdf.

Blei, David M, Andrew Y Ng, and Michael I Jordan. 2003. "Latent Dirichlet Allocation." *Journal of Machine Learning Research* 3 (Jan): 993–1022.

Hadley Wickham [aut, RStudio [cph], cre]. 2016. *Easily Harvest (Scrape) Web Pages*. https://cran.r-project.org/web/packages/rvest/rvest.pdf.

Hornik, Kurt. 2016a. *Apache Opennlp Tools Interface*. https://cran.r-project.org/web/packages/openNLP/openNLP.pdf.

———. 2016b. *Natural Language Processing Infrastructure*. https://cran.r-project.org/web/packages/NLP/NLP.pdf.

Ingo Feinerer, Artifex Software, Kurt Hornik. 2015. *Text Mining Package*. https://cran.r-project.org/web/packages/tm/tm.pdf.

R Core Team. 2014. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. http://www.R-project.org/.

Timothy P. Jurka, Amber E. Boydstun, Loren Collingwood. 2014. *Automatic Text Classification via Supervised Learning*. https://cran.r-project.org/web/packages/RTextTools/RTextTools.pdf.