Homework 3

Victoria Xie

2022-04-27

Question 1

(a)

In this case, I would choose variational inference for fitting LDA, given that the corpus of text is very large. Even though EM can be efficient, either of the "E" step or the "M" step can be too hard in practice, and with variational inference we can have a looser lower bound that is better suited for a large corpus and works faster.

(b)

1. Compute symmetric KL-Divergence of salient distributions that are derived from matrix factors split from a corpus, and then select the 'dip' of the divergence values as the optimal number of topics.
2. Compute the topic density of each topic, and find the most unstable topics under the old structure, and iteratively update the parameter K until the model is stable.
3. Used the Gibbs sampling algorithm to obtain samples from the posterior distribution over z at several choices of number of topics and evaluate the consequences.

(c)

i) α is the likely cause of this because it controls the prior distribution over topic weights in each document.

ii) η is the likely cause of this because it controls the prior distribution over word weights in each topic.

(d)

I would choose Gibbs Sampler for fitting the LDA model, because it works with full conditional distributions. It is not as efficient as variational inference but runtime is not a concern here.

(e)

$p(w|d) = \frac{V_d^T U_w}{\sum_{w \in V} V_d^T U_w}$, where V_d is the document embedding matrix, and U_w is the word embedding matrix. So no, the words are not independent of each other after conditioning on the topic, because they are still dependent on the documents.

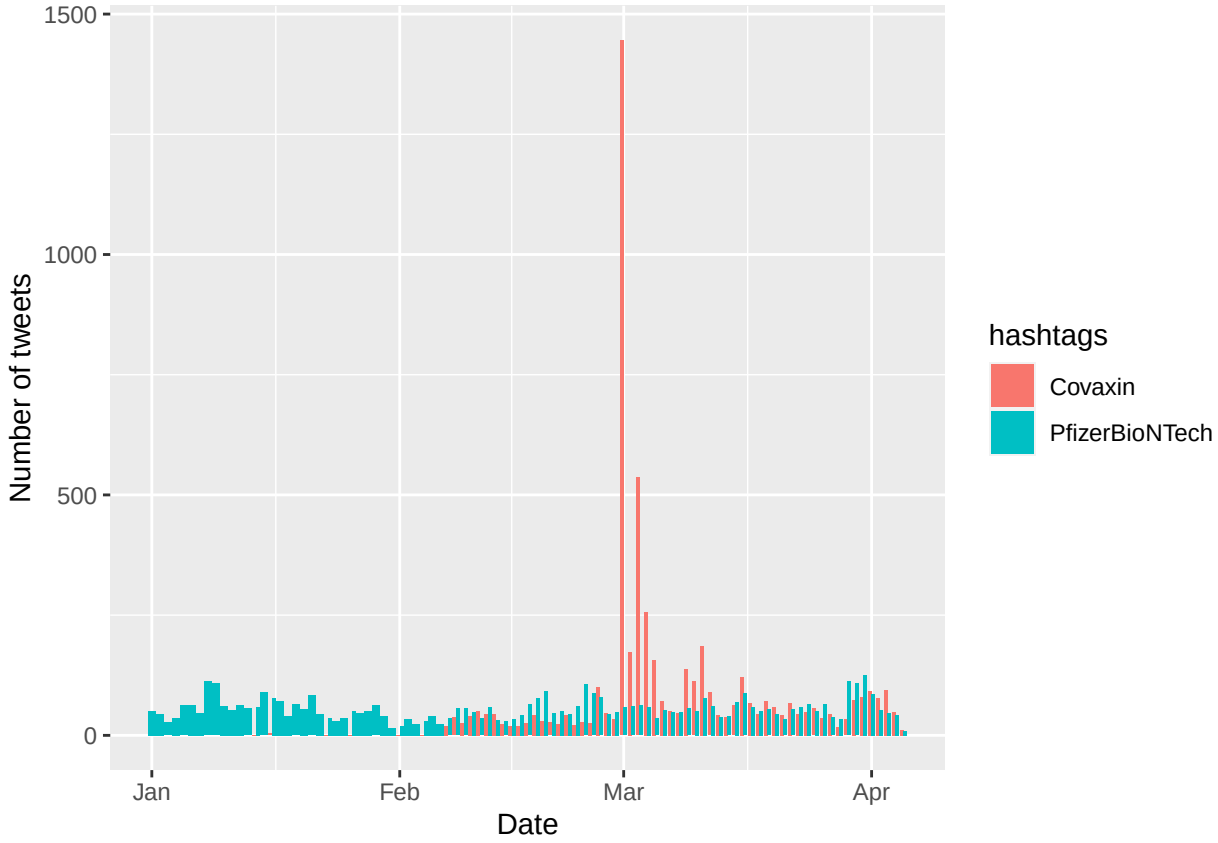
Question 2

(a)

```
data = data.table(read.csv('vaccinationtweets.csv'))

# separate date column
data[, newdate := sapply(strsplit(date, " "), `[`, 1)]
data[, newdate := as.Date(c(newdate), format = "%m/%d/%Y")]
class(data$newdate)
```

```
## [1] "Date"
```

```
# subset to date range
sub_data <- data[newdate <= as.Date("2021-04-30") & newdate >= as.Date("2021-01-01")]

# subset to hashtags
sub_data <- sub_data[hashtags %chin% "PfizerBioNTech" | hashtags %chin% "Covaxin"]

# plot
ggplot(data = sub_data, aes(x = newdate, fill = hashtags)) +
  geom_bar(stat = "count", position="dodge") +
  xlab("Date") +
  ylab("Number of tweets")
```



(b) I believe we should remove rare words based on their document frequency, because they might be attached high relevancy to specific topics just because they happen to appear in documents associated with those topics, not because they are actually relevant. If we maintain those words, our model might not have

strong predictive power.

(c)

```
# Remove non ASCII characters
sub_data$text <- stringi::stri_trans_general(sub_data$text, "latin-ascii")

# Removes solitary letters
sub_data$text <- gsub(" [A-z] ", " ", sub_data$text)

# Create DFM
vax_dfm <- dfm(sub_data$text,
               stem = F,
               remove_punct = T,
               remove_numbers = TRUE,
               tolower = T,
               remove = c(stopwords("english"), "http","https","rt", "t.co"))

# remove rare and common words
vax_dfm = dfm_trim(vax_dfm, min_docfreq = 0.01, max_docfreq = 0.9, docfreq_type = "prop")
```

We should also remove non-ASCII characters, numbers, solitary letters, punctuation, stop words, and words that are too common, and applying stemming and lower case.

(d)

```
# new binary var
sub_data[, bi_hashtag := ifelse(hashtags == "PfizerBioNTech", 1, 0)]

vax_dfm <- dfm(sub_data$text,
               stem = F,
               remove_punct = T,
               remove_numbers = TRUE,
               tolower = T,
               remove = c(stopwords("english"), "http","https","rt", "t.co"))
vax_dfm = dfm_trim(vax_dfm, min_docfreq = 0.01, max_docfreq = 0.9, docfreq_type = "prop")

vax_stm <- convert(vax_dfm, to = "stm")

a <- data.frame(sub_data$bi_hashtag)[-c(10631:10636),]
b <- data.frame(sub_data$newdate)[-c(10631:10636),]

#vax_stm_1 <- stm(vax_stm$documents, vax_stm$vocab, K = 30, data = vax_stm$meta, init.type = "Spectral"
```

124 iterations were completed before the model converged.

(e)

```
#save(vax_stm_1, file = "stm1.Rdata")
#vax_stm_saved = data("stm1.Rdata")
#plot(vax_stm_1, type = "summary")
```

Topics that occur in the highest proportion of documents: 1. Topic 10 (#covaxin, prime, will) 2. Topic 25 (vaccin, will, #ocugen) 3. Topic 12 (#pfizerbiotech, #pfizervaccin, shot) 4. Topic 24 (dose, thank) 5. Topic 27 (#pfizer, studi, variant)

(f)

```
#vax_stm_2 <- stm(vax_stm$documents, vax_stm$vocab, K = 100, data = vax_stm$meta, init.type = "Spectral
#plot(vax_stm_2, type = "summary")
```

My topics are substantively different after setting K=100. I prefer the model with a smaller value of K, because the topics seem more differentiable, whereas with K=100 the topics seem really similar.

(g)

```
#plot(vax_stm_1, type="perspectives", topics = c(1,2))
```

Question 3

(a)

Pretrained word embeddings are useful for document classification. One example is the "Soft" Dictionary – by embedding both the target and each element in the dictionary, we can return a classification by choosing the element with the highest cosine similarity. It is easy to downstream, and takes care of the limitation on the list of terms of a dictionary.

(b)

I think accuracy would be a good indicator for whether you've overcome the non-exhaustive dictionary problem. The standard dictionary-based method does not generalize well to unseen terms, whereas the soft dictionary method can assign more accurate class membership based on cosine similarity, thus achieving a higher accuracy on the test data.

(c)

We could fit a word2vec model with CBoW, to learn word embeddings using a single-layer neural net with backpropagation. The hidden units then become the word embeddings, which have a much smaller size than the vocabulary. This way, the model scales better with number of words and documents, and updates more easily.

we can also use word embeddings to reduce the vocabulary size, which is done by choosing a size for the word embeddings that is smaller than the vocabulary size. Some word embedding techniques can also split the terms into sub-units, which reduces the vocabulary size as well.

(d)

I would choose to train new embeddings from scratch. This is because these existing embeddings are static, meaning they don't change based on the context they are used in, and for a sentiment analysis task, we should take context into consideration. This can be addressed by training a model with new contextual embeddings.

(e)

Using pretrained word embeddings in a new classifier could incorporate societal biases, which may potentially perpetuate stereotypes, inequality, or even hate. For example, word embeddings pretrained in the 80s might contain a lot of racial stereotypes that affect the model's ability to assign sentiment scores.