Text As Data HW 3

Jean An (cya220)

4/28/2022

1. LDA concepts

During lab, you've gone over the code that you need in order to fit LDA topic models. Both R and Python have a rich set of tools and tutorials for fitting topic models. In this problem, we'll instead focus on the concepts underlying topic models, specifically latent Dirichlet allocation, and the decisions you might make in fitting them.

(a) Your boss has asked you to fit an LDA model on some proprietary text in your organization. The corpus of text is very large and he needs the model as soon as possible. Your LDA package gives you three options for fitting LDA: EM, variational inference, and Gibbs sampling. Which should you select, and why? (1-2 sentences)

I would select variational inference as the fitting method because it is likely to be the fastest. Although Gibbs sampling is guaranteed to approach the true distribution, we don't really know how long it will take for the MCMC model to run; therefore, the variational inference method, which approximates the true posterior, should be good enough in this case.

(b) Vanilla LDA requires you to specify k, the number of topics. Describe three approaches you could take to selecting the number of topics for your model. (3-4 sentences)

Method 1:Use a previously held-out validation set to evaluated and update the fitting of the model. Method 2: Select k based on prior knowledge or observation on the data set. Method 3: Silhouette method (maximize the silhouette coefficient for each data point).

(c) Poking around in your organization's Github, you find two LDA models that were fit by a previous employee.

i. You examine the document-topic distribution (θ_i) for a few documents ($i = 1 \dots N$) produced by Model 1, and find that the topics are very sparse, i.e., each document's topic distribution is highly concentrated in one or two topics, with little to no probability mass on other topics. Which hyperparameter is the likely cause of this and why? (1-2 sentences)

This is most likely caused by a low alpha value that is smaller than 1. This suggests that a given document is generally from one of a few topics.

ii. You examine Model 2 and find that the topic-word distribution (β_k) is very flat (i.e., each topic assigns a similar probability to each word within the topic). Which hyperparameter is the likely cause of this and why? (1-2 sentences)

This is most likely caused by a high eta value. This is a prior on the Dirichlet distribution that governs word probability per-topic.

(d) Your boss's boss examines the output of your model and has a concern. She has a PhD in political science and insists on having full uncertainty estimates around all parameters in the model. She asks you to refit the model, telling you that runtime is not a concern. Which method for fitting do you select now, and why? (1-2 sentences)

I would select Gibbs sampling as the fitting method because is guaranteed to approach the true distribution. Since runtime is not a concern, we can let the MCMC model do its work.

(e) Finally, you have a model you're satisfied with. You present it to your group and someone asks you, "after we condition on the topic θ_i, are the words in the document independent of each other?" You pull up your slide with the LDA plate diagram and write the conditional probability of a word p(w|????) = ????, answering your colleague's question. What is your answer and what expression do you write? (1 sentence with answer and explanation, plus one mathematical expression to justify it)

Yes, the words in the document are independent of each other because:

$p(w \mid d) = p(w \mid \theta, d)p(\theta \mid d) = p(w \mid \theta)p(\theta \mid d)$

2. STM: Topic Models with covariates

```
library(tidyverse)
library(quanteda)
library(stm)
library(topicmodels)
```

(a)

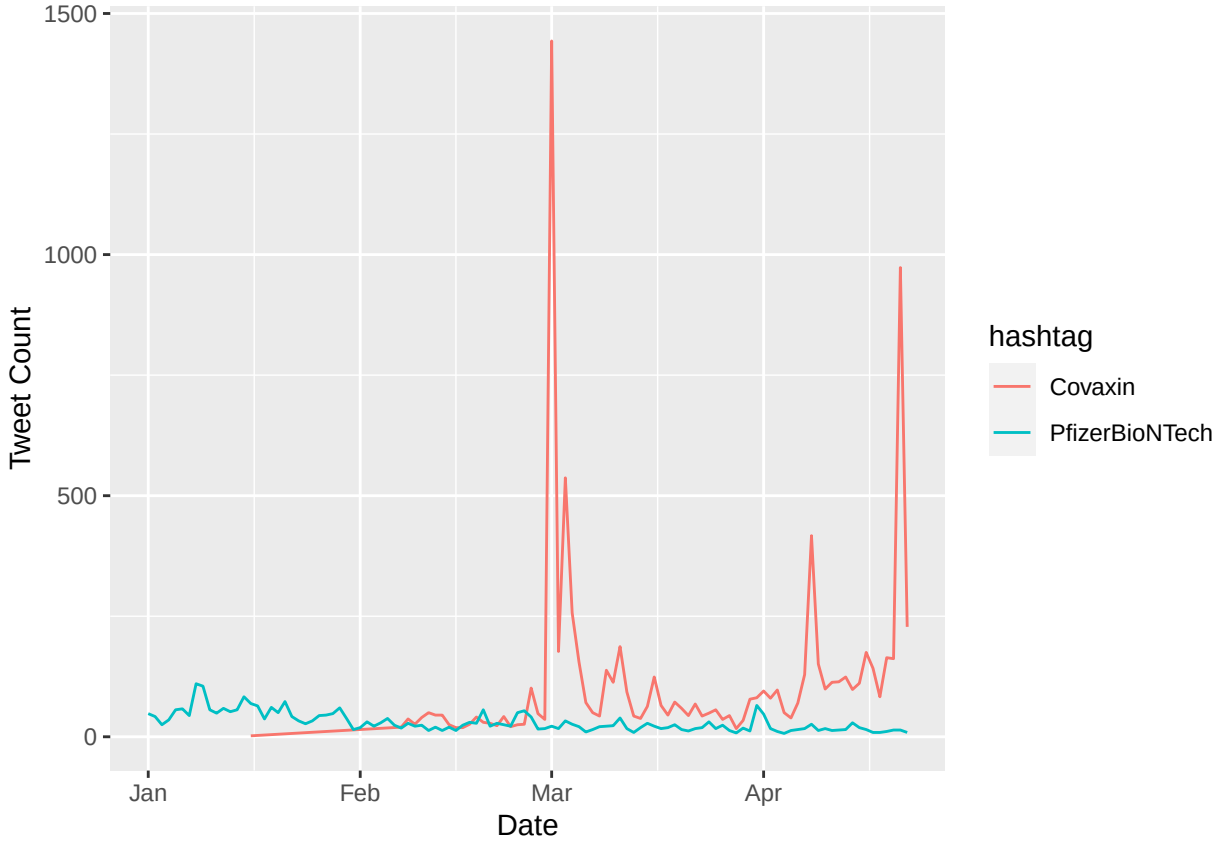
```
vaccination <- read_csv("vaccination_all_tweets.csv", show_col_types = FALSE)

vaccination <- vaccination %>%
  mutate(date = as.Date(date, "%m/%d/%Y"),
         hashtag = ifelse(str_detect(hashtags, "(?i)PfizerBioNTech"),"PfizerBioNTech",
                          ifelse(str_detect(hashtags, "(?i)Covaxin"),"Covaxin", NA)))

vax_tweet <- vaccination %>%
  filter(date >= "2021-01-01", date <= "2021-04-30",
         hashtag %in% c("PfizerBioNTech","Covaxin"))

vax_count <- vax_tweet %>%
  group_by(date, hashtag) %>%
  summarize(count = n())

ggplot(vax_count, aes(x = date, y = count,
                      group = hashtag, color = hashtag)) +
  geom_line() + xlab("Date") + ylab("Tweet Count")
```



(b) Preprocessing decisions can have substantive impacts on the topics created by topic model algorithms. Make a brief argument for or against removing rare terms from a dfm on which you plan to fit a topic model. (2-4 sentences)

On the one hand, removing rare terms from a dfm is beneficial because these rare terms will most likely not affect the outcome of the model (i.e., it will not be one of the most common topics). Removing them would reduce dimensionality of the data. On the other hand, some rare terms may just be the core of a certain piece; perhaps the word is so powerful that mentioning it once or twice would already convey a strong message. In this case, removing these rare terms may cause the model to miss the important topic.

(c) What other preprocessing you believe to be appropriate for a topic modeling problem like this one? Discuss your preprocessing choices and apply them to the tweet corpus. (1-4 sentences)

The preprocessing I believe to be appropriate for a topic modeling problem include converting to lower case, removing punctuation, removing stopwords, and stemming. These preprocessing steps are all aimed to reduce the dimension of the data set, and rarely do they lead to substantial changes to the meaning of the texts.

(d)

```
vax_tweet <- vax_tweet %>%
  mutate(binary = ifelse(hashtag == "PfizerBioNTech", 1, 0))

processed <- textProcessor(vax_tweet$text, metadata = vax_tweet)
```

```
## Building corpus...
## Converting to Lower Case...
## Removing punctuation...
## Removing stopwords...
## Removing numbers...
## Stemming...
## Creating Output...
```

```
out <- prepDocuments(processed$documents, processed$vocab,
                     processed$meta, lower.thresh = 10)
```

```
## Removing 22669 of 23847 terms (32803 of 120843 tokens) due to frequency
## Removing 1 Documents with No Words
## Your corpus now has 12185 documents, 1178 terms and 88040 tokens.
```

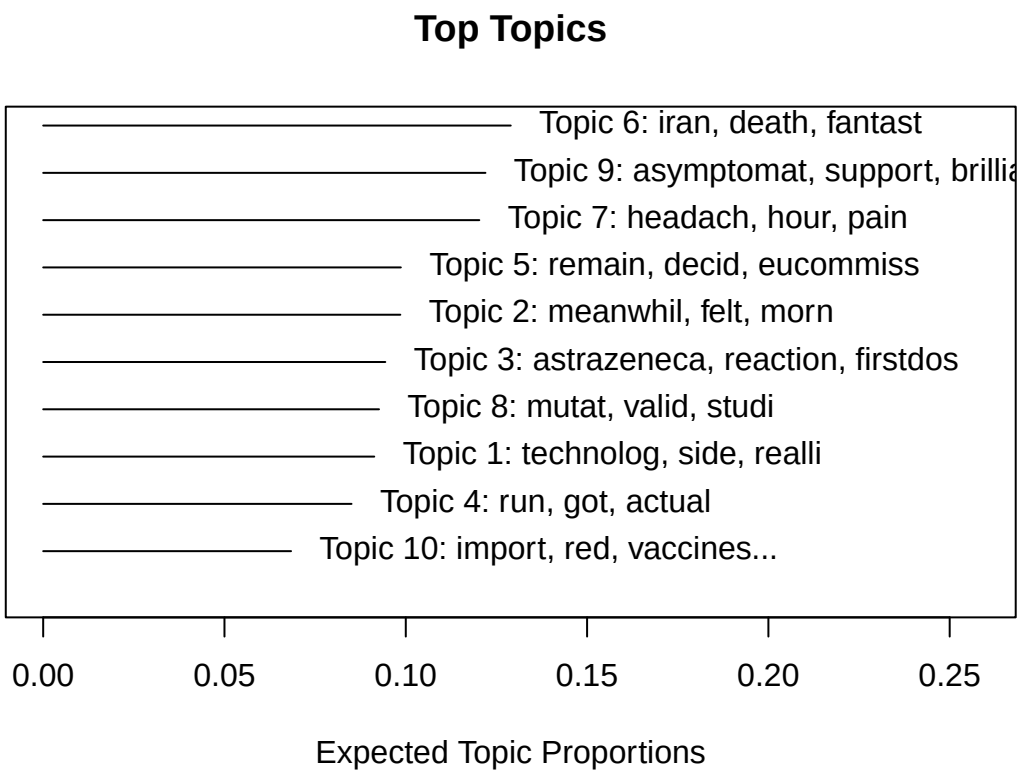
```
#stm_fit <- stm(out$documents, out$vocab, K = 10,
#               prevalence = ~binary + s(date),
#               content = ~binary, data = out$meta)
stm_fit <- readRDS("stm_fit.rds")
stm_fit$convergence$its
```

```
## [1] 80
```

A total of 79 iterations completed before the model converged.

(e)

```
plot(stm_fit, type = "summary")
```



The five topics that occur in the highest proportion of documents are:

Topic 6: iran, death, fantast
Topic 9: asymptomat, suppott, brilliant
Topic 7: headach, hour, pain
Topic 5: remain, decid, eucommiss
Topic 2: meanwhil, felt, morn

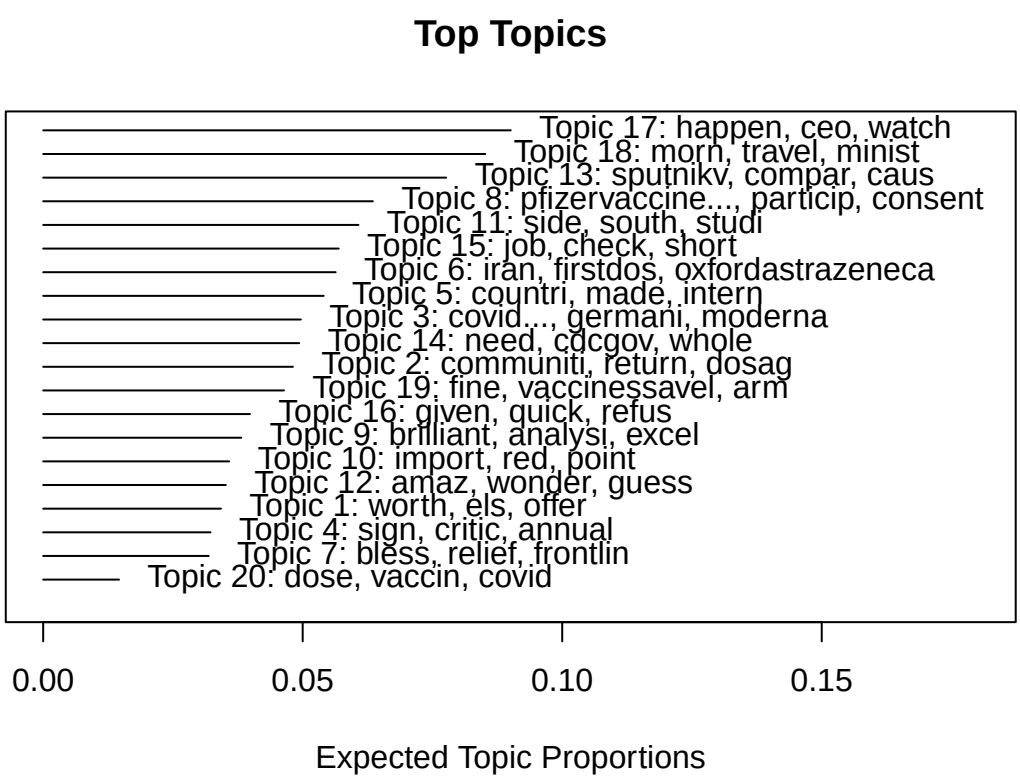
(f)

```
#stm_fit_2 <- stm(out$documents, out$vocab, K = 20,
#                  prevalence = ~binary + s(date),
#                  content = ~binary, data = out$meta)
stm_fit_2 <- readRDS("stm_fit_2.rds")
stm_fit_2$convergence$its
```

```
## [1] 4
```

A total of 3 iterations completed before the model converged.

```
plot(stm_fit_2, type = "summary")
```



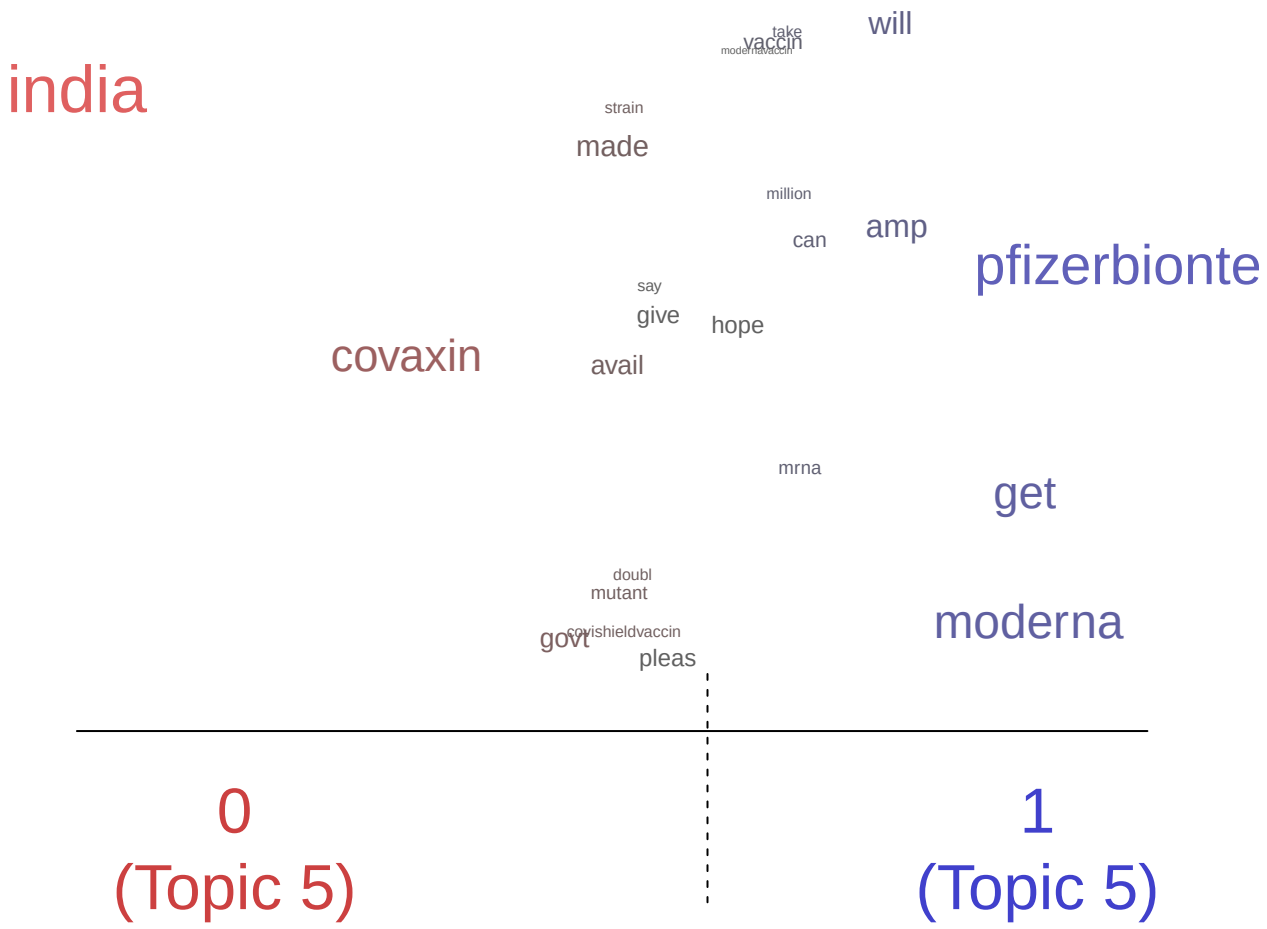
The five topics that occur in the highest proportion of documents are:

Topic 17: happen, ceo, watch
Topic 18: morn, travel, minist
Topic 13: sputnikv, compar, caus
Topic 8: pfizervaccine. . . , partocip, consent
Topic 11: side, south, studi

Yes, the topics are substantially different. I think I prefer the previous ones, with k = 10.

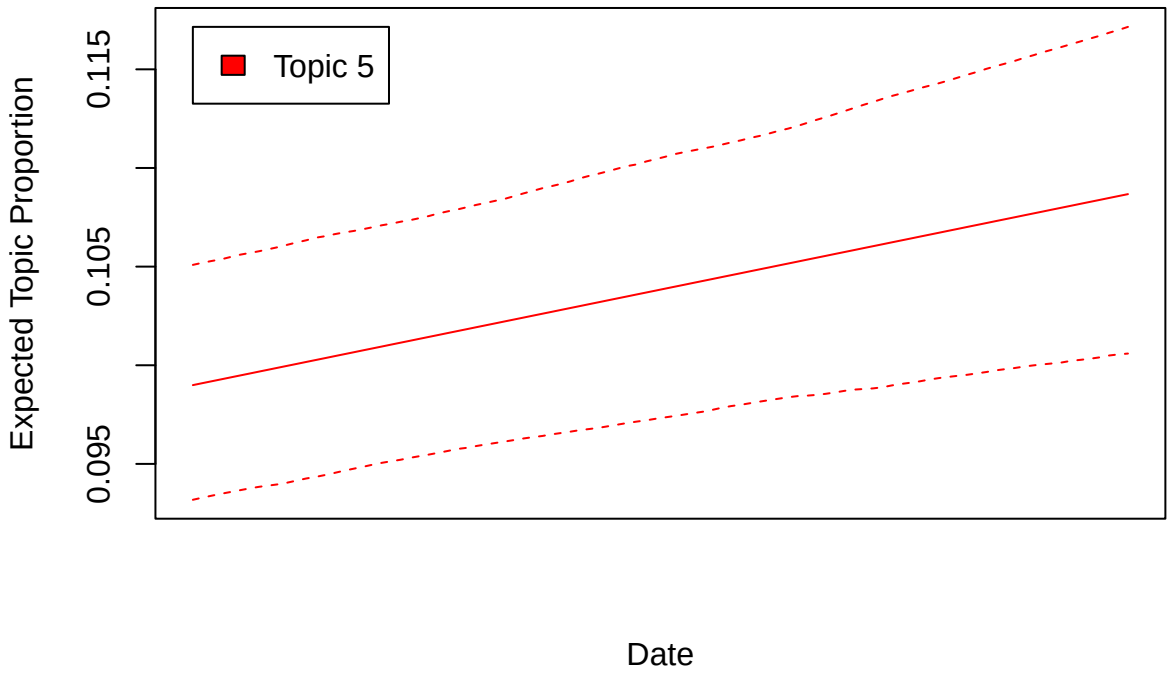
(g)

```
plot(stm_fit, type="perspectives", topics = 5)
```



In topic 5 from the first model, the content for tweets in the binary classification of "Covaxin" has "india" as its most common term. This makes total sense, as Covaxin is the COVID vaccine from India. On the other hand, it is interesting that for the "PfizerBioNTech" classification, "moderna" is actually one of the common terms.

```
out$meta$date <- as.numeric(out$meta$date)
prep <- estimateEffect(c(5) ~ binary + s(date) , stm_fit, meta = out$meta)
plot(prep, covariate = "binary", model = stm_fit, topics = 5,
     method = "continuous", xaxt = "n", xlab = "Date")
```



The prevalence increases over time.

3. Applications of Word Embeddings

(a) One of the limitations of dictionary-based methods is that dictionaries rarely include an exhaustive list of terms. Propose and justify a method for improving a dictionary using pretrained word embeddings. (2-3 sentences)

By using pretrained word embeddings, we could expand and increase the list of terms within a dictionary. For example, let's say I have a dictionary that includes many commonly used baseball terms; but there are obviously many more less-common baseball terms that are missing. Using pretrained word embeddings (perhaps trained using baseball articles), we can then find many of these less-common baseball terms and add them to the dictionary.

(b) Now imagine that you've implemented your method from the previous question and compared it to a standard dictionary-based method. Which accuracy measure (accuracy, F1, precision, recall...) would best indicate whether you've overcome the non-exhaustive dictionary problem? (1-2 sentences)

The accuracy measure that would best indicate that I have overcome the non-exhaustive dictionary problem would be accuracy and recall. The improved dictionary should increase the occurrences of true positives and true negatives.

(c) In many models that we apply to bag-of-words representations, the size of the vocabulary is the greatest determinant of model size and runtime. Much of the early class covered ways to reduce the dimensionality of our text through stopword removal, stemming, lowercasing, etc. Describe and justify a procedure for using word embeddings to reduce the vocabulary size of a bag-of-words document representation. (N.B.: we still want our model to operate on a (reduced) bag of words, not one directly on word embeddings). (2-3 sentences)

Word embeddings could be used to reduce the vocabulary size of a bag-of-words document representation by grouping similar words together. Some words that are synonyms often have nothing in common on the surface (i.e., letters) and are therefore treated as very different words in terms of tokens. Word embeddings would be able to solve this issue; however, it also has the downside of confusing words that have more than one meaning.

(d) Your impatient boss from problem 1 has heard about word embeddings and wants you to use them as features in a supervised model to label customer emails with sentiment scores. Should you train new embeddings from scratch or use existing, off-the-shelf embeddings? What are the factors in your decision? (2-3 sentences).

I should use off-the-shelf embeddings. Most of the time, customer emails contain very common words that should already be trained in existing models. Very rarely would there be terms that have to be trained newly. Therefore, existing, off-the-shelf embedding would be fine.

(e) Finally, briefly discuss the potential ethical issues of using pretrained word embeddings in your classifier. (2-3 sentences)

In the lecture slides, we were shown an example where a pretrained word embeddings produces very different sentiment score when one single word in the sentence is switched out: the specific ethnicity/country when discussing various cuisines. With the term "Italian" included, in produces a high score; with the term "Chinese" included, a lower score; and with the term "Mexican" included, a very low score. This would potentially lead to a problem in my classifier, as the pretrained embeddings has produced unintended consequences.