Lab 5: Sentiment & Topic Models

Partial Solutions

Social Media and Web Analytics @ TiSEM

Last updated: 25 May, 2021

Motivation

Understanding consumer sentiment towards a brand or product and being able to synthesize the large volume of text into a set of topics consumers are discussing are important tasks as marketing analysts. Using the text data available on social media platforms provides analysts with a valuable source of data to be able to gain insight into both of these areas.

Learning Goals

By the end of this tutorial you will be able to:

1. Classify the sentiment of a text as positive, negative or neutral.
2. Evaluate the performance of a sentiment classification using confusion matrices and accuracy metrics.
3. Discuss whether a pre-existing sentiment lexicon should be used in a particular application.
4. Estimate a topic model to identify topics within a corpus of texts.
5. Identify coherent topic themes from output of a topic model.
6. Evaluate the co-occurence of sentiment and/or fake reviews within particular topics.
7. Discuss the managerial relevance of your findings from text analysis.

Instructions to Students

These tutorials are **not graded**, but we encourage you to invest time and effort into working through them from start to finish. Add your solutions to the `lab-05_answer.Rmd` file as you work through the exercises so that you have a record of the work you have done.

Obtain a copy of both the question and answer files using Git. To clone a copy of this repository to your own PC, use the following command:

```
$ git clone https://github.com/tisem-digital-marketing/smwa-lab-05.git
```

Once you have your copy, open the answer document in RStudio as an RStudio project and work through the questions.

The goal of the tutorials is to explore how to "do" the technical side of social media analytics. Use this as an opportunity to push your limits and develop new skills. When you are uncertain or do not know what to do next - ask questions of your peers and the instructors on the class Slack channel `#lab-05-discussion`.

Getting Started: Data & R Packages

This Lab revisits the data on hotel reviews from Lab 4. As a reminder: The reviews are a collection of truthful and deceptive (i.e. fake) reviews of 20 hotels in the Chicago area, known in the computational linguistics community as the Deceptive Opinion Spam dataset.[1] Deceptive reviews are reviews that have been written by someone who has not stayed at the hotel they are reviewing. The data contains 1600 reviews:

- 400 truthful, positive reviews from TripAdvisor
- 400 deceptive positive reviews from Mechanical Turk
- 400 truthful, negative reviews from Expedia, Hotels.com, Orbitz, Priceline, TripAdvisor, and Yelp
- 400 deceptive negative reviews from Mechanical Turk

To gain access to the data, run the following code to download it and save it in the `data` directory:

```r
library(googledrive)

data_id <- "16zwq8pGhvTV8IrhTNZUQvwCddyHqpbFO"
out_file <- "data/lab_05.zip"

drive_download(
  as_id(data_id),
  path = out_file,
  overwrite = TRUE)

# --- Unzip and Clean up --- #
unzip(out_file,
      exdir = "data")

file.remove(out_file)
```

```
## [1] TRUE
```

You might need to use the following `R` libraries throughout this exercise:[2]

```r
library(readr)
library(dplyr)
library(tibble)
library(tidyr)
library(tidytext)
library(ggplot2)
library(textstem)
library(vader)
library(yardstick)
library(stm)
```

Exercise 1: Sentiment Analysis

One of the main tasks marketers perform with text is sentiment analysis, i.e. classifying text as positive, negative to neutral in tone. The VADER sentiment lexicon (Hutto and Gilbert, 2014) is one of the better performing methods for sentiment analysis if one does not want to engage in a complex statistical exercise to create a customized sentiment model.[3] This exercise is going to use the VADER lexicon to evaluate the

[1] The data originally was published in the paper "Finding Deceptive Opinion Spam by Any Stretch of the Imagination" by M. Ott, Y. Choi, C. Cardie, and J.T. Hancock in 2011 in the Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies

[2] If you haven't installed one or more of these packages, do so by entering `install.packages("PKG_NAME")` into the R console and pressing ENTER.

[3] Want to know more about VADER? Read the original paper here. The paper isn't too long. SentiBench (Riberio et al, 2016) provides a comprehensive evaluations of sentiment lexicons in English.

sentiment of hotel reviews from the Deceptive Opinion Spam dataset.

1. Why might the classification of text into positive, negative and neutral sentiment be useful for marketers and managers?
2. Load the hotel reviews data into R with the name `hotel_reviews`. After you have loaded the data, add a column `id` that creates a unique id to each row of the data.

solution

```
hotel_reviews <-
  read_csv('data/reviews.csv') %>%
  rownames_to_column("id")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   deceptive = col_character(),
##   hotel = col_character(),
##   polarity = col_character(),
##   source = col_character(),
##   text = col_character()
## )
```

3. Create a smaller dataset called `hotel_sentiment` that only includes the columns `id`, `polarity`, and `text`.

solution

```
hotel_sentiment <-
  hotel_reviews %>%
  select(id, polarity, text)
```

As discussed above, our weapon of choice for sentiment analysis will be the VADER lexicon. VADER doesn't want the data in a 'tidy' format because it uses the punctuation, capitalization and emojis when it evaluates the sentiment in a text. Let's get started using VADER.

4. To classify multiple review's sentiment in one go, the `vader` package has a function called `vader_df()`. The starter code below shows you how to use the function - you pass across the column of the dataset that has the text you want to analyse row by row. Adapt the code to run on the `hotel_sentiment` data set.

```
# VADER is pretty nice in that we shouldn't need to clean it
vader_sent <-
  vader_df(DATANAME$TEXTCOLUMN)
```

NOTE: Note that when you run it, it might take a while to run from start to finish.

solution

```
# VADER is pretty nice in that we shouldn't need to clean it
vader_sent <-
  vader_df(hotel_sentiment$text)
```

```
## Warning in vader_df(hotel_sentiment$text): 2 rows contain an error. Filter
## word_scores for 'ERROR' to identify the problematic text.
```

The output here is useful. First, some reviews generated errors, and we'll need to drop those for the rest of our analysis. The main column of interest is `compound` which computes the sentiment of a text as a number ranging between -1 (most negative) and +1 (most positive).

The code below uses the compound score to classify a review as positive or negative. To run it, change `eval=FALSE` to `eval = TRUE` in the Rmd file.

```
vader_sent2 <-
  vader_sent %>%
  # we need a row number to merge it back into
  # our original data
  rowid_to_column("id") %>%
  # remove any errors
  filter(word_scores != ’ERROR’) %>%
  # classify as positive or negative
  mutate(vader_class = case_when(
        compound < 0 ~ "negative",
        # the final case must always be written as
        # TRUE ~ SOMETHING
        TRUE ~ "positive"
        )
    ) %>%
  select(id, vader_class)
```

5. Merge the sentiment classifications from VADER back into the `hotel_sentiment` data. Use the following code to get started:

```
hotel_sentiment <-
    YOUR_CODE %>%
    # we need the ids to be the same type
    mutate(id = as.integer(id)) %>%
    inner_join(YOUR_CODE, by = "id")
```

solution

```
hotel_sentiment <-
    hotel_sentiment %>%
    mutate(id = as.integer(id)) %>%
    inner_join(vader_sent2, by = "id")
```

The hotel data already had a sentiment measure in it, `polarity`. Let's compare the VADER sentiment classification to this measure to see how well it performed.

6. We will measure VADER's performance using a confusion matrix and assessing model accuracy. Think of the `polarity` column as the true classification to compare predictions to.

(a) Explain what a confusion matrix is.
(b) Evaluate VADER's predictions relative to `polarity` using a confusion matrix (`conf_mat()` in R).
(c) Explain what model accuracy is.
(d) Evaluate VADER's accuracy relative to `polarity` using a confusion matrix (`accuracy()` in R).

solution

```
# how well do we do?
hotel_sentiment %>%
  conf_mat(polarity, vader_class)
```

```
## Warning in vec2table(truth = truth, estimate = estimate, dnn = dnn, ...):
## 'truth' was converted to a factor

## Warning in vec2table(truth = truth, estimate = estimate, dnn = dnn, ...):
## 'estimate' was converted to a factor

##            Truth
## Prediction negative positive
##   negative      376        3
##   positive      423      796
```

```
hotel_sentiment %>%
    accuracy(factor(polarity), factor(vader_class))
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.733
```

7. The authors of the VADER lexicon advocate for using three classes for prediction - positive, negative and **neutral**. Their suggestion is to classify text into these three classes as follows:

- Positive Tweet: $compound \in (0.05, 1]$
- Neutral Tweet: $compound \in [-0.05, 0.05]$
- Negative Tweet: $compound \in [-1, -0.05)$

Update the provided code above to implement this three class classification. (You should not need to re-run the `vader_df()` command to do this)

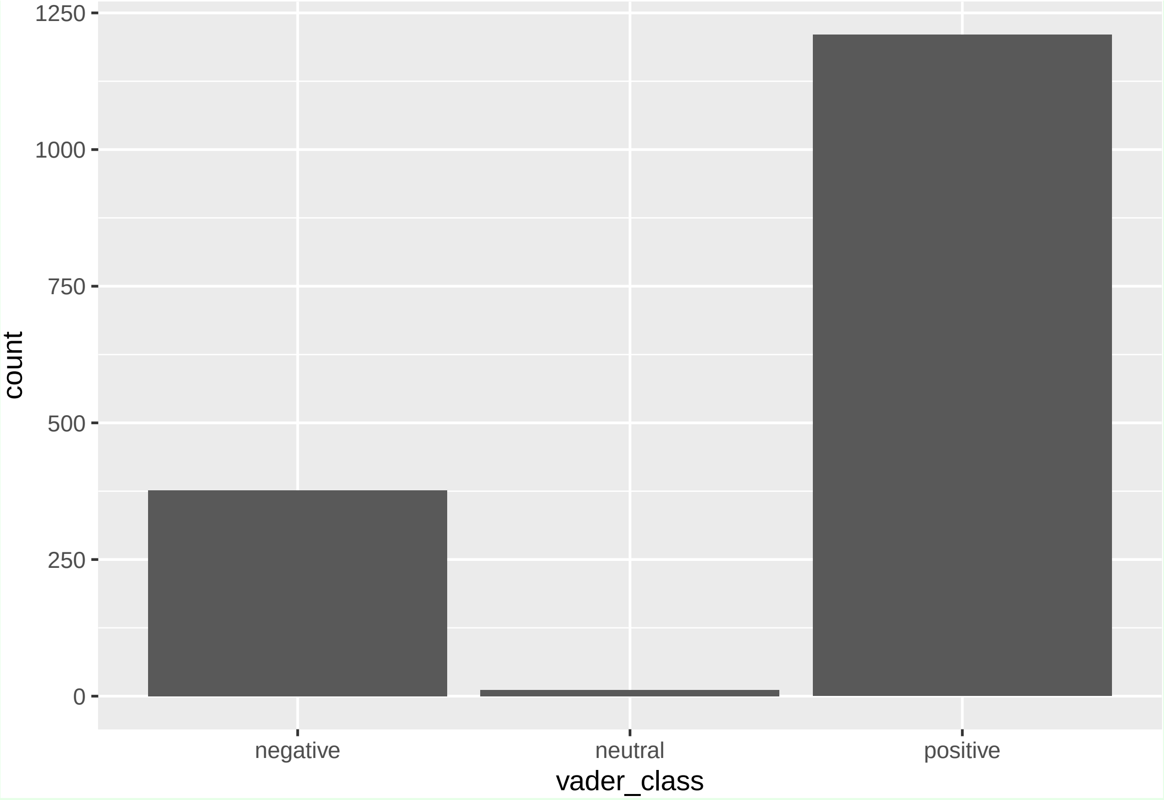
solution

```
# vader prefers a neutral class tho, can it help us?
vader_sent3 <-
  vader_sent %>%
  rowid_to_column("id") %>%
  filter(word_scores != 'ERROR') %>%
  mutate(vader_class = case_when(
        compound < -0.05 ~ "negative",
        compound > 0.05 ~ "positive",
        TRUE ~ "neutral"
        )
    ) %>%
  select(id, vader_class)
```

8. Plot the frequency of each class from (7) as a bar chart. Does the plot suggest that adding a neutral class provides an improvement in this example?

solution

```
vader_sent3 %>%
  ggplot(aes(x = vader_class)) +
  geom_bar()
```



9. Based on the performance we see here, would you recommend using VADER to classify hotel reviews as positive or negative if you were a marketing analyst for a hotel chain? Why or why not? If not, what might you do instead?

10. (Optional) The reading for this week, Text Mining with R, uses different sentiment lexicons to classify texts. Try one or more of these out on this text, and evaluate their performance. Can you find a lexicon that does better than VADER?

Exercise 2: Topic Models

Now that we've explored sentiment, we turn to using the text to learn the topics hotel reviewers are discussing. By the end of the exercise, you will be able to plot and discuss how the topics mentioned in a review differ across either (i) positive and negative reviews, or (ii) fake and truthful reviews.

To understand the topics in a text we are going to estimate a Topic Model.[4] We are going to estimate topic models using the `stm` package. In my personal experience, I have found the results from this package to be the most reliable out of the options available in `R`.

To get you started, we have tidied the hotel review text and saved the results in `data/tidy_reviews.csv`. Use this data as your starting point.[5]

1. Load the data from `data/tidy_reviews.csv` into R.

[4] See Chapter 6 of Text Mining with R for an introduction.

[5] If you are interested in the code we used to clean up the review text, we've included it for you to browse in `data/tidy_reviews.R`. It's lacking comments, but hopefully one can get the jist of what is going on.

solution

```
tidy_reviews <-
  read_csv("data/tidy_reviews.csv")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   id = col_double(),
##   deceptive = col_character(),
##   polarity = col_character(),
##   word = col_character()
## )
```

To make progress towards a identifying topics from a corpus of texts, we first need to transform the data to the right format.

2. Each review in the data is indexed by the column `id`. For each review, count the number of times a word occurs in it. Leave the data in it's 'long' format.

solution

```
doc_word_counts <-
    tidy_reviews %>%
    count(id, word) %>%
    ungroup()
```

3. Now, we need to transform the word counts from above into a "document term matrix". Since we are going to use the `stm` package, the correct command is `cast_sparse()`. Use the starter code below to make this transformation.

```
reviews_dtm <-
    YOUR_CODE %>%
    cast_sparse(id, word, n)
```

solution

```
reviews_dtm <-
    doc_word_counts %>%
    cast_sparse(id, word, n)
```

We are now ready to estimate a topic model. Let's get started.

4. What is a topic model? Intuitively how does it work?
5. We will estimate a topic model using the `stm package`. You need to pass the `stm` function the document term matrix from (3) and the number of topics you want the model to identify (called K). Use the starter code to estimate a model with 10 topics.

NOTE: When you run the code, it may take a little while to come up with a final set of topics. Please be patient!

```
reviews_lda <-
  stm(YOUR_CODE,
      K = YOUR_CODE,
      # seed fixes the random number draw
      # so we should all get the same results
      seed = 123456789)
```

solution

```r
reviews_lda <-
  stm(reviews_dtm,
      K = 10,
      seed = 123456789)
```

```
## Beginning Spectral Initialization
##   Calculating the gram matrix...
##   Finding anchor words...
##      ..........
##   Recovering initialization...
##      ................
## Initialization complete.
## .................................................................................................
## Completed E-Step (0 seconds).
## Completed M-Step.
## Completing Iteration 1 (approx. per word bound = -6.578)
## .................................................................................................
## Completed E-Step (0 seconds).
## Completed M-Step.
## Completing Iteration 2 (approx. per word bound = -6.408, relative change = 2.579e-02)
## .................................................................................................
## Completed E-Step (0 seconds).
## Completed M-Step.
## Completing Iteration 3 (approx. per word bound = -6.375, relative change = 5.228e-03)
## .................................................................................................
## Completed E-Step (0 seconds).
## Completed M-Step.
## Completing Iteration 4 (approx. per word bound = -6.364, relative change = 1.725e-03)
## .................................................................................................
## Completed E-Step (0 seconds).
## Completed M-Step.
## Completing Iteration 5 (approx. per word bound = -6.359, relative change = 8.005e-04)
## Topic 1: breakfast, staff, clean, night, free
##  Topic 2: chicago, service, feel, business, luxury
##  Topic 3: bathroom, wall, lobby, bed, carpet
##  Topic 4: service, staff, time, chicago, request
##  Topic 5: shower, night, service, day, nice
##  Topic 6: chicago, staff, location, walk, restaurant
##  Topic 7: hard, chicago, service, rock, experience
##  Topic 8: call, check, desk, reservation, front
##  Topic 9: chicago, tower, sheraton, meet, sofitel
##  Topic 10: bed, pillow, size, king, bar
## .................................................................................................
## Completed E-Step (0 seconds).
## Completed M-Step.
## Completing Iteration 6 (approx. per word bound = -6.356, relative change = 4.278e-04)
## .................................................................................................
## Completed E-Step (0 seconds).
## Completed M-Step.
## Completing Iteration 7 (approx. per word bound = -6.354, relative change = 2.419e-04)
## .................................................................................................
## Completed E-Step (0 seconds).
## Completed M-Step.
## Completing Iteration 8 (approx. per word bound = -6.353, relative change = 1.507e-04)
## .................................................................................................
## Completed E-Step (0 seconds).
## Completed M-Step.
## Completing Iteration 9 (approx. per word bound = -6.353, relative change = 8.310e-05)
## .................................................................................................
## Completed E-Step (0 seconds).
## Completed M-Step.
## Completing Iteration 10 (approx. per word bound = -6.353, relative change = 3.675e-05)
## Topic 1: breakfast, free, night, clean, staff
##  Topic 2: chicago, service, business, feel, relax
##  Topic 3: bathroom, wall, carpet, lobby, bed
##  Topic 4: service, staff, time, chicago, weekend
##  Topic 5: shower, night, water, day, service
##  Topic 6: chicago, staff, location, walk, restaurant
##  Topic 7: hard, service, chicago, rock, price
##  Topic 8: call, check, desk, front, reservation
##  Topic 9: chicago, tower, meet, sheraton, sofitel
##  Topic 10: bed, king, size, pillow, bar
## .................................................................................................
## Completed E-Step (0 seconds).
## Completed M-Step.
## Model Converged
```

Now that we have results, we want to look at the results and see what topics the model came up with.

6. Use the `labelTopics` function to print out the top words associated with each topic. Do these topics seem distinct **and** managerially relevant?

solution

```
labelTopics(reviews_lda)
```

```
## Topic 1 Top Words:
##       Highest Prob: breakfast, free, night, clean, staff, floor, nice
##       FREX: breakfast, free, homewood, complimentary, continental, amalfi, kitchen
##       Lift: bedbug, memorial, ta, cereal, continental, toast, breakfast
##       Score: bedbug, breakfast, free, homewood, buffet, continental, amalfi
## Topic 2 Top Words:
##       Highest Prob: chicago, service, business, feel, relax, james, luxury
##       FREX: james, relax, spa, pet, fitness, luxury, center
##       Lift: tech, convenience, relieve, affina, massage, lodge, spa
##       Score: tech, spa, james, relax, luxury, chicago, massage
## Topic 3 Top Words:
##       Highest Prob: bathroom, wall, carpet, lobby, bed, floor, light
##       FREX: carpet, wall, white, toilet, stain, renovation, paint
##       Lift: yellow, stark, visible, white, outlet, temp, peel
##       Score: yellow, carpet, wall, toilet, white, peel, bathroom
## Topic 4 Top Words:
##       Highest Prob: service, staff, time, chicago, weekend, day, request
##       FREX: package, child, doorman, kid, special, girl, wine
##       Lift: monoco, fish, washington, recognize, snooty, soda, goldfish
##       Score: monoco, request, fish, package, monaco, service, doorman
## Topic 5 Top Words:
##       Highest Prob: shower, night, water, day, nice, pay, service
##       FREX: shower, hot, tub, fix, cold, low, break
##       Lift: engineer, crack, pipe, spoon, flood, peninsula, ritz
##       Score: engineer, shower, hot, tub, fix, cold, elevator
## Topic 6 Top Words:
##       Highest Prob: chicago, staff, location, walk, restaurant, shop, recommend
##       FREX: shop, beautiful, mile, fantastic, amaze, distance, perfect
##       Lift: tidy, shedd, coast, aquarium, lockwood, fantastic, mile
##       Score: tidy, shop, wonderful, michigan, amaze, perfect, mile
## Topic 7 Top Words:
##       Highest Prob: hard, service, chicago, rock, price, experience, top
##       FREX: hard, rock, top, notch, worth, fun, expensive
##       Lift: uncomfortably, rock, chicken, hard, impeccable, hip, unique
##       Score: uncomfortably, rock, hard, service, fun, price, top
## Topic 8 Top Words:
##       Highest Prob: call, check, desk, front, reservation, arrive, chicago
##       FREX: finally, clerk, reservation, manager, call, rude, credit
##       Lift: representative, statement, rollaway, nightmare, confirmation, refund, resolve
##       Score: representative, call, manager, finally, clerk, smoke, reservation
## Topic 9 Top Words:
##       Highest Prob: chicago, tower, meet, sheraton, sofitel, view, ambassador
##       FREX: sofitel, sheraton, tower, meet, ambassador, wed, east
##       Lift: traditional, shula's, marry, sofitel, sheraton, supper, chef
##       Score: traditional, sheraton, tower, sofitel, meet, wed, ambassador
## Topic 10 Top Words:
##       Highest Prob: bed, king, size, pillow, bar, tv, double
##       FREX: pillow, size, king, double, queen, mini, club
##       Lift: cabinet, cloud, pillow, queen, size, feather, king
##       Score: cabinet, bed, pillow, size, king, double, queen
```

7. Can you identify coherent themes for each topic from above? If so, write them down.[6] Which of the word lists did you find most helpful when trying to assign a name to each topic?

Our final step will be to assign each review one if the topics we found above.[7] Run the code below to find the topic that each review is most likely to belong to:

```r
reviews_gamma <-
    tidy(reviews_lda,
         matrix = "gamma",
         document_names = rownames(reviews_dtm)
    ) %>%
    rename(id = document) %>%
    group_by(id) %>%
    slice_max(gamma) %>%
    select(-gamma)
```

solution

[6] There's a bit of disagreement about whether this human annotation of topics identified to a 'name' is an OK thing to do. Since we want to be pragmatic in this class, we'll do it. But do keep in mind that some people might get upset by you doing this (including me, most of the time).

[7] Topic models generally assign a probability to a review belonging to each topic. We're going to say the topic with the highest probability **is** the topic of the review.

```r
reviews_gamma <-
    tidy(reviews_lda,
         matrix = "gamma",
         document_names = rownames(reviews_dtm)
    ) %>%
    rename(id = document) %>%
    group_by(id) %>%
    slice_max(gamma) %>%
    select(-gamma)
```

8. Merge `reviews_gamma` with the original hotel reviews data from Exercise 1. You will want to use the `inner_join()` function.

solution

```r
hotel_topics <-
  hotel_reviews %>%
  inner_join(reviews_gamma, by = "id")
```

9. Use the starter code below to update the `topic` variable. Currently, it is the topic number that the review was identified to below with. Your answer to (7) created topic labels which will be nicer to data visualization.

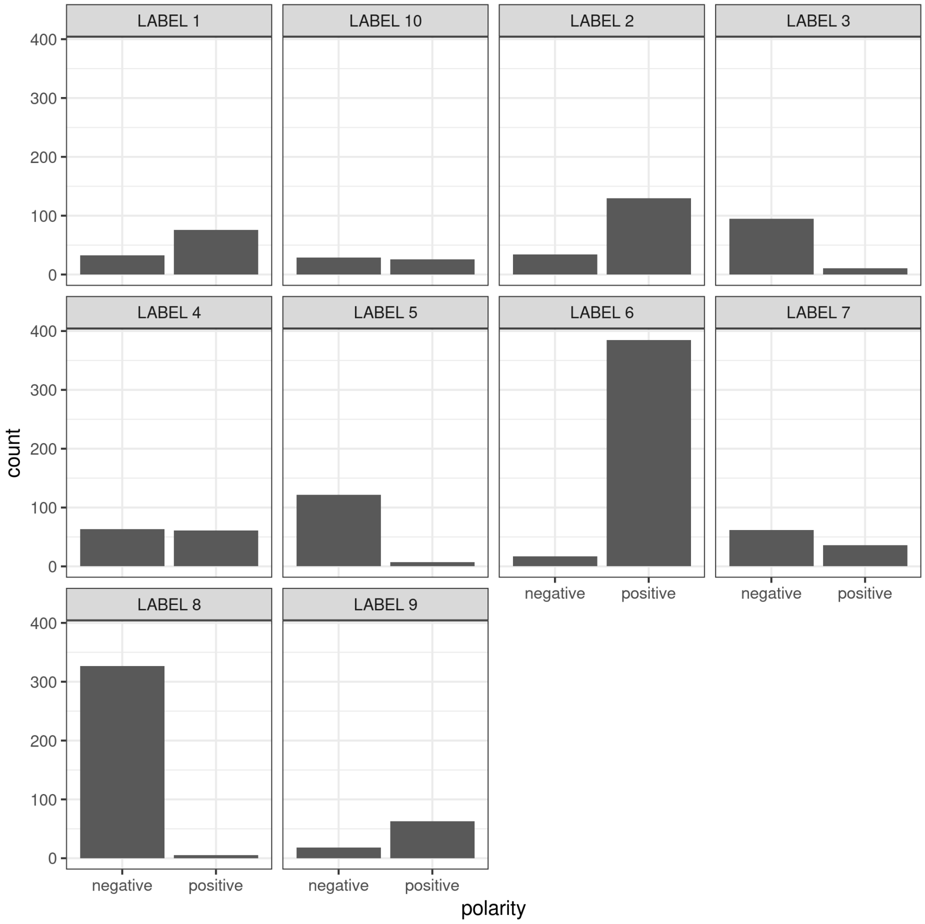
```r
# replace topic numbers with names
hotel_topics <-
  hotel_topics %>%
  mutate(topic = case_when(
    topic == 1 ~ "LABEL 1",
    topic == 2 ~ "LABEL 2",
    topic == 3 ~ "LABEL 3",
    topic == 4 ~ "LABEL 4",
    topic == 5 ~ "LABEL 5",
    topic == 6 ~ "LABEL 6",
    topic == 7 ~ "LABEL 7",
    topic == 8 ~ "LABEL 8",
    topic == 9 ~ "LABEL 9",
    # final case always uses this odd TRUE ~ notation
    TRUE ~ "LABEL 10"
    )
  )
```

solution

```r
# replace topic nums with names...
hotel_topics <-
  hotel_topics %>%
  mutate(topic = case_when(
    topic == 1 ~ "LABEL 1",
    topic == 2 ~ "LABEL 2",
    topic == 3 ~ "LABEL 3",
    topic == 4 ~ "LABEL 4",
    topic == 5 ~ "LABEL 5",
    topic == 6 ~ "LABEL 6",
    topic == 7 ~ "LABEL 7",
    topic == 8 ~ "LABEL 8",
    topic == 9 ~ "LABEL 9",
    TRUE ~ "LABEL 10"
    )
  )
```

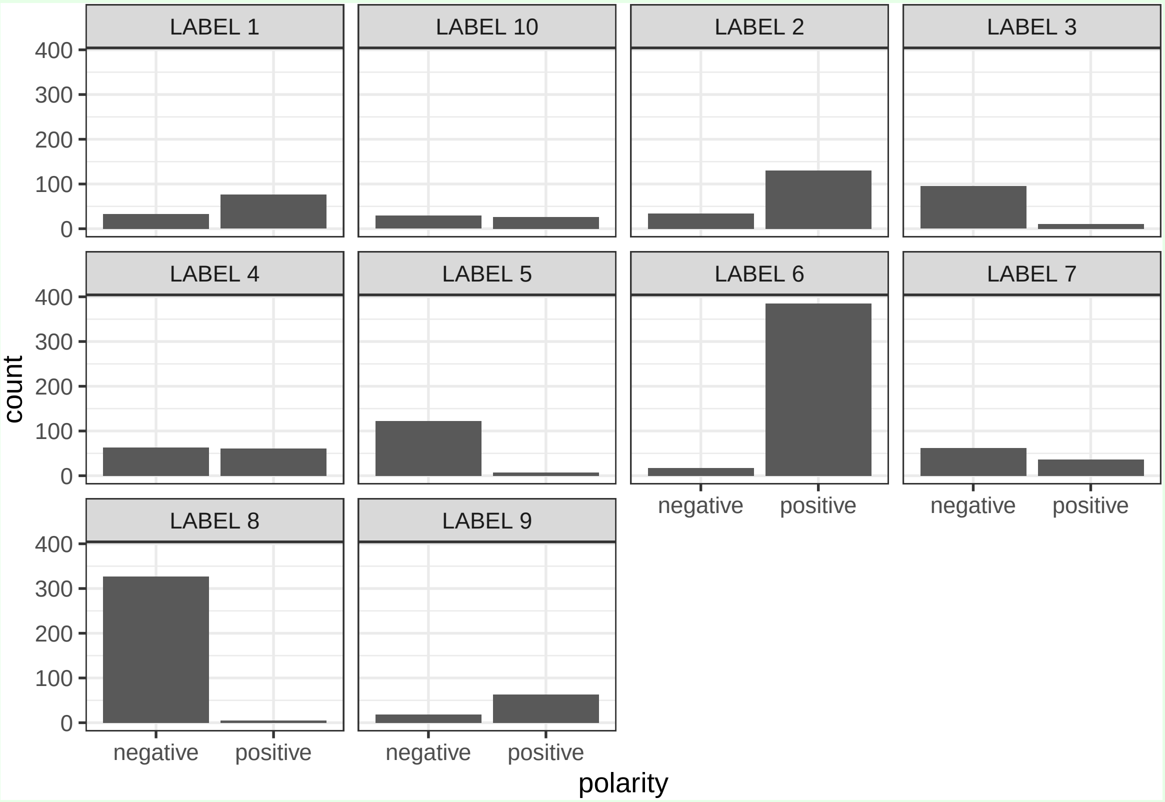
10. Create a plot that visualizes how each topic varies with the overall sentiment of the text (use the `polarity` variable for this). Discuss your findings and emphasize any managerial implications.

HINT: Your final plot should look resemble this one:



solution

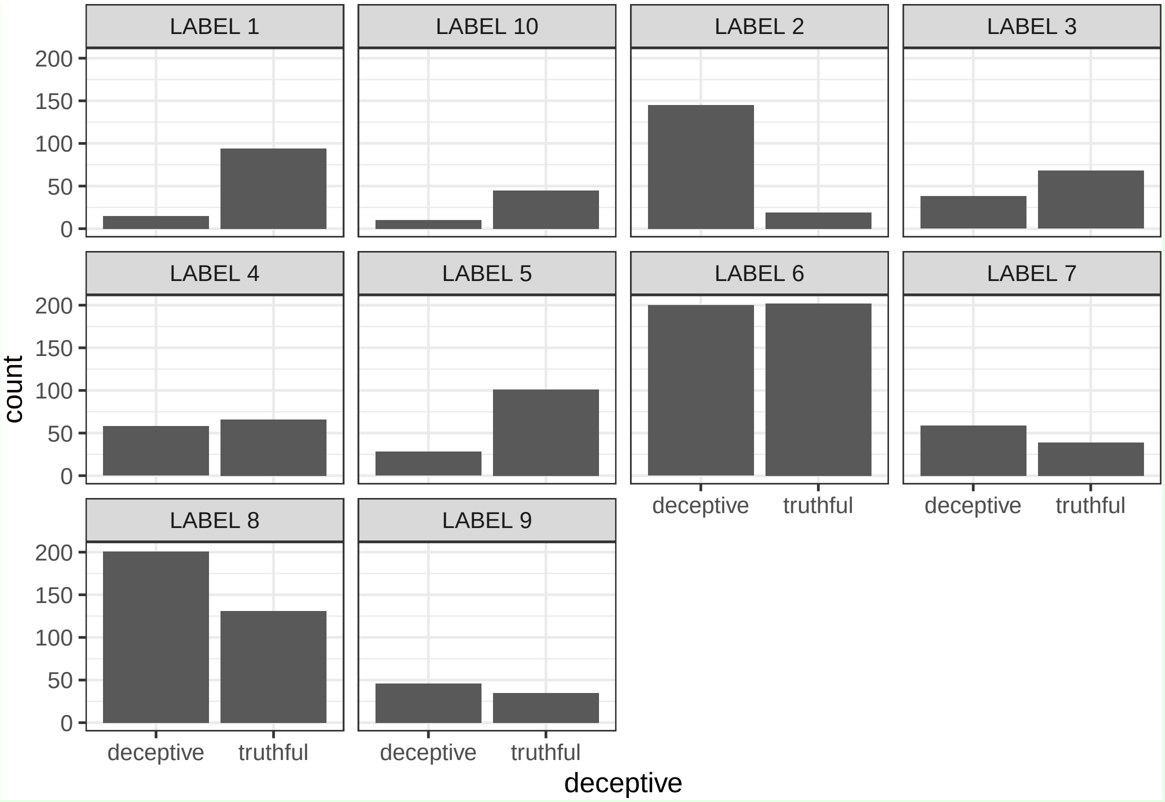
```
hotel_topics %>%
  ggplot(aes(x = polarity)) +
  geom_bar() +
  facet_wrap(~ topic) +
  theme_bw()
```



11. Create a plot that visualizes whether fake/deceptive reviews discuss different topics than truthful ones. Discuss your findings and emphasize any managerial implications regarding detecting fake reviews.

solution

```
hotel_topics %>%
  ggplot(aes(x = deceptive)) +
  geom_bar() +
  facet_wrap(~ topic) +
  theme_bw()
```


LABEL 1
LABEL 10
LABEL 2
LABEL 3
LABEL 4
LABEL 5
LABEL 6
LABEL 7
LABEL 8
LABEL 9
count
0
50
100
150
200
deceptive
truthful
deceptive