# Preparing text for modeling

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN R

**Kasey Jones**
Research Data Scientist

# Supervised learning in R: classification

# Classification modeling

- supervised learning approach
- classifies observations into categories
  - win/loss
  - dangerous, friendly, or indifferent
- can use a number of different techniques:
  - logistic regression
  - decision trees/random forest/xgboost
  - neural networks
  - etc.

# Modeling basics steps

1. Clean/prepare data
2. Create training and testing datasets
3. Train a model on the training dataset
4. Report accuracy on the testing dataset

# Character recognition

Napoloeon

Boxer

[1] https://comicvine.gamespot.com/napoleon/4005 [2] 141035/ [3] https://hero.fandom.com/wiki/Boxer_(Animal_Farm)

# Animal sentences

```
# Make sentences
sentences <- animal_farm %>%
  unnest_tokens(output = "sentence", token = "sentences", input = text_column)
```

```
# Label sentences by animal
sentences$boxer <- grepl('boxer', sentences$sentence)
sentences$napoleon <- grepl('napoleon', sentences$sentence)
```

```
# Replace the animal name
sentences$sentence <- gsub("boxer", "animal X", sentences$sentence)
sentences$sentence <- gsub("napoleon", "animal X", sentences$sentence)
animal_sentences <- sentences[sentences$boxer + sentences$napoleon == 1, ]
```

# Sentences continued

```
animal_sentences$Name <-
    as.factor(ifelse(animal_sentences$boxer, "boxer", "napoleon"))
```

```
# 75 of each
animal_sentences <-
  rbind(animal_sentences[animal_sentences$Name == "boxer", ][c(1:75), ],
        animal_sentences[animal_sentences$Name == "napoleon", ][c(1:75), ])
animal_sentences$sentence_id <- c(1:dim(animal_sentences)[1])
```

# Prepare the data

```
library(tm); library(tidytext)
library(dplyr); library(SnowballC)
```

```
animal_tokens <- animal_sentences %>%
  unnest_tokens(output = "word", token = "words", input = sentence) %>%
  anti_join(stop_words) %>%
  mutate(word = wordStem(word))
```

# Preparation continued

```
animal_matrix <- animal_tokens %>%
  count(sentence_id, word) %>%
  cast_dtm(document = sentence_id, term = word,
           value = n, weighting = tm::weightTfIdf)
animal_matrix
```

```
<<DocumentTermMatrix (documents: 150, terms: 694)>>
Non-/sparse entries: 1235/102865
Sparsity           : 99%
Maximal term length: 17
Weighting          : term frequency - inverse document frequency
```

# Remove sparse terms

- Non-empty (1,235) + empty (102,865)
- Matrix dimensions 150 * 694
- Sparsity: 102,865 / 104,100 (99%)

Solution: `removeSparseTerms()`

# How sparse is too sparse?

```
removeSparseTerms(animal_matrix, sparse = .90)
```

```
<<DocumentTermMatrix (documents: 150, terms: 4)>>
Non-/sparse entries: 207/393
Sparsity           : 66%
```

```
removeSparseTerms(animal_matrix, sparse = .99)
```

```
removeSparseTerms(animal_matrix, sparse = .99)
<<DocumentTermMatrix (documents: 150, terms: 172)>>
Non-/sparse entries: 713/25087
Sparsity           : 97%
```

# Let's practice!

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN R

# Classification modeling

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN R

**Kasey Jones**
Research Data Scientist

# Recap of the steps

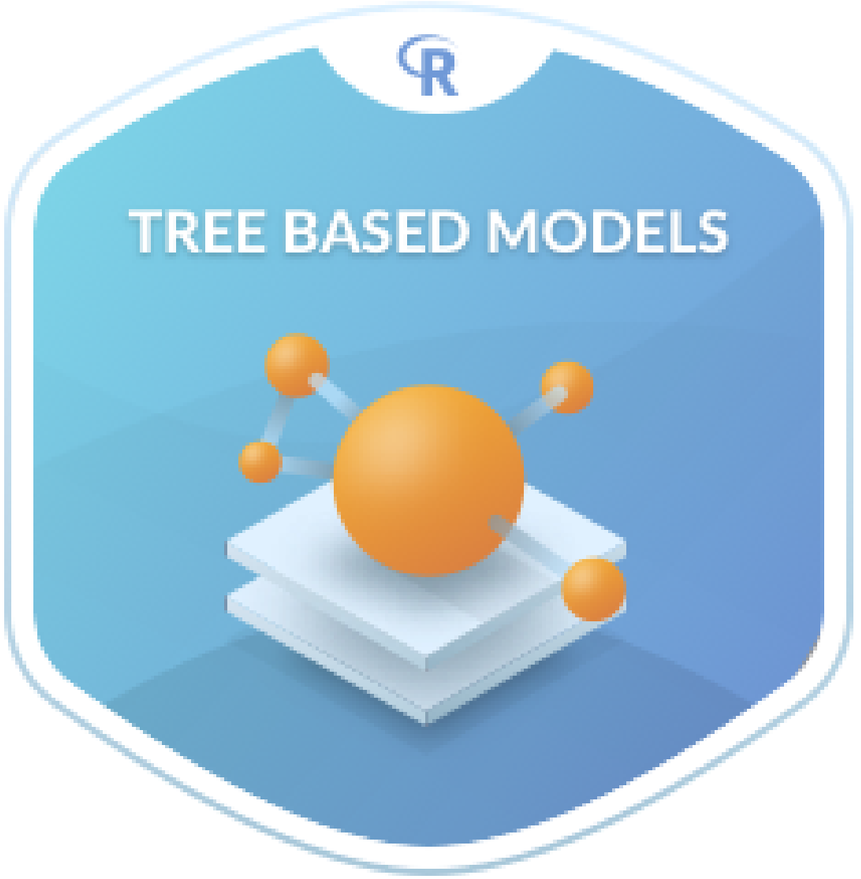
1. **Clean/prepare data**
   - Filter to Boxer/Napoleon Sentences
   - Created cleaned tokens of the words
   - Created a document-term matrix with TFIDF weighting
2. **Create training and testing datasets**
3. **Train a model on the training dataset**
4. **Report accuracy on the testing dataset**

# Step 2: split the data

```
set.seed(1111)
sample_size <- floor(0.80 * nrow(animal_matrix))
train_ind <- sample(nrow(animal_matrix), size = sample_size)
```

```
train <- animal_matrix[train_ind, ]
test <- animal_matrix[-train_ind, ]
```

# Random forest models



# Classification example

```
library(randomForest)
rfc <- randomForest(x = as.data.frame(as.matrix(train)),
                    y = animal_sentences$Name[train_ind], nTree = 50)
rfc
```

```
Call:
 randomForest(...
        OOB estimate of  error rate: 23.33%
Confusion matrix:
         boxer napoleon class.error
boxer       37       20   0.3508772
napoleon     8       55   0.1269841
```

# The confusion matrix

```
Call:
 randomForest(...
        OOB estimate of  error rate: 23.33%
Confusion matrix:
         boxer napoleon class.error
boxer       37       20   0.3508772
napoleon     8       55   0.1269841
```

Accuracy: (37 + 55) / (37 + 20 + 8 + 55) = 76%

# Test set predictions

```
y_pred <- predict(rfc, newdata = as.data.frame(as.matrix(test)))
```

```
table(animal_sentences[-train_ind, ]$Name, y_pred)
```

```
          y_pred
           boxer napoleon
  boxer       14         4
  napoleon     2        10
```

- Accuracy for boxer: 14/18
- Accuracy for napoleon: 10/12
- Overall accuracy: 24/30 = 80%

# Classification practice

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN R

# Introduction to topic modeling

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN R

**Kasey Jones**
Research Data Scientist

# Topic modeling

Sports Stories:

- scores
- player gossip
- team news
- etc.

Weather in Zambia:

- ?
- ?

# Latent dirichlet allocation

1. Documents are mixtures of topics
   - Team news 70%
   - Player Gossip 30%
2. Topics are mixtures of words
   - Team News: trade, pitcher, move, new
   - Player Gossip: angry, change, money


[1] https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation

# Preparing for LDA

Standing preparation:

```
animal_farm_tokens <- animal_farm %>%
  unnest_tokens(output = "word", token = "words", input = text_column) %>%
  anti_join(stop_words) %>%
  mutate(word = wordStem(word))
```

Document-term matrix:

```
animal_farm_matrix <- animal_farm_tokens %>%
  count(chapter, word) %>%
  cast_dtm(document = chapter, term = word,
           value = n, weighting = tm::weightTf)
```

# LDA

```
library(topicmodels)
animal_farm_lda <- LDA(train, k = 4, method = 'Gibbs',
                       control = list(seed = 1111))
animal_farm_lda
```

```
A LDA_Gibbs topic model with 4 topics.
```

# LDA results

```
animal_farm_betas <-
    tidy(animal_farm_lda, matrix = "beta")
animal_farm_betas
```

```
# A tibble: 11,004 x 3
   topic term          beta
   <int> <chr>        <dbl>
...
 5     1 abolish 0.0000360
 6     2 abolish 0.00129
 7     3 abolish 0.000355
 8     4 abolish 0.0000381
...
```

```
sum(animal_farm_betas$beta)
```

```
[1] 4
```

# Top words per topic

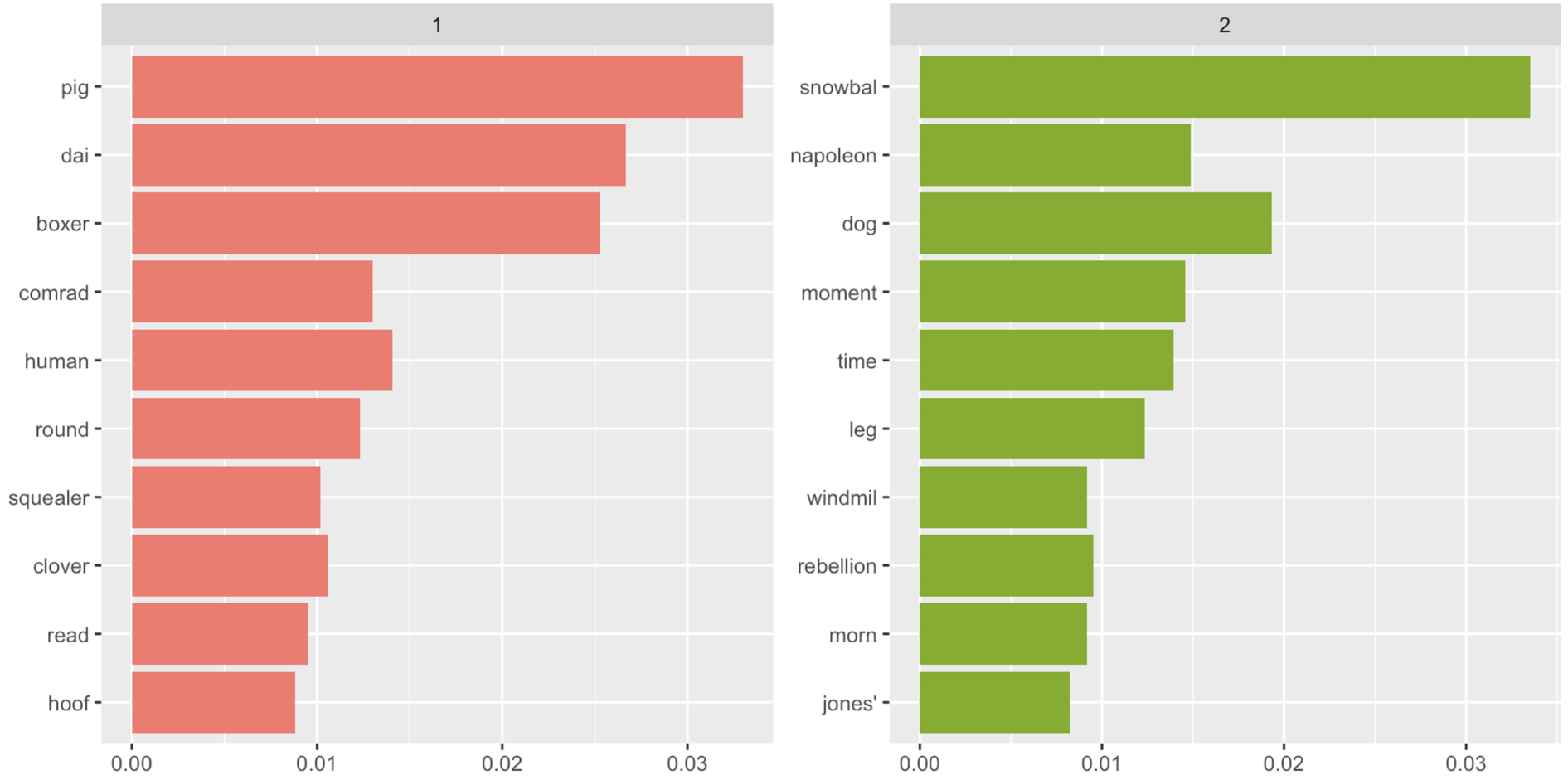
```
animal_farm_betas %>%
  group_by(topic) %>%
  top_n(10, beta) %>%
  arrange(topic, -beta) %>%
  filter(topic == 1)
```

```
  topic term           beta
  <int> <chr>         <dbl>
1     1 napoleon    0.0339
2     1 anim        0.0317
3     1 windmil     0.0144
4     1 squealer    0.0119
...
```

```
animal_farm_betas %>%
  group_by(topic) %>%
  top_n(10, beta) %>%
  arrange(topic, -beta) %>%
  filter(topic == 2)
```

```
  topic term         beta
  <int> <chr>       <dbl>
...
3     2 anim      0.0189
...
6     2 napoleon 0.0148
...
```

# Top words continued




[1] https://www.tidytextmining.com/topicmodeling.html

# Labeling documents as topics

```
animal_farm_chapters <- tidy(animal_farm_lda, matrix = "gamma")
animal_farm_chapters %>%
  filter(document == 'Chapter 1')
```

```
# A tibble: 4 x 3
  document  topic  gamma
  <chr>     <int>  <dbl>
1 Chapter 1     1 0.157
2 Chapter 1     2 0.136
3 Chapter 1     3 0.623
4 Chapter 1     4 0.0838
```

# LDA practice!

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN R

# LDA in practice

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN R

**Kasey Jones**
Research Data Scientist

# Finalizing LDA results

- select the number of topics
  - perplexity/other metrics
  - a solution that works for your situation

# Perplexity

- measure of how well a probability model fits new data
- lower is better
- used to compare models
  - In LDA parameter tuning
  - Selecting number of topics

```
sample_size <- floor(0.90 * nrow(doc_term_matrix))
set.seed(1111)
train_ind <- sample(nrow(doc_term_matrix), size = sample_size)
train <- matrix[train_ind, ]
test <- matrix[-train_ind, ]
```

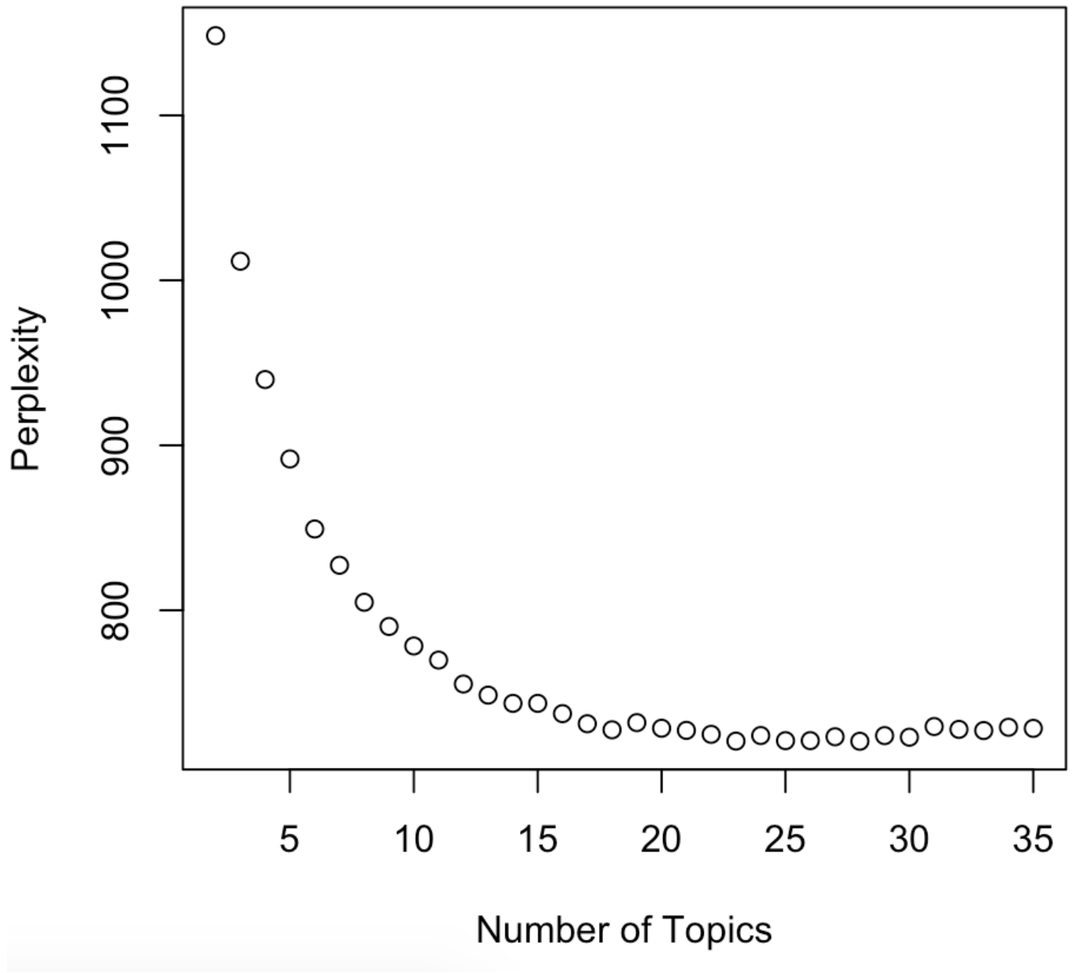
[1] https://en.wikipedia.org/wiki/Perplexity

# Perplexity in R

```
library(topicmodels)
values = c()
for(i in c(2:35)){
  lda_model <- LDA(train, k = i, method = "Gibbs",
                   control = list(iter = 25, seed = 1111))
  values <- c(values, perplexity(lda_model, newdata = test))
}
```

```
plot(c(2:35), values, main="Perplexity for Topics",
     xlab="Number of Topics", ylab="Perplexity")
```

# Perplexity again!



# Practical selection

- How many topics can the situation handle
  - 20 might be difficult to cover
- How are you displaying the results
  - Graphics with 5 topics are easier than graphics with 100 topics
- Rules of thumb:
  - Use a small number of topics where each topic is represented by several documents
  - Large topic counts can be used only if time allows exploring and dissecting each topic

# Using results

- Review or have reviewers find "themes" for each topic
  - provide reviewer with a list of top words in the topic
  - provide reviewer with a list of the top documents for that topic

# Review output

```
betas <- tidy(lda_model, matrix = "beta")
betas %>%
  filter(topic == 1) %>%
  arrange(desc(beta)) %>%
  select(term)
```

```
# A tibble: 2,000 x 1
   term
   <chr>
 1 athletic
 2 quick
 3 strong
 4 tough
...
```

```
gammas <- tidy(lda_model, matrix = "gamma")
gammas %>%
  filter(topic == 1) %>%
  arrange(desc(gamma)) %>%
  select(document)
```

```
# A tibble: 1,000 x 1
   document
   <chr>
 1 232
 2 292
 3 921
 4 643
 5 468
```

# Summarize output

```
gammas <- tidy(lda_model, matrix = "gamma")
gammas %>%
  group_by(document) %>%
  arrange(desc(gamma)) %>%
  slice(1) %>%
  group_by(topic) %>%
  tally(topic, sort=TRUE)
```

```
  topic     n
1     1  1326
2     5  1215
3     4   804
...
```

# Summarize output again

```
gammas %>%
  group_by(document) %>%
  arrange(desc(gamma)) %>%
  slice(1) %>%
  group_by(topic) %>%
  summarize(avg=mean(gamma)) %>%
  arrange(desc(avg))
```

```
  topic   avg
1     1 0.696
2     5 0.530
3     4 0.482
...
```

# LDA practice.

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN R