Text Analysis

Language is complex. The goal of text analysis is to strip away some of that complexity to extract meaning.
How to talk like a Democrat (or a Republican)
Reddit N-gram Viewer: FiveThirtyEight created “How the Internet Talks,” a tool to visualize the prevalence of terms on Reddit

Image Source
Language of the **alt-right**

![Graphs showing similarity scores of words similar to "Jewish" in mainstream and alt-right language.](Image Source)
Power and Agency in Hollywood Characters

A team at UW analyzed the language in nearly 800 movie scripts, quantifying how much power and agency those scripts give to individual characters. Women consistently used more submissive language, with less agency.

“In the movie Frozen, only the princess Elsa is portrayed with high power and positive agency, according to a new analysis of gender bias in movies. Her sister, Anna, is portrayed with similarly low levels of power as 1950s-era Cinderella.
Text Preprocessing
Text Preprocessing

Definition: a set of documents is called a corpus.

The first step in text analysis is preprocessing (cleaning) the corpus:

- **Tokenize**: parse documents into smaller units, such as words or phrases
- **Stemming & Lemmatization**: standardize words with similar meaning
- **Remove stop words** (e.g., a, the, and, etc.) and punctuation
- **Normalize**: convert to lowercase (carefully: e.g., US vs. us)
Tokenization

Bag-of-Words Model
Represents a corpus as an unordered set of word counts, ignoring stop words

Doc 1: David likes to play soccer. Ben likes to play tennis.
Doc 2: Mary likes to ride her bike.
Doc 3: David likes to go to the movie theater.
Tokenization

N-gram Model
An N-gram is a sequence of N words in a corpus.

The movie about the White House was not popular.

N=1 (unigram, bag-of-words): The, movie, about, the, White, House, was, not, popular

N=2 (bigram): The movie, movie about, about the, the White, White House, House was, was not, not popular

N=3 (trigram): The movie about, movie about the, about the White, the White House, White House was, House was not, was not popular

N=4 …
Stemming & Lemmatization

Goal: standardize words with a similar meaning

**Stemming** reduces words to their base, or root, form

**Lemmatization** makes words grammatically comparable (e.g., am, are, is ➔ be)

He ate a *tasty* cookie yesterday, and he is eating *tastier* cookies today.

He ate a *tasty* cookie yesterday, and he is *eat* *tasti* cookie today.

He *eat* a *tasty* cookie yesterday, and he is *eat* *tasty* cookie today.
Normalization

Examples:
- make all words lowercase
- remove any punctuation
- remove unwanted tags

Has Dr. Bob called? He is waiting for the results of Experiment #6.

has dr bob called he is waiting for the results of experiment 6

<p>Text.</p><!-- Comment --> <a href="#breakpoint">More text.</a>'

text more text
A Final Note

Preprocessing should be customized based on the type of corpus.

● Tweets should be preprocessed differently than academic texts.
  ○ The Reddit N-gram Viewer
● So should the names of bands: e.g., The The.
Regular Expressions
Regular Expressions (**Regex**)

*Regular expressions* are a handy tool for searching for patterns in text.

You can think of them as a fancy form of “find and replace”.

- In R, we can use `grep` to find a pattern in text:
  - `grep(regex, text)`

- And, we can use `gsub` to replace a pattern in text:
  - `gsub(regex, replacement, text)`
Regular Expressions

Consider a literature corpus, some written in American English, others in British English.

Let’s find the word “color,” which is equivalent to “colour.”

We have a few options: e.g.,

```python
grep("color|colour", text)
grep("colou?r", text)
```

| means “or”, and ? in this context means the preceding character is optional.

We also want to find “theater” and “theatre.”

```python
grep("theat(er|re)", text)
```
Regular Expressions

Next, let’s find words that rhyme with “light.”

```
grep("[a-zA-Z]+(ight|ite)", text)
```

`[a-zA-Z]` matches any letter

`+` matches 1 or more of the preceding characters

Tonight I might write a story about a knight with a snakebite.

```r
> text <- "Tonight I might write a story about a knight with a snakebite."
> text_vec <- unlist(strsplit(text, split = "\[\]"))
> grep("[a-zA-Z]+(ight|ite)", text_vec)
[1]  1  3  4  9 12
> text_vec[grep("[a-zA-Z]+(ight|ite)", text_vec)]
[1] "Tonight" "might" "write" "knight" "snakebite."
```
Regular Expressions

Let’s try numbers. For example, let’s find Rhode Island zip codes. Hint: they start with 028 or 029.

```
grep("02(8|9)[0-9]{2}\", text)
```

- `[0-9]` matches any digit
- `{2}` matches exactly 2 of the preceding characters

69 Brown Street, Providence, RI 02912

424 Bellevue Ave, Newport, RI 02840
Regular Expressions

Here’s how we might find all instances of parents, grandparents, great grandparents, and so on.

```
grep("((great )\*grand)?((fa|m)o)ther)\", text)
```

* captures 0 or more of the preceding characters

? in this context means the preceding expression is optional

My mother, father, grandfather, grandmother, great great grandmother, and great great great great grandfather were all born in Poland.
Text Visualizations
Word Clouds

Visualizes words in a document with sizes proportional to how frequently the words are used.

Example: *The Great Gatsby*
freedom will let ring day
2012 Democratic and Republican Conventions

← Words favored by Democrats

Words favored by Republicans →
Google Books Ngram Viewer: Charts word frequencies in books over time, offering insight into how language, culture, and literature have changed.
Text Analysis
Text Classification

- Who wrote the Federalist papers (anonymous essays in support of the U.S. constitution)?
  - John Jay, James Madison, or Alexander Hamilton?
- Topic modelling: assign topics (politics, sports, fashion, finance, etc.) to documents (e.g., articles or web pages)
- Spam detection
Naive Bayes Text Classification

Procedure

- For all classes $y$, calculate $\prod_i P(X_i | Y) P(Y = y)$
- Choose a class $y$ s.t. $\prod_i P(X_i | Y) P(Y = y)$ is maximal

### n-gram

<table>
<thead>
<tr>
<th>n-gram</th>
<th>Spam</th>
<th>Ham</th>
</tr>
</thead>
<tbody>
<tr>
<td>hello</td>
<td>.30</td>
<td>.33</td>
</tr>
<tr>
<td>friend</td>
<td>.08</td>
<td>.23</td>
</tr>
<tr>
<td>password</td>
<td>.28</td>
<td>.03</td>
</tr>
<tr>
<td>money</td>
<td>.40</td>
<td>.12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Spam</th>
<th>Ham</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spam</td>
<td>.1</td>
<td></td>
</tr>
<tr>
<td>Ham</td>
<td>.9</td>
<td></td>
</tr>
</tbody>
</table>
Naive Bayes Text Classification

An email that contains the words hello and friend, but not money and password:

- **Spam**: \( P(\text{hello} \mid \text{spam}) \cdot P(\text{friend} \mid \text{spam}) \cdot P(\text{spam}) = (.30)\cdot(.05)\cdot(.1) = 0.0015 \)
- **Ham**: \( P(\text{hello} \mid \text{ham}) \cdot P(\text{friend} \mid \text{ham}) \cdot P(\text{ham}) = (.33)\cdot(.25)\cdot(.9) = 0.07425 \)

An email that contains the words hello, money, and password:

- **Spam**: \( (.30)\cdot(0.20)\cdot(0.40)\cdot(.1) = 0.0024 \)
- **Ham**: \( (.33)\cdot(0.02)\cdot(0.10)\cdot(.9) = 0.000594 \)

<table>
<thead>
<tr>
<th>n-gram</th>
<th>Spam</th>
<th>Ham</th>
</tr>
</thead>
<tbody>
<tr>
<td>hello</td>
<td>.30</td>
<td>.33</td>
</tr>
<tr>
<td>friend</td>
<td>.05</td>
<td>.25</td>
</tr>
<tr>
<td>password</td>
<td>.20</td>
<td>.02</td>
</tr>
<tr>
<td>money</td>
<td>.40</td>
<td>.10</td>
</tr>
</tbody>
</table>

Spam: .1
Ham: .9
Text/Document Clustering

Biomedical Articles
Natural Language Generation: Image Captions

A person riding a motorcycle on a dirt road.

A group of young people playing a game of frisbee.

A herd of elephants walking across a dry grass field.

Two dogs play in the grass.

Two hockey players are fighting over the puck.

A close up of a cat laying on a couch.

A skateboarder does a trick on a ramp.

A little girl in a pink hat is blowing bubbles.

A red motorcycle parked on the side of the road.

A dog is jumping to catch a frisbee.

A refrigerator filled with lots of food and drinks.

A yellow school bus parked in a parking lot.

Figure 5. A selection of evaluation results, grouped by human rating.

Image Source
Natural Language Generation: Descriptions

E.g., Textual descriptions of quantitative geographical and hydrological sensor data.
Natural Language Generation: Humor

Researchers developed a language model to generate jokes of the form “I like my X like I like my Y, Z”

- E.g., “I like my coffee like I like my war, cold.”
- After testing, they claimed: “Our model generates funny jokes 16% of the time, compared to 33%, for human-generated jokes.”

There are also language models that generate puns:

- E.g., "What do you call a spicy missile? A hot shot!"
Automatic Document Summarization

Automatically summarize documents (e.g., news articles or research papers).

Ben and Ally drove their car to go grocery shopping. They bought bananas, watermelon, and a bag of lemons and limes.

1. **Extraction**: copying words or phrases that are deemed interesting by some metric; often results in clumsy or grammatically-incorrect summaries: Ben and Ally go grocery shopping. Bought bananas, watermelon, and bag.

2. **Abstraction**: paraphrasing; results similar to human speech, but requires complex language modeling; active area of research at places like Google Ben and Ally bought fruit at the grocery store.
Sentiment Analysis
Sentiment Analysis

- Classifies a document as expressing a positive, negative, or neutral opinion.
- Especially useful for analyzing reviews (for products, restaurants, etc.) and social media posts (tweets, Facebook posts, blogs, etc.).

Example:

![Yelp Review](image)

Mama Kim’s Korean BBQ

5/12/2015

I have been to mama Kim’s several times and every time it’s been delicious! The beef bulgogi slides are amazing and so are the sweet potato fries!! If the dumplings are on special definitely try them!!
Twitter Data

KANYE WEST
@kanyewest

McDonalds is my favorite brand

Ellen DeGeneres
@TheEllenShow

Congratulations, @SerenaWilliams! This tweet is just in case you couldn't hear me scream it from the stands. #Wimbledon.

Ariana Grande
@ArianaGrande

I highly recommend all of my fans watch #Blackfish and never go to @Seaworld again. 😢 Used to love that place. Beyond heartbroken #FreeTilly
Researchers have built lists of words with “positive” and “negative” connotations. For each chunk of our own text, we can calculate how many words lie in these “positive” or “negative” groupings.

I love all the delicious free food in the CIT, but working in the Sunlab makes me sad.
Some Challenges

Positive words that contrast with an overall negative message (and vice versa)
“I enjoyed the old version of this app, but I hate the newer version.”

Selecting the proper $N$-gram
• “This product is not reliable”
• “This product is unreliable”
If unigrams are used, “not reliable” will be split into “not” and “reliable,”
which could result in a neutral sentiment.

Sarcasm
“I loved having the fly in my soup! It was delicious!”
Sentiment Analysis of Tweets

1. Download tweets from twitter.com
2. Preprocess text
   a. Remove emojis and URLs
   b. Remove punctuation (e.g., hashtags)
   c. Split sentences into words; convert to lowercase; etc.
3. Feed words into a model: e.g., bag-of-words
4. Add common Internet slang to lists of “positive” and “negative” words: e.g., “luv”, “yay”, “ew”, “wtf”
5. Count how many words per tweet are positive, neutral, or negative
6. Score each tweet based on counts (positive = +1; neutral = 0; negative = -1)

Comment from a student: Emojis are informative. Might do better if they are used.
Starbucks’ Tweets

Using the R package `twitteR`, we can directly access Twitter data. Here’s how to access the 5000 most recent tweets about Starbucks in R:

```
starbucks_tweets <- searchTwitter('@Starbucks', n = 5000)
```
Starbucks’ Tweets (cont’d)

Here’s an example of 3 tweets that were returned:

"Wish @Starbucks would go back to fresh baked goods instead if the pre-packaged. #sad #pastries"

“Huge shout out: exemplary service from Emile @starbucks I left with a smile. @ Starbucks Canada https://t.co/WtXjeekCT1"

"Currently very angry at @Starbucks, for being out of their S'mores frap at seemingly every location \xed\xa0\xbd\xed\xb8\xa1"
Starbucks’ Tweets (cont’d)

Remove emojis: `starbucks_tweets <- iconv(starbucks_tweets, 'UTF-8', 'ASCII', sub = " ")`

Remove punctuation: `starbucks_tweets <- gsub('[[:punct:]]', ' ', starbucks_tweets)`

Remove URLs: `starbucks_tweets <- gsub('http.* *', ' ', starbucks_tweets)`

Convert to lowercase: `starbucks_tweets <- tolower(starbucks_tweets)`

"Wish @Starbucks would go back to fresh baked goods instead if the pre-packaged. #sad #pastries"

“Huge shout out: exemplary service from Emile @starbucks I left with a smile. @ Starbucks Canada https://t.co/WtXjeekCT1"

"Currently very angry at @Starbucks, for being out of their S'mores frap at seemingly every location \xed\xa0\xbd\xed\xb8\xa1"

"wish starbucks would go back to fresh baked goods instead if the prepackaged sad pastries"

"huge shout out exemplary service from emile starbucks i left with a smile starbucks canada "

"currently very angry at starbucks for being out of their smores frap at seemingly every location "
Next, we load lists of pre-determined positive and negative words (downloaded from the Internet):

```r
pos <- scan("/Downloads/positive-words.txt", what = "character", comment.char = ";")
neg <- scan("/Downloads/negative-words.txt", what = "character", comment.char = ";")
```

We add some informal terms of our own:

```r
pos <- c(pos, 'perf', 'luv', 'yum', 'epic', 'yay')
neg <- c(neg, 'wtf', 'ew', 'yuck', 'icky')
```
Next, we split our tweets into individual words.

```
starbucks_words = str_split(starbucks_tweets, ' ')
```

We then compare our words to the positive and negative terms.

```
match(starbucks_words, pos)
match(starbucks_words, neg)
```

"wish starbucks would go back to fresh baked goods instead if the prepackaged sad pastries"
Score: 0 (here we see limitations of this technique)

"huge shout out exemplary service from emile starbucks i left with a smile starbucks canada"
Score: +2

"currently very angry at starbucks for being out of their smores frap at seemingly every location "
Score: -1
Sentiment Analysis

Average Sentiment Per Tweet

- Coca Cola
- Pepsi
- Burger King
- McDonald's
- Dunkin' Donuts
- Starbucks
Sentiment Analysis

We can see that sentiment analysis can give a business insight into public opinion on its products and service. It can also reveal how consumers feel about a business compared to competing brands.

Businesses can also collect tweets over time, and see how sentiment changes. Can then try to build a causal model, using data about ad campaigns, new product releases, etc.
Extras
Text Analysis

The process of computationally retrieving information from text, such as books, articles, emails, speeches, and social media posts.

Analyzes word frequency, distribution, patterns, and meaning.
Images of the alt-right
Map of alt-right Twitter accounts