Text Mining, pt. I

Philip D. Waggoner

MACS 40500: Computational Methods for American Politics

November 19, 2019

Lecture Outline

Lecture Outline

Text Mining

- The goal of this week is to offer a well-rounded toolbox for text mining, which includes both supervised and unsupervised techniques

Text Mining

- The goal of this week is to offer a well-rounded toolbox for text mining, which includes both supervised and unsupervised techniques
- Today: **supervised** (basics, dictionaries, and sentiment scoring)

Text Mining

- The goal of this week is to offer a well-rounded toolbox for text mining, which includes both supervised and unsupervised techniques
- Today: **supervised** (basics, dictionaries, and sentiment scoring)
- Thursday: **unsupervised** (topic models)

Text Mining

- For most of its history text analysis was qualitative

Text Mining

- For most of its history text analysis was qualitative
- And today?

Text Mining

- For most of its history text analysis was qualitative
- And today? We are still interested in appying subjective judgement to explain and judge that which text is revealing

Text Mining

- For most of its history text analysis was qualitative
- And today? We are still interested in appying subjective judgement to explain and judge that which text is revealing
- Thus, our goal is *distant reading*, rather than *close reading*

Text Mining

- For most of its history text analysis was qualitative
- And today? We are still interested in appying subjective judgement to explain and judge that which text is revealing
- Thus, our goal is *distant reading*, rather than *close reading*
- The “quantification” of text analysis is a monumental advance in this ancient field in that quantitative work is reliable, replicable, and can easily handle large volumes of material

A General Process

1. Obtain some group of texts

A General Process

1. Obtain some group of texts
2. Create a document-term matrix

A General Process

1. Obtain some group of texts
2. Create a document-term matrix
3. Operate/mine ((dis)similarities, sentiments, topics, complexity, classification, etc.)

A General Process

1. Obtain some group of texts
2. Create a document-term matrix
3. Operate/mine ((dis)similarities, sentiments, topics, complexity, classification, etc.)
4. Draw inferences

Defining a Corpus

- Some set of texts or documents we wish to analyze, which can be

Defining a Corpus

- Some set of texts or documents we wish to analyze, which can be
  - *structured*: useful document information is known (e.g., beginning and end, authorship, etc.)

Defining a Corpus

- Some set of texts or documents we wish to analyze, which can be
  - *structured*: useful document information is known (e.g., beginning and end, authorship, etc.)
  - *unstructured*: desired quantity is unknown (e.g., sentiment)

Defining a Corpus

- Some set of texts or documents we wish to analyze, which can be
  - *structured*: useful document information is known (e.g., beginning and end, authorship, etc.)
  - *unstructured*: desired quantity is unknown (e.g., sentiment)
- Further, a corpus may be annotated where metadata – *data not part of the document itself* – is available, e.g., date, linguistic tagging, etc.

Sampling

- The corpus is made up of the documents within it, but these may be a sample of the total population of documents available

Sampling

- The corpus is made up of the documents within it, but these may be a sample of the total population of documents available
- Often sample due to time, resources or (*legal*) necessity,

Sampling

- The corpus is made up of the documents within it, but these may be a sample of the total population of documents available

- Often sample due to time, resources or (*legal*) necessity, e.g. Twitter gives you $\sim$ 1% of all their tweets, but it would presumably be prohibitively expensive to store 100%

Sampling

- The corpus is made up of the documents within it, but these may be a sample of the total population of documents available
- Often sample due to time, resources or (*legal*) necessity, e.g. Twitter gives you $\sim 1\%$ of all their tweets, but it would presumably be prohibitively expensive to store 100%
- Sometimes, authors claim to have the universe of cases in their corpus...

Sampling

- The corpus is made up of the documents within it, but these may be a sample of the total population of documents available
- Often sample due to time, resources or (*legal*) necessity, e.g. Twitter gives you $\sim 1\%$ of all their tweets, but it would presumably be prohibitively expensive to store 100%
- Sometimes, authors claim to have the universe of cases in their corpus...
- But, *you still need to think about sampling error*

Sampling

- The corpus is made up of the documents within it, but these may be a sample of the total population of documents available

- Often sample due to time, resources or (*legal*) necessity, e.g. Twitter gives you $\sim 1\%$ of all their tweets, but it would presumably be prohibitively expensive to store 100%

- Sometimes, authors claim to have the universe of cases in their corpus...

- But, *you still need to think about sampling error* $\rightsquigarrow$ there exists a *super*population of populations from which the universe you observed came from

Sampling

- The corpus is made up of the documents within it, but these may be a sample of the total population of documents available
- Often sample due to time, resources or (*legal*) necessity, e.g. Twitter gives you $\sim 1\%$ of all their tweets, but it would presumably be prohibitively expensive to store 100%
- Sometimes, authors claim to have the universe of cases in their corpus...
- But, *you still need to think about sampling error* $\rightsquigarrow$ there exists a *super*population of populations from which the universe you observed came from
- The point here?

Sampling

- The corpus is made up of the documents within it, but these may be a sample of the total population of documents available
- Often sample due to time, resources or (*legal*) necessity, e.g. Twitter gives you $\sim 1\%$ of all their tweets, but it would presumably be prohibitively expensive to store 100%
- Sometimes, authors claim to have the universe of cases in their corpus...
- But, *you still need to think about sampling error* $\rightsquigarrow$ there exists a *super*population of populations from which the universe you observed came from
- The point here? Be cautious as you approach text mining and think carefully about error, where texts came from, and so on, as the corpus *should be representative* in some sense for inferences to be meaningful

Reducing Complexity

- Language is extraordinarily complex, and involves great subtlety and nuanced interpretation

Reducing Complexity

- Language is extraordinarily complex, and involves great subtlety and nuanced interpretation

- Yet remarkably, we can *do very well* by simplifying, and representing documents as mathematical objects, making the modeling process much more tractable (more at the end)

Reducing Complexity

- Language is extraordinarily complex, and involves great subtlety and nuanced interpretation
- Yet remarkably, we can *do very well* by simplifying, and representing documents as mathematical objects, making the modeling process much more tractable (more at the end)
- 'Do very well' $\rightsquigarrow$ complicated representations add nothing to the quality of inferences, ability to predict outcomes, and model fit

Reducing Complexity

- Language is extraordinarily complex, and involves great subtlety and nuanced interpretation
- Yet remarkably, we can *do very well* by simplifying, and representing documents as mathematical objects, making the modeling process much more tractable (more at the end)
- 'Do very well' $\rightsquigarrow$ complicated representations add nothing to the quality of inferences, ability to predict outcomes, and model fit
- Inevitably, the degree to which we simplify is dependent on the task at hand;

Reducing Complexity

- Language is extraordinarily complex, and involves great subtlety and nuanced interpretation

- Yet remarkably, we can *do very well* by simplifying, and representing documents as mathematical objects, making the modeling process much more tractable (more at the end)

- 'Do very well' $\rightsquigarrow$ complicated representations add nothing to the quality of inferences, ability to predict outcomes, and model fit

- Inevitably, the degree to which we simplify is dependent on the task at hand; there is no "single best way" to go from texts to numeric data

Reducing Complexity

- Language is extraordinarily complex, and involves great subtlety and nuanced interpretation
- Yet remarkably, we can *do very well* by simplifying, and representing documents as mathematical objects, making the modeling process much more tractable (more at the end)
- 'Do very well' $\rightsquigarrow$ complicated representations add nothing to the quality of inferences, ability to predict outcomes, and model fit
- Inevitably, the degree to which we simplify is dependent on the task at hand; there is no "single best way" to go from texts to numeric data
- Though different in means, the end is the same in text mining: **reduce complexity for inferential clarity**

Quick Note on Terminology

- a **type** is a *unique* sequence of characters that are grouped together in some meaningful way (usually a word)
  - e.g. 'Australia', 'French Revolution', '1984'

Quick Note on Terminology

- a **type** is a *unique* sequence of characters that are grouped together in some meaningful way (usually a word)
  - e.g. 'Australia', 'French Revolution', '1984'
- a **token** is an instance of **type**
  - e.g. "Dog eat dog world", contains three **types**, but four **tokens**

Quick Note on Terminology

- a **type** is a *unique* sequence of characters that are grouped together in some meaningful way (usually a word)
  - e.g. 'Australia', 'French Revolution', '1984'
- a **token** is an instance of **type**
  - e.g. "Dog eat dog world", contains three **types**, but four **tokens**
- a **term** is a **type** that the technique recognizes as a *type to be recorded*
  - e.g. stemmed words like 'motivat' or 'applica'

The Mechanics of Moving from Text to Numeric Data

1. Collect raw text in machine-readable form; *decide what constitutes a document*

The Mechanics of Moving from Text to Numeric Data

1. Collect raw text in machine-readable form; *decide what constitutes a document*
2. Strip away extraneous material, e.g., ASCII characters, capitalization, punctuation, stop words, and so on

The Mechanics of Moving from Text to Numeric Data

1. Collect raw text in machine-readable form; *decide what constitutes a document*
2. Strip away extraneous material, e.g., ASCII characters, capitalization, punctuation, stop words, and so on
3. Cut document up into useful elementary pieces: tokenization

The Mechanics of Moving from Text to Numeric Data

1. Collect raw text in machine-readable form; *decide what constitutes a document*
2. Strip away extraneous material, e.g., ASCII characters, capitalization, punctuation, stop words, and so on
3. Cut document up into useful elementary pieces: tokenization
4. Map tokens to common form: lemmatization, stemming

The Mechanics of Moving from Text to Numeric Data

1. Collect raw text in machine-readable form; *decide what constitutes a document*
2. Strip away extraneous material, e.g., ASCII characters, capitalization, punctuation, stop words, and so on
3. Cut document up into useful elementary pieces: tokenization
4. Map tokens to common form: lemmatization, stemming
5. (Sometimes) Add descriptive annotations that preserve text context: tagging

The Mechanics of Moving from Text to Numeric Data

1. Collect raw text in machine-readable form; *decide what constitutes a document*
2. Strip away extraneous material, e.g., ASCII characters, capitalization, punctuation, stop words, and so on
3. Cut document up into useful elementary pieces: tokenization
4. Map tokens to common form: lemmatization, stemming
5. (Sometimes) Add descriptive annotations that preserve text context: tagging
6. Operate/model/mine

The Mechanics of Moving from Text to Numeric Data

- We can compress these steps a bit...

The Mechanics of Moving from Text to Numeric Data

- We can compress these steps a bit...
  1. Collect raw text in machine readable form; *decide what constitutes a document*

The Mechanics of Moving from Text to Numeric Data

- We can compress these steps a bit...

  1. Collect raw text in machine readable form; *decide what constitutes a document*
  2. **Preprocess Stage**

The Mechanics of Moving from Text to Numeric Data

- We can compress these steps a bit...

  1. Collect raw text in machine readable form; *decide what constitutes a document*
  2. **Preprocess Stage**
  3. Operate/model/mine

Stripping away extraneous material

- So you've collected some text and got it into electronic format

Stripping away extraneous material

- So you've collected some text and got it into electronic format
- Now, we have the first preprocessing step

Stripping away extraneous material

- So you've collected some text and got it into electronic format
- Now, we have the first preprocessing step
- Generally think control characters here $\rightsquigarrow$ if they do not contribute to substantive importance then remove them

Stripping away extraneous material

- So you've collected some text and got it into electronic format
- Now, we have the first preprocessing step
- Generally think control characters here $\rightsquigarrow$ if they do not contribute to substantive importance then remove them
- Punctuation may also be unhelpful, e.g.,

Stripping away extraneous material

- So you've collected some text and got it into electronic format
- Now, we have the first preprocessing step
- Generally think control characters here $\rightsquigarrow$ if they do not contribute to substantive importance then remove them
- Punctuation may also be unhelpful, e.g.,
- are, `wash`, `wash.`, `wash,`, `wash)` really different *words*?

Stop Words

- **stopwords** serve as linguistic connectors and can be removed at this first preprocessing stage

Stop Words

- **stopwords** serve as linguistic connectors and can be removed at this first preprocessing stage
- Removing stopwords simplifies the document with little information loss, e.g., they are ignored by search engines

Stop Words

- **stopwords** serve as linguistic connectors and can be removed at this first preprocessing stage
- Removing stopwords simplifies the document with little information loss, e.g., they are ignored by search engines
- So what are stopwords?

Stop Words

- **stopwords** serve as linguistic connectors and can be removed at this first preprocessing stage
- Removing stopwords simplifies the document with little information loss, e.g., they are ignored by search engines
- So what are stopwords? compiled lists in any NLP package (e.g., in NLTK, `https://gist.github.com/sebleier/554280` or the `stopwords` library in R)

Stop Words

- **stopwords** serve as linguistic connectors and can be removed at this first preprocessing stage
- Removing stopwords simplifies the document with little information loss, e.g., they are ignored by search engines
- So what are stopwords? compiled lists in any NLP package (e.g., in NLTK, `https://gist.github.com/sebleier/554280` or the `stopwords` library in R)
- But stopwords can also vary by application/domain-specific needs

Stop Words

- **stopwords** serve as linguistic connectors and can be removed at this first preprocessing stage
- Removing stopwords simplifies the document with little information loss, e.g., they are ignored by search engines
- So what are stopwords? compiled lists in any NLP package (e.g., in NLTK, `https://gist.github.com/sebleier/554280` or the `stopwords` library in R)
- But stopwords can also vary by application/domain-specific needs
- e.g. with Congressional speech data, `representative` might be a stop word; in British Parliamentary data, `honourable` might be

Stop Words

- **stopwords** serve as linguistic connectors and can be removed at this first preprocessing stage
- Removing stopwords simplifies the document with little information loss, e.g., they are ignored by search engines
- So what are stopwords? compiled lists in any NLP package (e.g., in NLTK, `https://gist.github.com/sebleier/554280` or the `stopwords` library in R)
- But stopwords can also vary by application/domain-specific needs
- e.g. with Congressional speech data, `representative` might be a stop word; in British Parliamentary data, `honourable` might be
- And inversely, there might be unique cases where stopwords should actually be retained, such as studying linguistic complexity and/or readability

Tokenization

- After basic cleaning, we want to pull out the meaningful subunits, which are the tokens

Tokenization

- After basic cleaning, we want to pull out the meaningful subunits, which are the tokens
- In other words, we want to turn human-readable text into machine-readable text

Tokenization

- After basic cleaning, we want to pull out the meaningful subunits, which are the tokens
- In other words, we want to turn human-readable text into machine-readable text
- While tokens are usually words, they might include numbers or punctuation too

Tokenization

- After basic cleaning, we want to pull out the meaningful subunits, which are the tokens
- In other words, we want to turn human-readable text into machine-readable text
- While tokens are usually words, they might include numbers or punctuation too
- A common rule for a tokenizer is to use whitespace as the marker, but some applicatins might require more subtlety

Tokenization

- After basic cleaning, we want to pull out the meaningful subunits, which are the tokens
- In other words, we want to turn human-readable text into machine-readable text
- While tokens are usually words, they might include numbers or punctuation too
- A common rule for a tokenizer is to use whitespace as the marker, but some applicatins might require more subtlety
- e.g., “Brown vs Board of Education” $\rightsquigarrow$ 'Brown', 'vs', 'Board', 'of', 'Education'

Tokenization

- After basic cleaning, we want to pull out the meaningful subunits, which are the tokens
- In other words, we want to turn human-readable text into machine-readable text
- While tokens are usually words, they might include numbers or punctuation too
- A common rule for a tokenizer is to use whitespace as the marker, but some applicatins might require more subtlety
- e.g., "Brown vs Board of Education" $\rightsquigarrow$ 'Brown', 'vs', 'Board', 'of', 'Education' ???

Tokenizers

- Thus, the naive whitespace approach implies many forms of "tokenizing" text: there are *many* tokenizers

Tokenizers

- Thus, the naive whitespace approach implies many forms of "tokenizing" text: there are *many* tokenizers
- **Word tokenizers**: whitespace (or with "word-stem," stemmed words distinguished by whitespace, *more in a bit*)

Tokenizers

- Thus, the naive whitespace approach implies many forms of "tokenizing" text: there are *many* tokenizers
- **Word tokenizers**: whitespace (or with "word-stem," stemmed words distinguished by whitespace, *more in a bit*)
- **Character and shingle tokenizers**: individual characters or small character-based subsets or "shingles" (e.g., `qu` or `th`)

Tokenizers

- Thus, the naive whitespace approach implies many forms of "tokenizing" text: there are *many* tokenizers
- **Word tokenizers**: whitespace (or with "word-stem," stemmed words distinguished by whitespace, *more in a bit*)
- **Character and shingle tokenizers**: individual characters or small character-based subsets or "shingles" (e.g., `qu` or `th`)
- **N-gram tokenizers**: some contiguous sequence of words (e.g., `he makes no sense`)

Tokenizers

- Thus, the naive whitespace approach implies many forms of "tokenizing" text: there are *many* tokenizers
- **Word tokenizers**: whitespace (or with "word-stem," stemmed words distinguished by whitespace, *more in a bit*)
- **Character and shingle tokenizers**: individual characters or small character-based subsets or "shingles" (e.g., `qu` or `th`)
- **N-gram tokenizers**: some contiguous sequence of words (e.g., `he makes no sense`)
- **Sentence and paragraph tokenizers**: single or multiple sentences, or even paragraphs

Tokenizers

- Thus, the naive whitespace approach implies many forms of "tokenizing" text: there are *many* tokenizers
- **Word tokenizers**: whitespace (or with "word-stem," stemmed words distinguished by whitespace, *more in a bit*)
- **Character and shingle tokenizers**: individual characters or small character-based subsets or "shingles" (e.g., `qu` or `th`)
- **N-gram tokenizers**: some contiguous sequence of words (e.g., `he makes no sense`)
- **Sentence and paragraph tokenizers**: single or multiple sentences, or even paragraphs
- **"Text chunking"**: similar to paragraph tokenizers, but can be even longer; stipulation is chunks of equal size (usually for large texts)

Tokenizers

- Thus, the naive whitespace approach implies many forms of "tokenizing" text: there are *many* tokenizers
- **Word tokenizers**: whitespace (or with "word-stem," stemmed words distinguished by whitespace, *more in a bit*)
- **Character and shingle tokenizers**: individual characters or small character-based subsets or "shingles" (e.g., `qu` or `th`)
- **N-gram tokenizers**: some contiguous sequence of words (e.g., `he makes no sense`)
- **Sentence and paragraph tokenizers**: single or multiple sentences, or even paragraphs
- **"Text chunking"**: similar to paragraph tokenizers, but can be even longer; stipulation is chunks of equal size (usually for large texts)
- And so on...

Tokenizers

- Thus, the naive whitespace approach implies many forms of "tokenizing" text: there are *many* tokenizers
- **Word tokenizers**: whitespace (or with "word-stem," stemmed words distinguished by whitespace, *more in a bit*)
- **Character and shingle tokenizers**: individual characters or small character-based subsets or "shingles" (e.g., `qu` or `th`)
- **N-gram tokenizers**: some contiguous sequence of words (e.g., `he makes no sense`)
- **Sentence and paragraph tokenizers**: single or multiple sentences, or even paragraphs
- **"Text chunking"**: similar to paragraph tokenizers, but can be even longer; stipulation is chunks of equal size (usually for large texts)
- And so on...
- Ultimately, the choice of tokenizer depends on the needs of the specific project

Stemming and Lemmatization

- Stemming is closely related to tokenizing

Stemming and Lemmatization

- Stemming is closely related to tokenizing
- Documents may use different forms of words ('lacked', 'lacking', 'lack'), or words which are similar in concept ('political', 'politician', 'politicizing')

Stemming and Lemmatization

- Stemming is closely related to tokenizing
- Documents may use different forms of words ('lacked', 'lacking', 'lack'), or words which are similar in concept ('political', 'politician', 'politicizing')
- **Stemming** maps these variants to the root of the word using a crude heuristic that chops off the affixes and returns a **stem**

Stemming and Lemmatization

- Stemming is closely related to tokenizing
- Documents may use different forms of words ('lacked', 'lacking', 'lack'), or words which are similar in concept ('political', 'politician', 'politicizing')
- **Stemming** maps these variants to the root of the word using a crude heuristic that chops off the affixes and returns a **stem**
- Lemmatization is similar, but stems based on vocabulary, parts of speech, and mapping rules, and returns a word in the dictionary

Stemming and Lemmatization

- Stemming is closely related to tokenizing
- Documents may use different forms of words ('lacked', 'lacking', 'lack'), or words which are similar in concept ('political', 'politician', 'politicizing')
- **Stemming** maps these variants to the root of the word using a crude heuristic that chops off the affixes and returns a **stem**
- Lemmatization is similar, but stems based on vocabulary, parts of speech, and mapping rules, and returns a word in the dictionary
- e.g. lemmatization would return 'see' or 'saw' if it came across 'saw' (clearly this is subjective, and requires constant quality checking)

Stemming and Lemmatization

Original Word		Stemmed Word
abolish	$\mapsto$	abolish
abolished	$\mapsto$	abolish
abolishing	$\mapsto$	abolish
abolition	$\mapsto$	abolit
abortion	$\mapsto$	abort
abortions	$\mapsto$	abort
abortive	$\mapsto$	abort
treasure	$\mapsto$	treasure
treasured	$\mapsto$	treasure
treasures	$\mapsto$	treasure
treasuring	$\mapsto$	treasure
treasury	$\mapsto$	treasuri

Stemming and Lemmatization

NYT

Emergency measures adopted for Beijing's first ''red alert" over air pollution left millions of schoolchildren cooped up at home, forced motorists off the roads and shut down factories across the region on Tuesday, but they failed to dispel the toxic air that shrouded the Chinese capital in a soupy, metallic haze.

marked up

Emergenc$\boxed{\text{y}}$ measur$\boxed{\text{es}}$ adopt$\boxed{\text{ed}}$ for Beij$\boxed{\text{ing}}$ s first red alert over air pollut$\boxed{\text{ion}}$ left million$\boxed{\text{s}}$ of schoolchildren coop$\boxed{\text{ed}}$ up at home, forc$\boxed{\text{ed}}$ motorist$\boxed{\text{s}}$ off the road$\boxed{\text{s}}$ and shut down factor$\boxed{\text{ies}}$ across the region on Tuesday, but they fail$\boxed{\text{ed}}$ to dispel the toxic air that shroud$\boxed{\text{ed}}$ the Chines$\boxed{\text{e}}$ capit$\boxed{\text{al}}$ in a soupy, metal$\boxed{\text{lic}}$ haze.

Stemming and Lemmatization

NYT

Emergency measures adopted for Beijing's first \red alert" over air pollution left millions of schoolchildren cooped up at home, forced motorists off the roads and shut down factories across the region on Tuesday, but they failed to dispel the toxic air that shrouded the Chinese capital in a soupy, metallic haze.

Stemmed

Emergenc measur adopt for Beij s first red alert over air pollut left million of schoolchildren coop up at home forc motorist off the road and shut down factori across the region on Tuesdai but thei fail to dispel the toxic air that shroud the Chines capit in a soupi metal haze.

Tagging

- To this point, there have been no distinctions drawn between nouns, verbs, and so on regarding our tokens, which makes sense for some applications (e.g., classification)

Tagging

- To this point, there have been no distinctions drawn between nouns, verbs, and so on regarding our tokens, which makes sense for some applications (e.g., classification)

- But, sometimes we may want to know information about the part-of-speech a word represents

Tagging

- To this point, there have been no distinctions drawn between nouns, verbs, and so on regarding our tokens, which makes sense for some applications (e.g., classification)
- But, sometimes we may want to know information about the part-of-speech a word represents
- E.g., we are often interested in recording who did what to whom, e.g., 'the UK bombing will force ISIS to surrender'

Tagging

- To this point, there have been no distinctions drawn between nouns, verbs, and so on regarding our tokens, which makes sense for some applications (e.g., classification)
- But, sometimes we may want to know information about the part-of-speech a word represents
- E.g., we are often interested in recording who did what to whom, e.g., 'the UK bombing will force ISIS to surrender'
- Here, **force** is a verb, not a noun

Tagging

- To this point, there have been no distinctions drawn between nouns, verbs, and so on regarding our tokens, which makes sense for some applications (e.g., classification)
- But, sometimes we may want to know information about the part-of-speech a word represents
- E.g., we are often interested in recording who did what to whom, e.g., 'the UK bombing will force ISIS to surrender'
- Here, **force** is a verb, not a noun
- Annotating in this way is called parts-of-speech tagging (e.g., the `RDRPOSTagger` library in R)

A Bag of Words

- We have now pre-processed our texts

A Bag of Words

- We have now pre-processed our texts!!!

A Bag of Words

- We have now preprocessed our texts $\rightsquigarrow$ *yay*

A Bag of Words

- We have now preprocessed our texts $\rightsquigarrow$ *yay*

- Generally, we are willing to ignore the order of the words in a document, which drastically simplifies things

A Bag of Words

- We have now preprocessed our texts $\rightsquigarrow$ *yay*
- Generally, we are willing to ignore the order of the words in a document, which drastically simplifies things
- And we do (almost) as well without ordering as when we retain it

A Bag of Words

- We have now preprocessed our texts $\rightsquigarrow$ *yay*
- Generally, we are willing to ignore the order of the words in a document, which drastically simplifies things
- And we do (almost) as well without ordering as when we retain it
- In such a world, we are treating a document as a "bag of words"

A Bag of Words

- We have now preprocessed our texts $\rightsquigarrow$ *yay*
- Generally, we are willing to ignore the order of the words in a document, which drastically simplifies things
- And we do (almost) as well without ordering as when we retain it
- In such a world, we are treating a document as a "bag of words"
- e.g. "The leading Republican presidential candidate has said Muslims should be banned from entering the US." $\rightsquigarrow$

A Bag of Words

- We have now preprocessed our texts $\rightsquigarrow$ *yay*
- Generally, we are willing to ignore the order of the words in a document, which drastically simplifies things
- And we do (almost) as well without ordering as when we retain it
- In such a world, we are treating a document as a "bag of words"
- e.g. "The leading Republican presidential candidate has said Muslims should be banned from entering the US." $\rightsquigarrow$
- "lead republican presidenti candid said muslim ban enter us" $\equiv$

A Bag of Words

- We have now preprocessed our texts $\rightsquigarrow$ *yay*
- Generally, we are willing to ignore the order of the words in a document, which drastically simplifies things
- And we do (almost) as well without ordering as when we retain it
- In such a world, we are treating a document as a "bag of words"
- e.g. "The leading Republican presidential candidate has said Muslims should be banned from entering the US." $\rightsquigarrow$
- "lead republican presidenti candid said muslim ban enter us" $\equiv$ "us lead said candid presidenti ban muslim republican enter"

Vector Space Models

- Another way to think about collections of text is a vector space model

Vector Space Models

- Another way to think about collections of text is a vector space model
- We have some document, which is considered a collection of W terms/features (words, tokens, etc.), where each term is a dimension

Vector Space Models

- Another way to think about collections of text is a vector space model
- We have some document, which is considered a collection of W terms/features (words, tokens, etc.), where each term is a dimension
- Documents (comprising these terms) are the linear combinations of vectors along the axes, implying document $W \rightsquigarrow \mathbb{R}^W$

Vector Space Models

- Another way to think about collections of text is a vector space model
- We have some document, which is considered a collection of W terms/features (words, tokens, etc.), where each term is a dimension
- Documents (comprising these terms) are the linear combinations of vectors along the axes, implying document $W \rightsquigarrow \mathbb{R}^W$
- The angle between vectors (cosine of the angle) captures the "_ness" of the document

Vector Space Models

- Another way to think about collections of text is a vector space model
- We have some document, which is considered a collection of W terms/features (words, tokens, etc.), where each term is a dimension
- Documents (comprising these terms) are the linear combinations of vectors along the axes, implying document $W \rightsquigarrow \mathbb{R}^W$
- The angle between vectors (cosine of the angle) captures the "_ness" of the document
- e.g. the document "Philip is short" can be thought of a vector in 3 dimensions:

Vector Space Models

- Another way to think about collections of text is a vector space model
- We have some document, which is considered a collection of W terms/features (words, tokens, etc.), where each term is a dimension
- Documents (comprising these terms) are the linear combinations of vectors along the axes, implying document $W \rightsquigarrow \mathbb{R}^W$
- The angle between vectors (cosine of the angle) captures the "_ness" of the document
- e.g. the document "Philip is short" can be thought of a vector in 3 dimensions: d_1 corresponds to how 'Philip'-ish it is, d_2 corresponds to how 'is'-ish it is, and d_3 corresponds to how 'short'-ish it is

Vector Space Models

- Another way to think about collections of text is a vector space model
- We have some document, which is considered a collection of W terms/features (words, tokens, etc.), where each term is a dimension
- Documents (comprising these terms) are the linear combinations of vectors along the axes, implying document $W \rightsquigarrow \mathbb{R}^W$
- The angle between vectors (cosine of the angle) captures the "_ness" of the document
- e.g. the document "Philip is short" can be thought of a vector in 3 dimensions: d_1 corresponds to how 'Philip'-ish it is, d_2 corresponds to how 'is'-ish it is, and d_3 corresponds to how 'short'-ish it is
- Features are typically unigram frequencies of the tokens in the document

Vector Space Models

- Another way to think about collections of text is a vector space model
- We have some document, which is considered a collection of W terms/features (words, tokens, etc.), where each term is a dimension
- Documents (comprising these terms) are the linear combinations of vectors along the axes, implying document $W \rightsquigarrow \mathbb{R}^W$
- The angle between vectors (cosine of the angle) captures the "_ness" of the document
- e.g. the document "Philip is short" can be thought of a vector in 3 dimensions: d_1 corresponds to how 'Philip'-ish it is, d_2 corresponds to how 'is'-ish it is, and d_3 corresponds to how 'short'-ish it is
- Features are typically unigram frequencies of the tokens in the document
- e.g. "the dog chased the cat" becomes $(2, 1, 1, 1)$ for dimensions defined as simple counts across `the`, `dog`, `chased`, `cat`

Vector Space Models

- Another way to think about collections of text is a vector space model
- We have some document, which is considered a collection of W terms/features (words, tokens, etc.), where each term is a dimension
- Documents (comprising these terms) are the linear combinations of vectors along the axes, implying document $W \rightsquigarrow \mathbb{R}^W$
- The angle between vectors (cosine of the angle) captures the "_ness" of the document
- e.g. the document "Philip is short" can be thought of a vector in 3 dimensions: d_1 corresponds to how 'Philip'-ish it is, d_2 corresponds to how 'is'-ish it is, and d_3 corresponds to how 'short'-ish it is
- Features are typically unigram frequencies of the tokens in the document
- e.g. "the dog chased the cat" becomes $(2, 1, 1, 1)$ for dimensions defined as simple counts across `the`, `dog`, `chased`, `cat`
- Thus, the vector space model represents a document vector in d-dimensional feature space

Vector Space Models: Notation

- $d = 1, \ldots, D \rightsquigarrow$ indexes documents in the corpus

Vector Space Models: Notation

- $d = 1, \ldots, D \rightsquigarrow$ indexes documents in the corpus
- $w = 1, \ldots, W \rightsquigarrow$ indexes features in the documents

Vector Space Models: Notation

- $d = 1, \ldots, D \rightsquigarrow$ indexes documents in the corpus
- $w = 1, \ldots, W \rightsquigarrow$ indexes features in the documents
- $\mathbf{y}_d \in \mathbb{R}^W \rightsquigarrow$ representation of document d in a some feature space

Vector Space Models: Notation

- $d = 1, \ldots, D \rightsquigarrow$ indexes documents in the corpus
- $w = 1, \ldots, W \rightsquigarrow$ indexes features in the documents
- $\mathbf{y}_d \in \mathbb{R}^W \rightsquigarrow$ representation of document d in a some feature space
- So each document is now a vector, with each entry representing the frequency of a particular token or feature

Vector Space Models: Notation

- $d = 1, \ldots, D \rightsquigarrow$ indexes documents in the corpus
- $w = 1, \ldots, W \rightsquigarrow$ indexes features in the documents
- $\mathbf{y}_d \in \mathbb{R}^W \rightsquigarrow$ representation of document d in a some feature space
- So each document is now a vector, with each entry representing the frequency of a particular token or feature
- Stacking these vectors on top of each other gives the document term matrix (DTM) (sometimes called the document feature matrix (DFM))

Weighting Word Importance

- The most common approach to weighting word importance *across* documents is the tf-idf weighting scheme, based on Zipf's law (frequency a word appears is inversely proportional to its rank)

Weighting Word Importance

- The most common approach to weighting word importance *across* documents is the tf-idf weighting scheme, based on Zipf's law (frequency a word appears is inversely proportional to its rank)

- The frequency of a term is adjusted for how often and/or rarely it is used

Weighting Word Importance

- The most common approach to weighting word importance *across* documents is the tf-idf weighting scheme, based on Zipf's law (frequency a word appears is inversely proportional to its rank)

- The frequency of a term is adjusted for how often and/or rarely it is used
    1. Calculate a term's frequency (tf)

Weighting Word Importance

- The most common approach to weighting word importance *across* documents is the tf-idf weighting scheme, based on Zipf's law (frequency a word appears is inversely proportional to its rank)

- The frequency of a term is adjusted for how often and/or rarely it is used
  1. Calculate a term's frequency (tf)
  2. Calculate a term's inverse document frequency (idf, $ln(\frac{N_{docs}}{N_{docs.containing.term}})$)

Weighting Word Importance

- The most common approach to weighting word importance *across* documents is the tf-idf weighting scheme, based on Zipf's law (frequency a word appears is inversely proportional to its rank)

- The frequency of a term is adjusted for how often and/or rarely it is used
  1. Calculate a term's frequency (tf)
  2. Calculate a term's inverse document frequency (idf, $ln(\frac{N_{docs}}{N_{docs.containing.term}})$)
  3. tf-idf $\rightsquigarrow$ product of tf and idf

Weighting Word Importance

- The most common approach to weighting word importance *across* documents is the tf-idf weighting scheme, based on Zipf's law (frequency a word appears is inversely proportional to its rank)

- The frequency of a term is adjusted for how often and/or rarely it is used
  1. Calculate a term's frequency (tf)
  2. Calculate a term's inverse document frequency (idf, $ln(\frac{N_{docs}}{N_{docs.containing.term}})$)
  3. tf-idf $\rightsquigarrow$ product of tf and idf

- Functionally, the tf-idf captures a decrease in the weight for commonly used words and, for more rarely used words, an increase in the weight for words use more rarely in some set of documents, D, that are not used very much in a collection of documents

Lexical Diversity

- We may also be interested in mining the diversity of some document or set of documents $\rightsquigarrow$ lexical diversity

Lexical Diversity

- We may also be interested in mining the diversity of some document or set of documents $\rightsquigarrow$ lexical diversity
- Recall the elementary components of some text are called **tokens** (usually words, but could be numbers, etc.)

Lexical Diversity

- We may also be interested in mining the diversity of some document or set of documents $\rightsquigarrow$ lexical diversity

- Recall the elementary components of some text are called **tokens** (usually words, but could be numbers, etc.)

- The **types** in a document are the set of *unique* tokens

Lexical Diversity

- We may also be interested in mining the diversity of some document or set of documents $\rightsquigarrow$ lexical diversity
- Recall the elementary components of some text are called **tokens** (usually words, but could be numbers, etc.)
- The **types** in a document are the set of *unique* tokens
- Thus we typically have many more **tokens** than **types**, because authors repeat tokens

Lexical Diversity

- We may also be interested in mining the diversity of some document or set of documents $\rightsquigarrow$ lexical diversity
- Recall the elementary components of some text are called **tokens** (usually words, but could be numbers, etc.)
- The **types** in a document are the set of *unique* tokens
- Thus we typically have many more **tokens** than **types**, because authors repeat tokens
- To measure lexical diversity, we can use the type-to-token ratio TTR

$$TTR = \frac{N_{types}}{N_{tokens}}$$

Lexical Diversity

- We may also be interested in mining the diversity of some document or set of documents $\rightsquigarrow$ lexical diversity
- Recall the elementary components of some text are called **tokens** (usually words, but could be numbers, etc.)
- The **types** in a document are the set of *unique* tokens
- Thus we typically have many more **tokens** than **types**, because authors repeat tokens
- To measure lexical diversity, we can use the type-to-token ratio TTR

$$TTR = \frac{N_{types}}{N_{tokens}}$$

- This may provide increased clarity of the text, e.g., authors with limited vocabularies might have a low lexical diversity

Lecture Outline

Supervised Learning for Classification of Text

- We've covered the basics of text mining, simplification, document representation, and so on

Supervised Learning for Classification of Text

- We've covered the basics of text mining, simplification, document representation, and so on
- We can build on this to begin to think about documents as members of categories or classes

Supervised Learning for Classification of Text

- We've covered the basics of text mining, simplification, document representation, and so on
- We can build on this to begin to think about documents as members of categories or classes
- Integral to classification is the dictionary

Supervised Learning for Classification of Text

- We've covered the basics of text mining, simplification, document representation, and so on
- We can build on this to begin to think about documents as members of categories or classes
- Integral to classification is the dictionary
- This allows us to move on to supervised learning problems

Supervised Learning for Classification of Text

- Recall in supervised learning that labels are key

Supervised Learning for Classification of Text

- Recall in supervised learning that labels are key
- We start by labelling some examples of each category, e.g. some reviews that were positive ($y = 1$), some that were negative ($y = 0$);

Supervised Learning for Classification of Text

- Recall in supervised learning that labels are key
- We start by labelling some examples of each category, e.g. some reviews that were positive ($y = 1$), some that were negative ($y = 0$); whether some statements that were liberal, some that were conservative; etc.

Supervised Learning for Classification of Text

- Recall in supervised learning that labels are key

- We start by labelling some examples of each category, e.g. some reviews that were positive ($y = 1$), some that were negative ($y = 0$); whether some statements that were liberal, some that were conservative; etc.

- Next, we train a machine (model) on these examples, using the features (DTM) as the independent variables,

Supervised Learning for Classification of Text

- Recall in supervised learning that labels are key
- We start by labelling some examples of each category, e.g. some reviews that were positive ($y = 1$), some that were negative ($y = 0$); whether some statements that were liberal, some that were conservative; etc.
- Next, we train a machine (model) on these examples, using the features (DTM) as the independent variables, e.g. does the commentator use the word "fetus" or "baby" in discussing abortion law?

Supervised Learning for Classification of Text

- Recall in supervised learning that labels are key
- We start by labelling some examples of each category, e.g. some reviews that were positive ($y = 1$), some that were negative ($y = 0$); whether some statements that were liberal, some that were conservative; etc.
- Next, we train a machine (model) on these examples, using the features (DTM) as the independent variables, e.g. does the commentator use the word "fetus" or "baby" in discussing abortion law? Gives a clue as to the "ideology" of the speaker/author

Supervised Learning for Classification of Text

- Recall in supervised learning that labels are key
- We start by labelling some examples of each category, e.g. some reviews that were positive ($y = 1$), some that were negative ($y = 0$); whether some statements that were liberal, some that were conservative; etc.
- Next, we train a machine (model) on these examples, using the features (DTM) as the independent variables, e.g. does the commentator use the word "fetus" or "baby" in discussing abortion law? Gives a clue as to the "ideology" of the speaker/author
- The **goal**, then, is to classify some text by using the **learned relationship between labels and features** to predict the classes of *future* documents (e.g., $y \in \{0, 1\}$, sentiment) not in the training set

Lecture Outline

A Dictionary

- The most fundamental concept in supervised text classification:

A Dictionary

- The most fundamental concept in supervised text classification: dictionary

A Dictionary

- The most fundamental concept in supervised text classification: dictionary
- Set of pre-defined words with specific connotations that allow us to classify documents automatically, quickly and (usually) accurately

A Dictionary

- The most fundamental concept in supervised text classification: dictionary
- Set of pre-defined words with specific connotations that allow us to classify documents automatically, quickly and (usually) accurately
- Very common in opinion mining/sentiment analysis, and also in coding events or party platforms

A Dictionary

- The most fundamental concept in supervised text classification: dictionary
- Set of pre-defined words with specific connotations that allow us to classify documents automatically, quickly and (usually) accurately
- Very common in opinion mining/sentiment analysis, and also in coding events or party platforms
- Often used in supervised learning problems, as a starting point

A Dictionary

- The most fundamental concept in supervised text classification: dictionary
- Set of pre-defined words with specific connotations that allow us to classify documents automatically, quickly and (usually) accurately
- Very common in opinion mining/sentiment analysis, and also in coding events or party platforms
- Often used in supervised learning problems, as a starting point
- As such, we will cover them in this context $\rightsquigarrow$ sentiment analysis

Classification with Dictionaries

- Goal: usually, we are trying to do one of two closely related things:

Classification with Dictionaries

- Goal: usually, we are trying to do one of two closely related things:

  1. Categorize documents in unique classes
     - This review is 'positive'; this speech is 'liberal'

Classification with Dictionaries

- Goal: usually, we are trying to do one of two closely related things:

  1. Categorize documents in unique classes
     - This review is 'positive'; this speech is 'liberal'
  2. Measure extent to which document is *associated* with a category
     - This review is generally 'positive', but has some negative elements

Classification with Dictionaries

- Goal: usually, we are trying to do one of two closely related things:
  1. Categorize documents in unique classes
     - ★ This review is 'positive'; this speech is 'liberal'
  2. Measure extent to which document is *associated* with a category
     - ★ This review is generally 'positive', but has some negative elements
- A dictionary guides us, the (weighted) presence of which helps us with either of these goals

Classification with Dictionaries

- Goal: usually, we are trying to do one of two closely related things:

  1. Categorize documents in unique classes
     - This review is 'positive'; this speech is 'liberal'
  2. Measure extent to which document is *associated* with a category
     - This review is generally 'positive', but has some negative elements

- A dictionary guides us, the (weighted) presence of which helps us with either of these goals
- Some weights are binary (either $+1$ or -1), while others are continuous (e.g., ranging from -5 to $+5$)

Classification with Dictionaries

- Goal: usually, we are trying to do one of two closely related things:
  1. Categorize documents in unique classes
     - ★ This review is 'positive'; this speech is 'liberal'
  2. Measure extent to which document is *associated* with a category
     - ★ This review is generally 'positive', but has some negative elements
- A dictionary guides us, the (weighted) presence of which helps us with either of these goals
- Some weights are binary (either $+1$ or -1), while others are continuous (e.g., ranging from -5 to $+5$)
- Some dictionaries capture broad sentiment (e.g., positive/negative),

Classification with Dictionaries

- Goal: usually, we are trying to do one of two closely related things:

  1. Categorize documents in unique classes
     - ⋆ This review is 'positive'; this speech is 'liberal'
  2. Measure extent to which document is *associated* with a category
     - ⋆ This review is generally 'positive', but has some negative elements

- A dictionary guides us, the (weighted) presence of which helps us with either of these goals
- Some weights are binary (either $+1$ or -1), while others are continuous (e.g., ranging from -5 to $+5$)
- Some dictionaries capture broad sentiment (e.g., positive/negative), others capture emotions (e.g., anger, joy, disgust),

Classification with Dictionaries

- Goal: usually, we are trying to do one of two closely related things:

  1. Categorize documents in unique classes
     - ★ This review is 'positive'; this speech is 'liberal'
  2. Measure extent to which document is *associated* with a category
     - ★ This review is generally 'positive', but has some negative elements

- A dictionary guides us, the (weighted) presence of which helps us with either of these goals
- Some weights are binary (either $+1$ or -1), while others are continuous (e.g., ranging from -5 to $+5$)
- Some dictionaries capture broad sentiment (e.g., positive/negative), others capture emotions (e.g., anger, joy, disgust), still others are non-sentiment (e.g., politics, food, places)

Classification with Dictionaries

- In solving the classification problem, have a set of key words with scores, e.g., “terrible” $= -1$; “fantastic” $= +1$

Classification with Dictionaries

- In solving the classification problem, have a set of key words with scores, e.g., "terrible" $= -1$; "fantastic" $= +1$

- The relative rate of occurrence of these terms tells us about the *overall tone*/class the document should be placed in

Classification with Dictionaries

- In solving the classification problem, have a set of key words with scores, e.g., "terrible" $= -1$; "fantastic" $= +1$

- The relative rate of occurrence of these terms tells us about the *overall tone*/class the document should be placed in

- The tone, Y, based on document i and words $m = 1, \ldots, M$ in the **dictionary**,

$$Y_i = \sum_{m=1}^{M} \frac{s_m w_{im}}{N_i}$$

  where
  - s_m is the score of the word m
  - w_{im} is the number of occurences of the mth dictionary word in document i
  - N_i is the total number of all **dictionary** words in the document

Classification with Dictionaries

- To classify, simply add up the number of times the **dictionary** words appear and multiply by the score (normalizing by document dictionary presence)

Classification with Dictionaries

- To classify, simply add up the number of times the **dictionary** words appear and multiply by the score (normalizing by document dictionary presence)
- Dependent on the dictionary, the score, Y ("tone") impacts document classification,

Classification with Dictionaries

- To classify, simply add up the number of times the **dictionary** words appear and multiply by the score (normalizing by document dictionary presence)
- Dependent on the dictionary, the score, Y ("tone") impacts document classification,
  - $Y > 0 \rightsquigarrow$ positive

Classification with Dictionaries

- To classify, simply add up the number of times the **dictionary** words appear and multiply by the score (normalizing by document dictionary presence)
- Dependent on the dictionary, the score, Y ("tone") impacts document classification,
  - $Y > 0 \rightsquigarrow$ positive
  - $Y < 0 \rightsquigarrow$ negative

Classification with Dictionaries

- To classify, simply add up the number of times the **dictionary** words appear and multiply by the score (normalizing by document dictionary presence)

- Dependent on the dictionary, the score, Y ("tone") impacts document classification,
  - $Y > 0 \rightsquigarrow$ positive
  - $Y < 0 \rightsquigarrow$ negative
  - $Y \approx 0 \rightsquigarrow$ ambiguous

For example... The Big Short (newsreview.com)

Director and co-screenwriter Adam McKay (Step Brothers) bungles a great opportunity to savage the architects of the 2008 financial crisis in The Big Short, wasting an A-list ensemble cast in the process. Steve Carell, Brad Pitt, Christian Bale and Ryan Gosling play various tenuously related members of the finance industry, men who made made a killing by betting against the housing market, which at that point had superficially swelled to record highs. All of the elements are in place for a lacerating satire, but almost every aesthetic choice in the film is bad, from the U-Turn-era Oliver Stone visuals to Carell's sketch-comedy performance to the cheeky cutaways where Selena Gomez and Anthony Bourdain explain complex financial concepts. After a brutal opening half, it finally settles into a groove, and there's a queasy charge in watching a credit-drunk America walking towards that cliff's edge, but not enough to save the film.

For example... The Big Short (newsreview.com)

great *savage*
crisis *wasting*

tenuously
killing
superficially swelled

bad

complex *brutal*

drunk
enough

Simple Math Using Bing

- Using the Bing dictionary, we can see that there are 11 negative words, and 2 positive words

Simple Math Using Bing

- Using the Bing dictionary, we can see that there are 11 negative words, and 2 positive words

- Thus, the tone could be calculated as,

$$= \frac{2 - 11}{13}$$

Simple Math Using Bing

- Using the Bing dictionary, we can see that there are 11 negative words, and 2 positive words

- Thus, the tone could be calculated as,

$$= \frac{2 - 11}{13}$$

$$= \frac{-9}{13}$$

Simple Math Using Bing

- Using the Bing dictionary, we can see that there are 11 negative words, and 2 positive words

- Thus, the tone could be calculated as,

$$= \frac{2 - 11}{13}$$

$$= \frac{-9}{13}$$

$$-0.6923$$

Simple Math Using Bing

- Using the Bing dictionary, we can see that there are 11 negative words, and 2 positive words

- Thus, the tone could be calculated as,

$$= \frac{2 - 11}{13}$$

$$= \frac{-9}{13}$$

$$-0.6923$$

bad

Exercise Caution...

- Proper use of dictionaries is dependent on contextual/project-based goals

Exercise Caution...

- Proper use of dictionaries is dependent on contextual/project-based goals
- Most dictionaries do not take into account qualifiers (e.g. "no good")

Exercise Caution...

- Proper use of dictionaries is dependent on contextual/project-based goals
- Most dictionaries do not take into account qualifiers (e.g. "no good")
- All ignore sarcasm, irony, nuance

Exercise Caution...

- Proper use of dictionaries is dependent on contextual/project-based goals
- Most dictionaries do not take into account qualifiers (e.g. "no good")
- All ignore sarcasm, irony, nuance
- Ultimately context matters, and any dictionary used should be clearly justified

Exercise Caution...

- Common sentiment dictionaries:

Exercise Caution...

- Common sentiment dictionaries:
  - **ANEW** (Affective Norms for English Words): "Happiness" dictionary with 1034 words, measuring affective reactions to words

Exercise Caution...

- Common sentiment dictionaries:
  - **ANEW** (Affective Norms for English Words): "Happiness" dictionary with 1034 words, measuring affective reactions to words
  - **Bing**: Binary classification for positive/negative sentiments

Exercise Caution...

- Common sentiment dictionaries:
  - **ANEW** (Affective Norms for English Words): "Happiness" dictionary with 1034 words, measuring affective reactions to words
  - **Bing**: Binary classification for positive/negative sentiments
  - **NRC**: Binary classification for a variety of categories like positive, negative, disgust, fear, joy, trust, surprise

Exercise Caution...

- Common sentiment dictionaries:
    - **ANEW** (Affective Norms for English Words): "Happiness" dictionary with 1034 words, measuring affective reactions to words
    - **Bing**: Binary classification for positive/negative sentiments
    - **NRC**: Binary classification for a variety of categories like positive, negative, disgust, fear, joy, trust, surprise
    - **LIWC**: 2300 words grouped into 70 classes on positive/negative emotions

Exercise Caution...

- Common sentiment dictionaries:
  - **ANEW** (Affective Norms for English Words): "Happiness" dictionary with 1034 words, measuring affective reactions to words
  - **Bing**: Binary classification for positive/negative sentiments
  - **NRC**: Binary classification for a variety of categories like positive, negative, disgust, fear, joy, trust, surprise
  - **LIWC**: 2300 words grouped into 70 classes on positive/negative emotions
  - **AFINN**: Continuous scores between -5 and 5 for negative and positive sentiment, respectively

Exercise Caution...

- Common sentiment dictionaries:
  - **ANEW** (Affective Norms for English Words): "Happiness" dictionary with 1034 words, measuring affective reactions to words
  - **Bing**: Binary classification for positive/negative sentiments
  - **NRC**: Binary classification for a variety of categories like positive, negative, disgust, fear, joy, trust, surprise
  - **LIWC**: 2300 words grouped into 70 classes on positive/negative emotions
  - **AFINN**: Continuous scores between -5 and 5 for negative and positive sentiment, respectively
  - **General Inquirer Database**: 3627 negative and positive word strings (widely used across domains)

Lecture Outline

Creating Dictionaries

- These dictionaries are great and widely used

Creating Dictionaries

- These dictionaries are great and widely used
- But what if we are interested in weighting word frequencies with non-sentiment-based words, like, political language, gender language, racist language, and so on...

Creating Dictionaries

- These dictionaries are great and widely used
- But what if we are interested in weighting word frequencies with non-sentiment-based words, like, political language, gender language, racist language, and so on...
- In this case, we might consider creating our own dictionaries

Creating Dictionaries

- Though there are a several ways, here are the three most widely used:

Creating Dictionaries

- Though there are a several ways, here are the three most widely used:

    1. Separating methods

Creating Dictionaries

- Though there are a several ways, here are the three most widely used:

  1. Separating methods
  2. Manual generation $\rightsquigarrow$ careful thought about useful words

Creating Dictionaries

- Though there are a several ways, here are the three most widely used:
    1. Separating methods
    2. Manual generation $\rightsquigarrow$ careful thought about useful words
    3. Populations of people who are (*surprisingly*) willing to perform ill-defined tasks (i.e., crowd-sourcing)

Creating Dictionaries

- Though there are a several ways, here are the three most widely used:

  1. Separating methods
  2. Manual generation $\rightsquigarrow$ careful thought about useful words
  3. Populations of people who are (*surprisingly*) willing to perform ill-defined tasks (i.e., crowd-sourcing)
     - Undergraduates

Creating Dictionaries

- Though there are a several ways, here are the three most widely used:

1. Separating methods
2. Manual generation $\rightsquigarrow$ careful thought about useful words
3. Populations of people who are (*surprisingly*) willing to perform ill-defined tasks (i.e., crowd-sourcing)
    - Undergraduates: Pizza $\rightsquigarrow$ Research Output

Creating Dictionaries

- Though there are a several ways, here are the three most widely used:

1. Separating methods
2. Manual generation $\rightsquigarrow$ careful thought about useful words
3. Populations of people who are (*surprisingly*) willing to perform ill-defined tasks (i.e., crowd-sourcing)
    - ★ Undergraduates: Pizza $\rightsquigarrow$ Research Output
    - ★ Mechanical turkers

Creating Dictionaries

- Though there are a several ways, here are the three most widely used:

1. Separating methods
2. Manual generation $\rightsquigarrow$ careful thought about useful words
3. Populations of people who are (*surprisingly*) willing to perform ill-defined tasks (i.e., crowd-sourcing)
   - Undergraduates: Pizza $\rightsquigarrow$ Research Output
   - Mechanical turkers, e.g., ask turkers: how happy is, `elevator`, `car`, `pretty`, `young`

Creating Dictionaries

- Though there are a several ways, here are the three most widely used:

  1. Separating methods
  2. Manual generation $\rightsquigarrow$ careful thought about useful words
  3. Populations of people who are (*surprisingly*) willing to perform ill-defined tasks (i.e., crowd-sourcing)

     - ★ Undergraduates: Pizza $\rightsquigarrow$ Research Output
     - ★ Mechanical turkers, e.g., ask turkers: how happy is, `elevator`, `car`, `pretty`, `young` $\rightsquigarrow$ Output as dictionary

Separating Methods

- What is the goal here?

Separating Methods

- What is the goal here? **Initialize the creation of a dictionary by finding words that discriminate between groups of texts**

Separating Methods

- What is the goal here? **Initialize the creation of a dictionary by finding words that discriminate between groups of texts**

- There are three main ways to separate, and thus manually locate distinctive words in text

Separating Methods

- What is the goal here? **Initialize the creation of a dictionary by finding words that discriminate between groups of texts**
- There are three main ways to separate, and thus manually locate distinctive words in text
  1. Exclusive/unique use: words used in a single document

Separating Methods

- What is the goal here? **Initialize the creation of a dictionary by finding words that discriminate between groups of texts**

- There are three main ways to separate, and thus manually locate distinctive words in text

  1. Exclusive/unique use: words used in a single document

  2. Difference in frequencies: *absolute* differences frequency of use

Separating Methods

- What is the goal here? **Initialize the creation of a dictionary by finding words that discriminate between groups of texts**

- There are three main ways to separate, and thus manually locate distinctive words in text

  1. Exclusive/unique use: words used in a single document
  2. Difference in frequencies: *absolute* differences frequency of use
  3. Difference in rates/average usage: difference between proportions of same word use across documents (where high difference = good/distinct)

Separating Methods in R

Quick demo of each in R

Validation?

- Dictionary methods are **context invariant**

Validation?

- Dictionary methods are **context invariant**
  - Optimization $\rightsquigarrow$ allows you to incorporate information specific to context

Validation?

- Dictionary methods are **context invariant**
  - Optimization $\rightsquigarrow$ allows you to incorporate information specific to context
  - No optimization $\rightsquigarrow$ same word weights regardless of texts or contexts

Validation?

- Dictionary methods are **context invariant**
  - Optimization $\rightsquigarrow$ allows you to incorporate information specific to context
  - No optimization $\rightsquigarrow$ same word weights regardless of texts or contexts
  - Without optimization, its unclear about dictionaries' performances (e.g., did they correctly classify?)

Validation?

- Dictionary methods are **context invariant**
  - Optimization $\rightsquigarrow$ allows you to incorporate information specific to context
  - No optimization $\rightsquigarrow$ same word weights regardless of texts or contexts
  - Without optimization, its unclear about dictionaries' performances (e.g., did they correctly classify?)
- **Importantly**, just because we get a positive or negative score, does **NOT** mean that these are accurate measures in our text

Validation?

- Dictionary methods are **context invariant**
  - Optimization $\rightsquigarrow$ allows you to incorporate information specific to context
  - No optimization $\rightsquigarrow$ same word weights regardless of texts or contexts
  - Without optimization, its unclear about dictionaries' performances (e.g., did they correctly classify?)
- **Importantly**, just because we get a positive or negative score, does **NOT** mean that these are accurate measures in our text
- This points to the need for validation

Validation!

- Classification validity (*requires hand coded documents*):
  - *Training*: build dictionary on subset of documents with known labels
  - *Testing*: apply dictionary method to other documents with known labels
    - Is the classification scheme well defined for your texts?
    - Can humans accomplish the coding task with consistency (e.g., Cronbach's α)?
    - Is the dictionary appropriate?

Validation!

- Classification validity (*requires hand coded documents*):
  - *Training*: build dictionary on subset of documents with known labels
  - *Testing*: apply dictionary method to other documents with known labels
    - ★ Is the classification scheme well defined for your texts?
    - ★ Can humans accomplish the coding task with consistency (e.g., Cronbach's α)?
    - ★ Is the dictionary appropriate?
- Replicate classification exercise
  - How well does our method perform on *held out* documents?
  - Why "held out"? Over-fitting
  - (Cross)validation
  - Can also use off-the-shelf dictionaries to compare

The Confusion Matrix

- We gauge accuracy using our friend the confusion matrix, discussed earlier in the quarter;

The Confusion Matrix

- We gauge accuracy using our friend the confusion matrix, discussed earlier in the quarter; as a refresher:

The Confusion Matrix

- We gauge accuracy using our friend the confusion matrix, discussed earlier in the quarter; as a refresher:
  - Sensitivity (recall or "hit rate"): $\frac{TP}{TP+FN}$

The Confusion Matrix

- We gauge accuracy using our friend the confusion matrix, discussed earlier in the quarter; as a refresher:
  - Sensitivity (recall or "hit rate"): $\frac{TP}{TP+FN}$
  - Specificity: $\frac{TN}{TN+FP}$

The Confusion Matrix

- We gauge accuracy using our friend the confusion matrix, discussed earlier in the quarter; as a refresher:
  - Sensitivity (recall or "hit rate"): $\frac{TP}{TP+FN}$
  - Specificity: $\frac{TN}{TN+FP}$
  - Precision ("positive predicted value"): $\frac{TP}{TP+FP}$

The Confusion Matrix

- We gauge accuracy using our friend the confusion matrix, discussed earlier in the quarter; as a refresher:
  - Sensitivity (recall or "hit rate"): $\frac{TP}{TP+FN}$
  - Specificity: $\frac{TN}{TN+FP}$
  - Precision ("positive predicted value"): $\frac{TP}{TP+FP}$
  - Accuracy: $\frac{TP+TN}{TP+TN+FP+FN}$

The Confusion Matrix

- We gauge accuracy using our friend the confusion matrix, discussed earlier in the quarter; as a refresher:
  - Sensitivity (recall or "hit rate"): $\frac{TP}{TP+FN}$
  - Specificity: $\frac{TN}{TN+FP}$
  - Precision ("positive predicted value"): $\frac{TP}{TP+FP}$
  - Accuracy: $\frac{TP+TN}{TP+TN+FP+FN}$
  - F1 score ("F-measure"): $\frac{2TP}{2TP+FP+FN}$

Lecture Outline

Components of Supervised Learning for Classifying Texts

1. Set of categories, e.g., sentiments
   - Positive Tone, Negative Tone

Components of Supervised Learning for Classifying Texts

1. Set of categories, e.g., sentiments
   - Positive Tone, Negative Tone
2. Set of (human) hand-coded documents labeling according to the categories in step 1
   - Training Set: documents we'll use to learn how to code
   - Test/Validation Set: documents we'll use to learn how well we code

Components of Supervised Learning for Classifying Texts

1. Set of categories, e.g., sentiments
   - Positive Tone, Negative Tone
2. Set of (human) hand-coded documents labeling according to the categories in step 1
   - Training Set: documents we'll use to learn how to code
   - Test/Validation Set: documents we'll use to learn how well we code
3. Obtain a new set of unlabeled documents

Components of Supervised Learning for Classifying Texts

1. Set of categories, e.g., sentiments
   - Positive Tone, Negative Tone
2. Set of (human) hand-coded documents labeling according to the categories in step 1
   - Training Set: documents we'll use to learn how to code
   - Test/Validation Set: documents we'll use to learn how well we code
3. Obtain a new set of unlabeled documents
4. Rinse and repeat

Components of Supervised Learning for Classifying Texts

1. Set of categories, e.g., sentiments
   - Positive Tone, Negative Tone
2. Set of (human) hand-coded documents labeling according to the categories in step 1
   - Training Set: documents we'll use to learn how to code
   - Test/Validation Set: documents we'll use to learn how well we code
3. Obtain a new set of unlabeled documents
4. Rinse and repeat

- But *how* do we classify?

Components of Supervised Learning for Classifying Texts

1. Set of categories, e.g., sentiments
   - Positive Tone, Negative Tone
2. Set of (human) hand-coded documents labeling according to the categories in step 1
   - Training Set: documents we'll use to learn how to code
   - Test/Validation Set: documents we'll use to learn how well we code
3. Obtain a new set of unlabeled documents
4. Rinse and repeat

- But *how* do we classify? many ways, but we will cover regression *as an example*

Regression models

Suppose we have N documents, with each document i having label $y_i \in \{-1, 1\} \rightsquigarrow \{\text{negative, positive}\}$

Regression models

Suppose we have N documents, with each document i having label $y_i \in \{-1, 1\} \rightsquigarrow \{\text{negative, positive}\}$
We represent each document i is $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{iJ})$.

Regression models

Suppose we have N documents, with each document i having label $y_i \in \{-1, 1\} \rightsquigarrow \{\text{negative, positive}\}$
We represent each document i is $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{iJ})$.

$$f(\beta, \boldsymbol{X}, \boldsymbol{Y}) = \sum_{i=1}^{N} \left(y_i - \boldsymbol{\beta}^{'} \boldsymbol{x}_i\right)^2$$

Regression models

Suppose we have N documents, with each document i having label
$y_i \in \{-1, 1\} \rightsquigarrow \{\text{negative, positive}\}$
We represent each document i is $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{iJ})$.

$$
\begin{aligned}
f(\boldsymbol{\beta}, \boldsymbol{X}, \boldsymbol{Y}) &= \sum_{i=1}^{N} \left(y_i - \boldsymbol{\beta}^{'} \boldsymbol{x}_i\right)^2 \\
\widehat{\boldsymbol{\beta}} &= \arg\min_{\boldsymbol{\beta}} \left\{ \sum_{i=1}^{N} \left(y_i - \boldsymbol{\beta}^{'} \boldsymbol{x}_i\right)^2 \right\} \qquad (1)
\end{aligned}
$$

Problem:

Regression models

Suppose we have N documents, with each document i having label $y_i \in \{-1, 1\} \rightsquigarrow \{\text{negative, positive}\}$
We represent each document i is $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{iJ})$.

$$
\begin{aligned}
f(\boldsymbol{\beta}, \boldsymbol{X}, \boldsymbol{Y}) &= \sum_{i=1}^{N} \left(y_i - \boldsymbol{\beta}' \boldsymbol{x}_i\right)^2 \\
\widehat{\boldsymbol{\beta}} &= \text{arg min}_{\boldsymbol{\beta}} \left\{ \sum_{i=1}^{N} \left(y_i - \boldsymbol{\beta}' \boldsymbol{x}_i\right)^2 \right\} \qquad (1)
\end{aligned}
$$

Problem:

- J will likely be large (perhaps even $J > N$)

Regression models

Suppose we have N documents, with each document i having label
$y_i \in \{-1, 1\} \rightsquigarrow$ {negative, positive}
We represent each document i is $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{iJ})$.

$$\begin{aligned} f(\boldsymbol{\beta}, \boldsymbol{X}, \boldsymbol{Y}) &= \sum_{i=1}^{N} \left(y_i - \boldsymbol{\beta}' \boldsymbol{x}_i\right)^2 \\ \widehat{\boldsymbol{\beta}} &= \text{arg min}_{\boldsymbol{\beta}} \left\{ \sum_{i=1}^{N} \left(y_i - \boldsymbol{\beta}' \boldsymbol{x}_i\right)^2 \right\} \end{aligned} \tag{1}$$

Problem:

- J will likely be large (perhaps even $J > N$)
- There many correlated variables

Regression models

Suppose we have N documents, with each document i having label $y_i \in \{-1, 1\} \rightsquigarrow \{\text{negative, positive}\}$
We represent each document i is $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{iJ})$.

$$
\begin{aligned}
f(\boldsymbol{\beta}, \boldsymbol{X}, \boldsymbol{Y}) &= \sum_{i=1}^{N} \left(y_i - \boldsymbol{\beta}' \boldsymbol{x}_i\right)^2 \\
\widehat{\boldsymbol{\beta}} &= \text{arg min}_{\boldsymbol{\beta}} \left\{ \sum_{i=1}^{N} \left(y_i - \boldsymbol{\beta}' \boldsymbol{x}_i\right)^2 \right\} \qquad (1)
\end{aligned}
$$

Problem:

- J will likely be large (perhaps even $J > N$)
- There many correlated variables
- Predictions will be *variable*

Ridge (Penalized) Regression

Penalty for model complexity

Ridge (Penalized) Regression

Penalty for model complexity

$$f(\beta, \boldsymbol{X}, \boldsymbol{Y})$$

Ridge (Penalized) Regression

Penalty for model complexity

$$f(\boldsymbol{\beta}, \boldsymbol{X}, \boldsymbol{Y}) = \sum_{i=1}^{N}\left(y_i - \beta_0 - \sum_{j=1}^{J}\beta_j x_{ij}\right)^2$$

Ridge (Penalized) Regression

Penalty for model complexity

$$f(\boldsymbol{\beta}, \boldsymbol{X}, \boldsymbol{Y}) = \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{J} \beta_j x_{ij} \right)^2 + \underbrace{\lambda \sum_{j=1}^{J} \beta_j^2}_{\text{Penalty}}$$

Ridge (Penalized) Regression

Penalty for model complexity

$$f(\boldsymbol{\beta}, \boldsymbol{X}, \boldsymbol{Y}) \;=\; \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{J} \beta_j x_{ij} \right)^2 + \underbrace{\lambda \sum_{j=1}^{J} \beta_j^2}_{\text{Penalty}}$$

where:

Ridge (Penalized) Regression

Penalty for model complexity

$$f(\beta, \boldsymbol{X}, \boldsymbol{Y}) \;=\; \sum_{i=1}^{N}\left(y_i - \;\beta_0 - \sum_{j=1}^{J}\beta_j x_{ij}\right)^2 + \underbrace{\lambda\sum_{j=1}^{J}\beta_j^2}_{\text{Penalty}}$$

where:

- $\beta_0 \rightsquigarrow$ intercept

Ridge (Penalized) Regression

Penalty for model complexity

$$f(\boldsymbol{\beta}, \boldsymbol{X}, \boldsymbol{Y}) = \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{J} \beta_j x_{ij} \right)^2 + \underbrace{\lambda \sum_{j=1}^{J} \beta_j^2}_{\text{Penalty}}$$

where:

- $\beta_0 \rightsquigarrow$ intercept
- $\lambda \rightsquigarrow$ penalty parameter

Ridge (Penalized) Regression

Penalty for model complexity

$$f(\boldsymbol{\beta}, \boldsymbol{X}, \boldsymbol{Y}) = \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{J} \beta_j x_{ij} \right)^2 + \underbrace{\lambda \sum_{j=1}^{J} \beta_j^2}_{\text{Penalty}}$$

where:

- $\beta_0 \rightsquigarrow$ intercept
- $\lambda \rightsquigarrow$ penalty parameter
- Standardized $\boldsymbol{X}$ (coefficients on same scale)

Ridge Regression $\rightsquigarrow$ Optimization

$$\beta^{\text{Ridge}} = \arg\min_{\boldsymbol{\beta}} \{f(\boldsymbol{\beta}, \boldsymbol{X}, \boldsymbol{Y})\}$$

Ridge Regression ⇝ Optimization

$$
\begin{aligned}
\boldsymbol{\beta}^{\text{Ridge}} &= \text{arg min}_{\boldsymbol{\beta}} \left\{ f(\boldsymbol{\beta}, \boldsymbol{X}, \boldsymbol{Y}) \right\} \\
&= \text{arg min}_{\boldsymbol{\beta}} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{J} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{J} \beta_j^2 \right\}
\end{aligned}
$$

Other Penalized Objective Functions

Different Penalty for Model Complexity: LASSO

$$f(\boldsymbol{\beta}, \boldsymbol{X}, \boldsymbol{Y}) = \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{J} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{J} \underbrace{|\beta_j|}_{\text{Penalty}}$$

Other Penalized Objective Functions

Different Penalty for Model Complexity: LASSO

$$f(\boldsymbol{\beta}, \boldsymbol{X}, \boldsymbol{Y}) = \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{J} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{J} \underbrace{|\beta_j|}_{\text{Penalty}}$$

And combining the two criteria $\rightsquigarrow$ Elastic-Net

$$f(\boldsymbol{\beta}, \boldsymbol{X}, \boldsymbol{Y}) = \frac{1}{2N} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{J} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{J} \left(\frac{1}{2}(1-\alpha)\beta_j^2 + \alpha|\beta_j| \right)$$

Concluding Remarks

- Ultimately, via more complex supervised techniques, we can bypass the lack of optimization problem inherent in basic dictionary methods

Concluding Remarks

- Ultimately, via more complex supervised techniques, we can bypass the lack of optimization problem inherent in basic dictionary methods

- We can optimize soem cost function with respect to **context**, by incorporating explanatory features (*all* word frequencies) in the probability of assignment of some document to a class

Concluding Remarks

- Ultimately, via more complex supervised techniques, we can bypass the lack of optimization problem inherent in basic dictionary methods
- We can optimize soem cost function with respect to **context**, by incorporating explanatory features (*all* word frequencies) in the probability of assignment of some document to a class
- The result is more realistic estimated classes of text based on the entirety of the document, *not* isolated term frequencies

Concluding Remarks

- Ultimately, via more complex supervised techniques, we can bypass the lack of optimization problem inherent in basic dictionary methods
- We can optimize soem cost function with respect to **context**, by incorporating explanatory features (*all* word frequencies) in the probability of assignment of some document to a class
- The result is more realistic estimated classes of text based on the entirety of the document, *not* isolated term frequencies
- Further, we can do so in a statistically principled manner with penalities, though we can reach a similar conclusion with any number of classification techniques (e.g., naive Bayes)

Concluding Remarks

- Ultimately, via more complex supervised techniques, we can bypass the lack of optimization problem inherent in basic dictionary methods
- We can optimize soem cost function with respect to **context**, by incorporating explanatory features (*all* word frequencies) in the probability of assignment of some document to a class
- The result is more realistic estimated classes of text based on the entirety of the document, *not* isolated term frequencies
- Further, we can do so in a statistically principled manner with penalities, though we can reach a similar conclusion with any number of classification techniques (e.g., naive Bayes)
- Always think very carefully about the context of texts, assumptions of classifiers, and validation of any initial patterns by testing,

Concluding Remarks

- Ultimately, via more complex supervised techniques, we can bypass the lack of optimization problem inherent in basic dictionary methods
- We can optimize soem cost function with respect to **context**, by incorporating explanatory features (*all* word frequencies) in the probability of assignment of some document to a class
- The result is more realistic estimated classes of text based on the entirety of the document, *not* isolated term frequencies
- Further, we can do so in a statistically principled manner with penalities, though we can reach a similar conclusion with any number of classification techniques (e.g., naive Bayes)
- Always think very carefully about the context of texts, assumptions of classifiers, and validation of any initial patterns by testing, re-testing,

Concluding Remarks

- Ultimately, via more complex supervised techniques, we can bypass the lack of optimization problem inherent in basic dictionary methods
- We can optimize soem cost function with respect to **context**, by incorporating explanatory features (*all* word frequencies) in the probability of assignment of some document to a class
- The result is more realistic estimated classes of text based on the entirety of the document, *not* isolated term frequencies
- Further, we can do so in a statistically principled manner with penalities, though we can reach a similar conclusion with any number of classification techniques (e.g., naive Bayes)
- Always think very carefully about the context of texts, assumptions of classifiers, and validation of any initial patterns by testing, re-testing, re-testing,

Concluding Remarks

- Ultimately, via more complex supervised techniques, we can bypass the lack of optimization problem inherent in basic dictionary methods
- We can optimize soem cost function with respect to **context**, by incorporating explanatory features (*all* word frequencies) in the probability of assignment of some document to a class
- The result is more realistic estimated classes of text based on the entirety of the document, *not* isolated term frequencies
- Further, we can do so in a statistically principled manner with penalities, though we can reach a similar conclusion with any number of classification techniques (e.g., naive Bayes)
- Always think very carefully about the context of texts, assumptions of classifiers, and validation of any initial patterns by testing, re-testing, re-testing, and... you get the idea

Lecture Outline

- R
  - tm package (e.g., Corpus(), DocumentTermMatrix(), etc.)
  - tidytext package (e.g., get_sentiments(), etc.)
  - wordcloud package
  - (more for next class, but still for your problem set) topicmodels
  - stm package for structural topic models (may or may not get there)
- Python:
  - NLTK
  - spaCy
  - Scikit-Learn
  - For viz: Matplotlib and Seaborn