Text as Data: An Applied Introduction

Rebecca Johnson[1]

AU Data Science Institute. January 2020

[1] Assistant Professor (Summer 2020), Quantitative Social Science, Dartmouth College and Associate Fellow, Office of Evaluation Sciences at GSA. Thanks to Brandon Stewart for topic modeling slides, with the slides I flag taken from *LDA and Beyond: Topic Models in the Social Sciences*, Summer Institute in Computational Social Science 2019

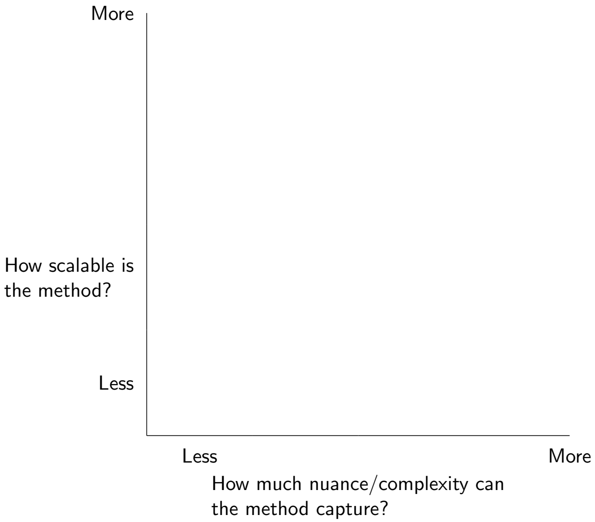
Outline

- Computational text analysis: why, and where from?
- Computational text analysis: how? Two categories:
  - Text mining/supervised: how can we search for (1) things we know in advance we want to look for and can (2) easily operationalize/define?
    - Intro or review or basic `Python` data structures and control flow
    - Workhorse tools for basic text mining: `re` (regular expressions); pandas str operations
  - Unsupervised: how can we more inductively discover themes/patterns in texts?
    - Preprocessing to prepare for a topic model: overview
    - Preprocessing: mechanics with nltk in Python
    - Topic modeling: concepts
    - Topic modeling: extension to structural topic model

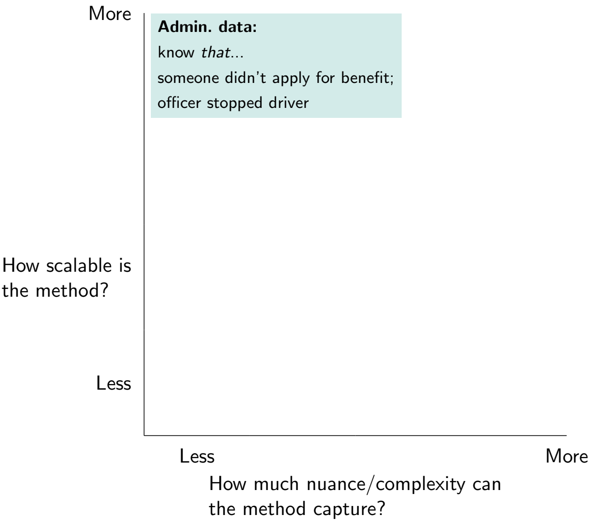
Outline

- **Computational text analysis: why, and where from?**
- Computational text analysis: how? Two categories:
  - Text mining/supervised: how can we search for (1) things we know in advance we want to look for and can (2) easily operationalize/define?
    - Intro or review or basic `Python` data structures and control flow
    - Workhorse tools for basic text mining: `re` (regular expressions); pandas str operations
  - Unsupervised: how can we more inductively discover themes/patterns in texts?
    - Preprocessing to prepare for a topic model: overview
    - Preprocessing: mechanics with nltk in Python
    - Topic modeling: concepts
    - Topic modeling: extension to structural topic model

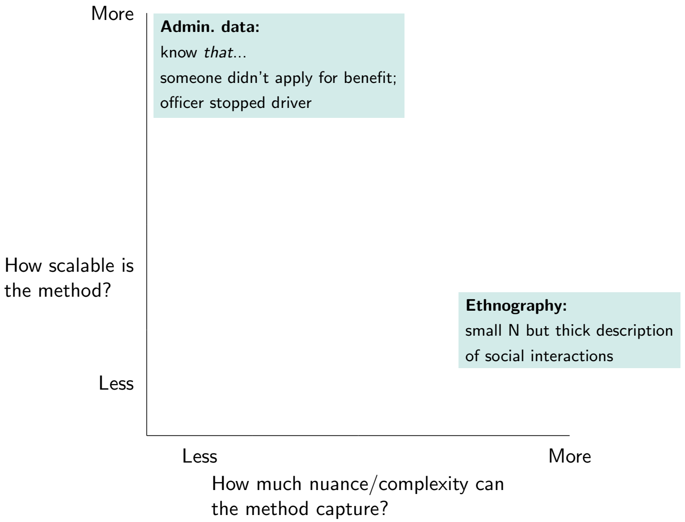
Why computational text analysis?



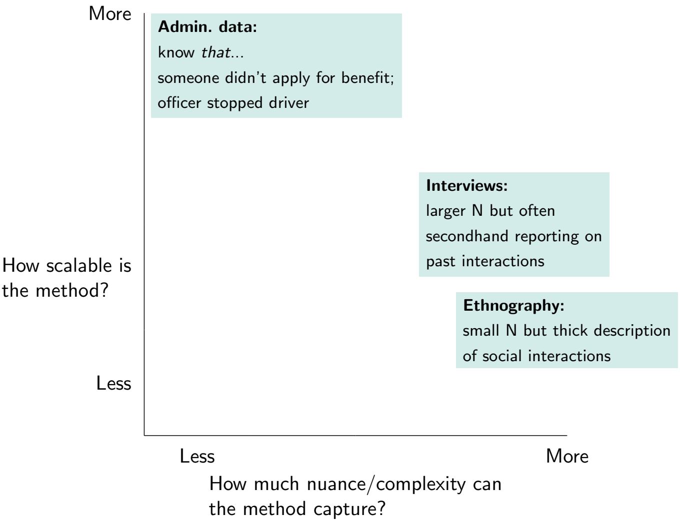
Why computational text analysis?



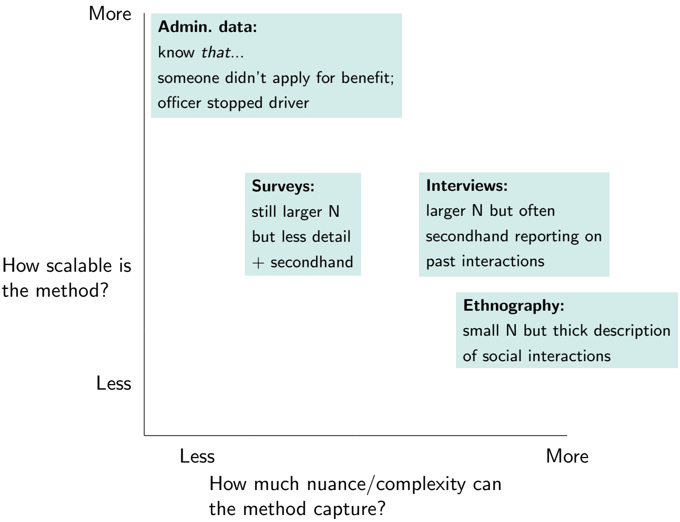
Why computational text analysis?



Why computational text analysis?



Why computational text analysis?

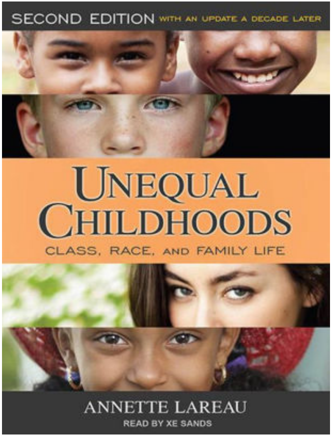


Why computational text analysis?

Example with inequality and education

Ethnography: more nuance, small scale

Size: 8 families
Method: followed parents in their interactions with teachers, doctors, and others who made decisions that impacted the parent's child



Surveys: less nuance; medium scale

Size: can be fielded to many parents but expensive
Method: yes/no self report

Parental Involvement Variables			
Parents attended PTA meeting, 1996	0–1	.550	.498
Parents attended one-on-one meeting with teacher or school official, 1996	0–1	.937	.242
Parents volunteered in classroom, 1996	0–1	.608	.488
Parents volunteered outside the classroom, 1996	0–1	.641	.480
How often parents helped with homework, 1996	0–5	3.11	1.72
How often parents checked homework, 1996	0–5	3.88	1.65

Administrative "metadata": less nuance; large scale

Size: if digitized, collected inexpensively from many schools
Method: direct observation of *that* parent met with teacher; no observation of *what* happened

Parents School Meeting Sign In Sheet

Day/Date :

Parents Name	Kids Name	Kids Class	Sign

Computational text analysis

- Less nuanced than an ethnographer following a parent around and observing his/her interactions with teachers and their child

Computational text analysis

- Less nuanced than an ethnographer following a parent around and observing his/her interactions with teachers and their child
- More nuanced than a measure of *whether* a parent met with their child's teacher, since can give insight into content of interaction

Computational text analysis

- Less nuanced than an ethnographer following a parent around and observing his/her interactions with teachers and their child
- More nuanced than a measure of *whether* a parent met with their child's teacher, since can give insight into content of interaction
- Less potential for bias than parent reports of whether they met with a teacher

Computational text analysis

- Less nuanced than an ethnographer following a parent around and observing his/her interactions with teachers and their child
- More nuanced than a measure of *whether* a parent met with their child's teacher, since can give insight into content of interaction
- Less potential for bias than parent reports of whether they met with a teacher
- *Possible sources:*

Computational text analysis

- Less nuanced than an ethnographer following a parent around and observing his/her interactions with teachers and their child
- More nuanced than a measure of *whether* a parent met with their child's teacher, since can give insight into content of interaction
- Less potential for bias than parent reports of whether they met with a teacher
- *Possible sources:*
  - Emails or text messages between parents and teachers about the child's progress or disciplinary issues (informal, frequent interactions)

Computational text analysis

- Less nuanced than an ethnographer following a parent around and observing his/her interactions with teachers and their child
- More nuanced than a measure of *whether* a parent met with their child's teacher, since can give insight into content of interaction
- Less potential for bias than parent reports of whether they met with a teacher
- *Possible sources:*
  - Emails or text messages between parents and teachers about the child's progress or disciplinary issues (informal, frequent interactions)
  - Transcripts of parent-teacher conferences (more formal, less frequent interactions); transcripts of conversations in the child's home (informal, frequent interactions; Princeton Video project)

Computational text analysis

- Less nuanced than an ethnographer following a parent around and observing his/her interactions with teachers and their child
- More nuanced than a measure of *whether* a parent met with their child's teacher, since can give insight into content of interaction
- Less potential for bias than parent reports of whether they met with a teacher
- *Possible sources:*
    - Emails or text messages between parents and teachers about the child's progress or disciplinary issues (informal, frequent interactions)
    - Transcripts of parent-teacher conferences (more formal, less frequent interactions); transcripts of conversations in the child's home (informal, frequent interactions; Princeton Video project)
    - Text of complaints parents file about things services they believe their child deserves (more formal, less frequent interactions)– basis for your hands-on activity

Where can you find text to use as data?

- General guide: just as an ethnographer thinks carefully about a field site, begin with your substantive interests–e.g., how do police treat residents of different races? What issues do low-income tenants face?–and think about text generated as things unfold in that area

Where can you find text to use as data?

- General guide: just as an ethnographer thinks carefully about a field site, begin with your substantive interests–e.g., how do police treat residents of different races? What issues do low-income tenants face?–and think about text generated as things unfold in that area
- Two broad types:

Where can you find text to use as data?

- General guide: just as an ethnographer thinks carefully about a field site, begin with your substantive interests–e.g., how do police treat residents of different races? What issues do low-income tenants face?–and think about text generated as things unfold in that area
- Two broad types:
  1. **One-way text outputs:** official documents (e.g., legislation; news articles; court cases); informal broadcasts (tweets, Yelp reviews, 311 complaints, and other social media data); informal notes professionals write about clients (e.g., caseworker notes; free text fields in medical records)

Where can you find text to use as data?

- General guide: just as an ethnographer thinks carefully about a field site, begin with your substantive interests–e.g., how do police treat residents of different races? What issues do low-income tenants face?–and think about text generated as things unfold in that area
- Two broad types:
  1. **One-way text outputs:** official documents (e.g., legislation; news articles; court cases); informal broadcasts (tweets, Yelp reviews, 311 complaints, and other social media data); informal notes professionals write about clients (e.g., caseworker notes; free text fields in medical records)
  2. **Two-way dialogues/interactions (may involve transforming video data $\implies$ audio data $\implies$ text)**: transcripts from body camera data (Voigt et al. 2017); transcripts from physician-patient conversations (Hagiwara et al. 2017); message board data (Dimaggio et al., 2019); Slack data

Where can you find, *openly-available* text to use as data?

- Usually combined with web scraping or interacting with an API to acquire efficiently

Where can you find, *openly-available* text to use as data?

- Usually combined with web scraping or interacting with an API to acquire efficiently
- Examples (first three provide text already stored in flat file form; fourth is .txt files; last two require scraping):

Where can you find, *openly-available* text to use as data?

- Usually combined with web scraping or interacting with an API to acquire efficiently
- Examples (first three provide text already stored in flat file form; fourth is .txt files; last two require scraping):
  1. Kaggle text data: DOJ press releases; IMDB movie reviews data

Where can you find, *openly-available* text to use as data?

- Usually combined with web scraping or interacting with an API to acquire efficiently
- Examples (first three provide text already stored in flat file form; fourth is .txt files; last two require scraping):
  1. Kaggle text data: DOJ press releases; IMDB movie reviews data
     - Example: "If you like original gut wrenching laughter you will like this movie. If you are young or old then you will love this movie, hell even my mom liked it."

Where can you find, *openly-available* text to use as data?

- Usually combined with web scraping or interacting with an API to acquire efficiently
- Examples (first three provide text already stored in flat file form; fourth is .txt files; last two require scraping):
    1. Kaggle text data: DOJ press releases; IMDB movie reviews data
        - Example: “If you like original gut wrenching laughter you will like this movie. If you are young or old then you will love this movie, hell even my mom liked it.”
    2. Restaurant reviews dataset

Where can you find, *openly-available* text to use as data?

- Usually combined with web scraping or interacting with an API to acquire efficiently
- Examples (first three provide text already stored in flat file form; fourth is .txt files; last two require scraping):
  1. Kaggle text data: DOJ press releases; IMDB movie reviews data
     - Example: "If you like original gut wrenching laughter you will like this movie. If you are young or old then you will love this movie, hell even my mom liked it."
  2. Restaurant reviews dataset
  3. NYC housing code violations data

Where can you find, *openly-available* text to use as data?

- Usually combined with web scraping or interacting with an API to acquire efficiently
- Examples (first three provide text already stored in flat file form; fourth is .txt files; last two require scraping):
  1. Kaggle text data: DOJ press releases; IMDB movie reviews data
     - Example: "If you like original gut wrenching laughter you will like this movie. If you are young or old then you will love this movie, hell even my mom liked it."
  2. Restaurant reviews dataset
  3. NYC housing code violations data
     - Example: "Abate the nuisance consisting of roaches in the entire apartment)

Where can you find, *openly-available* text to use as data?

- Usually combined with web scraping or interacting with an API to acquire efficiently
- Examples (first three provide text already stored in flat file form; fourth is .txt files; last two require scraping):
  1. Kaggle text data: DOJ press releases; IMDB movie reviews data
     - Example: "If you like original gut wrenching laughter you will like this movie. If you are young or old then you will love this movie, hell even my mom liked it."
  2. Restaurant reviews dataset
  3. NYC housing code violations data
     - Example: "Abate the nuisance consisting of roaches in the entire apartment)
  4. Congressional bills data

Where can you find, *openly-available* text to use as data?

- Usually combined with web scraping or interacting with an API to acquire efficiently
- Examples (first three provide text already stored in flat file form; fourth is .txt files; last two require scraping):
  1. Kaggle text data: DOJ press releases; IMDB movie reviews data
     - Example: "If you like original gut wrenching laughter you will like this movie. If you are young or old then you will love this movie, hell even my mom liked it."
  2. Restaurant reviews dataset
  3. NYC housing code violations data
     - Example: "Abate the nuisance consisting of roaches in the entire apartment)
  4. Congressional bills data
  5. Tutorial on scraping Craigslist, which can be used to study things like how people describe gentrifying neighborhoods

Where can you find, *openly-available* text to use as data?

- Usually combined with web scraping or interacting with an API to acquire efficiently
- Examples (first three provide text already stored in flat file form; fourth is .txt files; last two require scraping):
  1. Kaggle text data: DOJ press releases; IMDB movie reviews data
     - Example: "If you like original gut wrenching laughter you will like this movie. If you are young or old then you will love this movie, hell even my mom liked it."
  2. Restaurant reviews dataset
  3. NYC housing code violations data
     - Example: "Abate the nuisance consisting of roaches in the entire apartment)
  4. Congressional bills data
  5. Tutorial on scraping Craigslist, which can be used to study things like how people describe gentrifying neighborhoods
  6. DC urban mom and dad: "Boring is obviously subjective but most of the fast casual places in Tenleytown are actually local chains except for Panera. While I would prefer fine dining in TT..."

Outline

- Computational text analysis: why, and where from?
- **Computational text analysis: how? Two categories:**
  - **Text mining/supervised: how can we search for (1) things we know in advance we want to look for and can (2) easily operationalize/define?**
    - Intro or review or basic `Python` data structures and control flow
    - Workhorse tools for basic text mining: `re` (regular expressions); pandas str operations
  - Unsupervised: how can we more inductively discover themes/patterns in texts?
    - Preprocessing to prepare for a topic model: overview
    - Preprocessing: mechanics with nltk in Python
    - Topic modeling: concepts
    - Topic modeling: extension to structural topic model

What do I mean by "supervised"?

1. *Text mining:* look for a pre-specified concept or category. Methods:

What do I mean by "supervised"?

1. *Text mining:* look for a pre-specified concept or category. Methods:
   - Pattern matching: look for a match for a specific sequence of characters

What do I mean by “supervised”?

1. *Text mining:* look for a pre-specified concept or category. Methods:
   - Pattern matching: look for a match for a specific sequence of characters
   - Dictionary: we have a list of words we think represent a category or concept (e.g., if we want to classify a review as negative, we might have a list of words or phrases we think represent the category like *boring*; *terrible*; *awful*; *literally the worst*)

What do I mean by "supervised"?

1. *Text mining:* look for a pre-specified concept or category. Methods:
   - Pattern matching: look for a match for a specific sequence of characters
   - Dictionary: we have a list of words we think represent a category or concept (e.g., if we want to classify a review as negative, we might have a list of words or phrases we think represent the category like *boring*; *terrible*; *awful*; *literally the worst*)
2. *Supervised machine learning for classification:* pre-specify a category at the document level and learn how text predicts that category

What do I mean by "supervised"?

1. *Text mining:* look for a pre-specified concept or category. Methods:
   - Pattern matching: look for a match for a specific sequence of characters
   - Dictionary: we have a list of words we think represent a category or concept (e.g., if we want to classify a review as negative, we might have a list of words or phrases we think represent the category like *boring*; *terrible*; *awful*; *literally the worst*)
2. *Supervised machine learning for classification:* pre-specify a category at the document level and learn how text predicts that category
   - Inputs, training data:

What do I mean by "supervised"?

1. *Text mining:* look for a pre-specified concept or category. Methods:
   - Pattern matching: look for a match for a specific sequence of characters
   - Dictionary: we have a list of words we think represent a category or concept (e.g., if we want to classify a review as negative, we might have a list of words or phrases we think represent the category like *boring*; *terrible*; *awful*; *literally the worst*)
2. *Supervised machine learning for classification:* pre-specify a category at the document level and learn how text predicts that category
   - Inputs, training data:
     - Text: movie review; legislative bill; message board chain; admissions essay

What do I mean by "supervised"?

1. *Text mining:* look for a pre-specified concept or category. Methods:
   - Pattern matching: look for a match for a specific sequence of characters
   - Dictionary: we have a list of words we think represent a category or concept (e.g., if we want to classify a review as negative, we might have a list of words or phrases we think represent the category like *boring*; *terrible*; *awful*; *literally the worst*)
2. *Supervised machine learning for classification:* pre-specify a category at the document level and learn how text predicts that category
   - Inputs, training data:
     - Text: movie review; legislative bill; message board chain; admissions essay
     - Label for that text: negative or not; Repub. sponsor or not; contentious or not; accepted or not

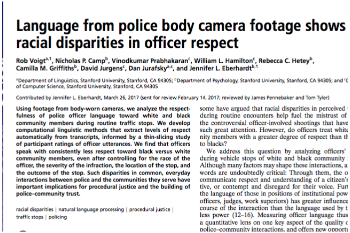
What do I mean by "supervised"?

1. *Text mining:* look for a pre-specified concept or category. Methods:
    - Pattern matching: look for a match for a specific sequence of characters
    - Dictionary: we have a list of words we think represent a category or concept (e.g., if we want to classify a review as negative, we might have a list of words or phrases we think represent the category like *boring*; *terrible*; *awful*; *literally the worst*)
2. *Supervised machine learning for classification:* pre-specify a category at the document level and learn how text predicts that category
    - Inputs, training data:
        - Text: movie review; legislative bill; message board chain; admissions essay
        - Label for that text: negative or not; Repub. sponsor or not; contentious or not; accepted or not
    - Method: often binary classification

What do I mean by "supervised"?

1. *Text mining:* look for a pre-specified concept or category. Methods:
   - Pattern matching: look for a match for a specific sequence of characters
   - Dictionary: we have a list of words we think represent a category or concept (e.g., if we want to classify a review as negative, we might have a list of words or phrases we think represent the category like *boring*; *terrible*; *awful*; *literally the worst*)
2. *Supervised machine learning for classification:* pre-specify a category at the document level and learn how text predicts that category
   - Inputs, training data:
     - Text: movie review; legislative bill; message board chain; admissions essay
     - Label for that text: negative or not; Repub. sponsor or not; contentious or not; accepted or not
   - Method: often binary classification
   - Output: classifier that one can use for unlabeled data

Example of combining text mining with supervised ML



Language from police body camera footage shows racial disparities in officer respect

Rob Voigt[a,1], Nicholas P. Camp[b], Vinodkumar Prabhakaran[c], William L. Hamilton[c], Rebecca C. Hetey[b], Camilla M. Griffiths[b], David Jurgens[c], Dan Jurafsky[a,c], and Jennifer L. Eberhardt[b,1]

[a]Department of Linguistics, Stanford University, Stanford, CA 94305; [b]Department of Psychology, Stanford University, Stanford, CA 94305; and [c]D of Computer Science, Stanford University, Stanford, CA 94305

Contributed by Jennifer L. Eberhardt, March 26, 2017 (sent for review February 14, 2017; reviewed by James Pennebaker and Tom Tyler)

Using footage from body-worn cameras, we analyze the respectfulness of police officer language toward white and black community members during routine traffic stops. We develop computational linguistic methods that extract levels of respect automatically from transcripts, informed by a thin-slicing study of participant ratings of officer utterances. We find that officers speak with consistently less respect toward black versus white community members, even after controlling for the race of the officer, the severity of the infraction, the location of the stop, and the outcome of the stop. Such disparities in common, everyday interactions between police and the communities they serve have important implications for procedural justice and the building of police–community trust.

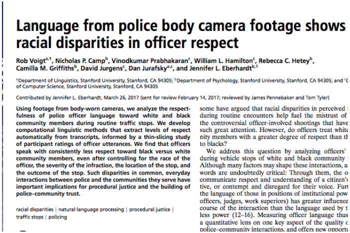
racial disparities | natural language processing | procedural justice | traffic stops | policing

some have argued that racial disparities in perceived during routine encounters help fuel the mistrust of the controversial officer-involved shootings that have such great attention. However, do officers treat white nity members with a greater degree of respect than th to blacks?

We address this question by analyzing officers' during vehicle stops of white and black community Although many factors may shape these interactions, a words are undoubtedly critical: Through them, the o communicate respect and understanding of a citizen's tive, or contempt and disregard for their voice. Furt the language of those in positions of institutional pow officers, judges, work superiors) has greater influence course of the interaction than the language used by t less power (12–16). Measuring officer language thus a quantitative lens on one key aspect of the quality c police–community interactions, and offers new opportu

- Had human raters code snippets of transcripts to generate labels of whether the interaction was "respectful" or not in a smaller sample of documents

Example of combining text mining with supervised ML



Language from police body camera footage shows racial disparities in officer respect

Rob Voigt[a,1], Nicholas P. Camp[b], Vinodkumar Prabhakaran[c], William L. Hamilton[c], Rebecca C. Hetey[b], Camilla M. Griffiths[b], David Jurgens[c], Dan Jurafsky[a,c], and Jennifer L. Eberhardt[b,1]

[a]Department of Linguistics, Stanford University, Stanford, CA 94305; [b]Department of Psychology, Stanford University, Stanford, CA 94305; and [c]D
of Computer Science, Stanford University, Stanford, CA 94305

Contributed by Jennifer L. Eberhardt, March 26, 2017 (sent for review February 14, 2017; reviewed by James Pennebaker and Tom Tyler)

Using footage from body-worn cameras, we analyze the respectfulness of police officer language toward white and black community members during routine traffic stops. We develop computational linguistic methods that extract levels of respect automatically from transcripts, informed by a thin-slicing study of participant ratings of officer utterances. We find that officers speak with consistently less respect toward black versus white community members, even after controlling for the race of the officer, the severity of the infraction, the location of the stop, and the outcome of the stop. Such disparities in common, everyday interactions between police and the communities they serve have important implications for procedural justice and the building of police–community trust.

racial disparities | natural language processing | procedural justice | traffic stops | policing

some have argued that racial disparities in perceived
during routine encounters help fuel the mistrust of
the controversial officer-involved shootings that have
such great attention. However, do officers treat white
nity members with a greater degree of respect than th
to blacks?

We address this question by analyzing officers'
during vehicle stops of white and black community
Although many factors may shape these interactions, a
words are undoubtedly critical: Through them, the o
communicate respect and understanding of a citizen's
tive, or contempt and disregard for their voice. Furt
the language of those in positions of institutional pow
officers, judges, work superiors) has greater influence
course of the interaction than the language used by t
less power (12–16). Measuring officer language thus
a quantitative lens on one key aspect of the quality o
police–community interactions, and offers new opportu

- Had human raters code snippets of transcripts to generate labels of whether the interaction was "respectful" or not in a smaller sample of documents
- Generated features from the text using dictionary-based methods, e.g.

Example of combining text mining with supervised ML

Language from police body camera footage shows racial disparities in officer respect

Rob Voigt[a,1], Nicholas P. Camp[b], Vinodkumar Prabhakaran[c], William L. Hamilton[c], Rebecca C. Hetey[b], Camilla M. Griffiths[b], David Jurgens[c], Dan Jurafsky[a,c], and Jennifer L. Eberhardt[b,1]

[a]Department of Linguistics, Stanford University, Stanford, CA 94305; [b]Department of Psychology, Stanford University, Stanford, CA 94305; and [c]D of Computer Science, Stanford University, Stanford, CA 94305

Contributed by Jennifer L. Eberhardt, March 26, 2017 (sent for review February 14, 2017; reviewed by James Pennebaker and Tom Tyler)

Using footage from body-worn cameras, we analyze the respectfulness of police officer language toward white and black community members during routine traffic stops. We develop computational linguistic methods that extract levels of respect automatically from transcripts, informed by a thin-slicing study of participant ratings of officer utterances. We find that officers speak with consistently less respect toward black versus white community members, even after controlling for the race of the officer, the severity of the infraction, the location of the stop, and the outcome of the stop. Such disparities in common, everyday interactions between police and the communities they serve have important implications for procedural justice and the building of police–community trust.

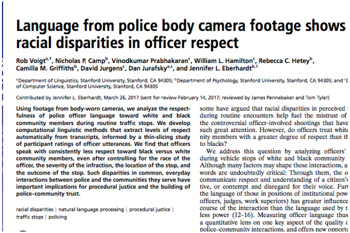
racial disparities | natural language processing | procedural justice | traffic stops | policing

some have argued that racial disparities in perceived during routine encounters help fuel the mistrust of the controversial officer-involved shootings that have such great attention. However, do officers treat white nity members with a greater degree of respect than th to blacks?

We address this question by analyzing officers' during vehicle stops of white and black community Although many factors may shape these interactions, a words are undoubtedly critical: Through them, the o communicate respect and understanding of a citizen's tive, or contempt and disregard for their voice. Furt the language of those in positions of institutional pow officers, judges, work superiors) has greater influence course of the interaction than the language used by t less power (12–16). Measuring officer language thus a quantitative lens on one key aspect of the quality c police–community interactions, and offers new opportu

- Had human raters code snippets of transcripts to generate labels of whether the interaction was "respectful" or not in a smaller sample of documents
- Generated features from the text using dictionary-based methods, e.g.
  - Informal titles: ["dude*", "bro*", "boss", "bud", "buddy", "champ", "man", "guy*", "guy", "brotha", "sista", "son", "sonny", "chief"]

Example of combining text mining with supervised ML



Language from police body camera footage shows racial disparities in officer respect

Rob Voigt[a,1], Nicholas P. Camp[b], Vinodkumar Prabhakaran[c], William L. Hamilton[c], Rebecca C. Hetey[b], Camilla M. Griffiths[b], David Jurgens[c], Dan Jurafsky[a,c], and Jennifer L. Eberhardt[b,1]

[a]Department of Linguistics, Stanford University, Stanford, CA 94305; [b]Department of Psychology, Stanford University, Stanford, CA 94305; and [c]D
of Computer Science, Stanford University, Stanford, CA 94305

Contributed by Jennifer L. Eberhardt, March 26, 2017 (sent for review February 14, 2017; reviewed by James Pennebaker and Tom Tyler)

Using footage from body-worn cameras, we analyze the respectfulness of police officer language toward white and black community members during routine traffic stops. We develop computational linguistic methods that extract levels of respect automatically from transcripts, informed by a thin-slicing study of participant ratings of officer utterances. We find that officers speak with consistently less respect toward black versus white community members, even after controlling for the race of the officer, the severity of the infraction, the location of the stop, and the outcome of the stop. Such disparities in common, everyday interactions between police and the communities they serve have important implications for procedural justice and the building of police–community trust.

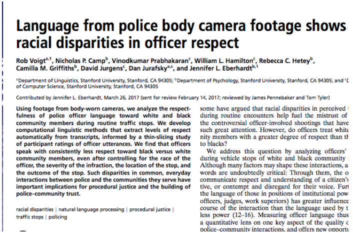
racial disparities | natural language processing | procedural justice | traffic stops | policing

some have argued that racial disparities in perceived
during routine encounters help fuel the mistrust of
the controversial officer-involved shootings that have
such great attention. However, do officers treat whit
nity members with a greater degree of respect than th
to blacks?

We address this question by analyzing officers'
during vehicle stops of white and black community
Although many factors may shape these interactions, a
words are undoubtedly critical: Through them, the o
communicate respect and understanding of a citizen's
tive, or contempt and disregard for their voice. Furt
the language of those in positions of institutional pow
officers, judges, work superiors) has greater influence
course of the interaction than the language used by t
less power (12–16). Measuring officer language thus
a quantitative lens on one key aspect of the quality c
police–community interactions, and offers new opport

- Had human raters code snippets of transcripts to generate labels of whether the interaction was "respectful" or not in a smaller sample of documents
- Generated features from the text using dictionary-based methods, e.g.
  - Informal titles: ["dude*", "bro*", "boss", "bud", "buddy", "champ", "man", "guy*", "guy", "brotha", "sista", "son", "sonny", "chief"]
  - Time minimizing: "(minute—min—second—sec—moment)s?—this[^ .,?!]+quick—right back"

Example of combining text mining with supervised ML



Language from police body camera footage shows racial disparities in officer respect

Rob Voigt[a,1], Nicholas P. Camp[b], Vinodkumar Prabhakaran[c], William L. Hamilton[c], Rebecca C. Hetey[b], Camilla M. Griffiths[b], David Jurgens[c], Dan Jurafsky[a,c], and Jennifer L. Eberhardt[b,1]

[a]Department of Linguistics, Stanford University, Stanford, CA 94305; [b]Department of Psychology, Stanford University, Stanford, CA 94305; and [c]D of Computer Science, Stanford University, Stanford, CA 94305

Contributed by Jennifer L. Eberhardt, March 26, 2017 (sent for review February 14, 2017; reviewed by James Pennebaker and Tom Tyler)

Using footage from body-worn cameras, we analyze the respectfulness of police officer language toward white and black community members during routine traffic stops. We develop computational linguistic methods that extract levels of respect automatically from transcripts, informed by a thin-slicing study of participant ratings of officer utterances. We find that officers speak with consistently less respect toward black versus white community members, even after controlling for the race of the officer, the severity of the infraction, the location of the stop, and the outcome of the stop. Such disparities in common, everyday interactions between police and the communities they serve have important implications for procedural justice and the building of police–community trust.

racial disparities | natural language processing | procedural justice | traffic stops | policing

some have argued that racial disparities in perceived
during routine encounters help fuel the mistrust of
the controversial officer-involved shootings that have
such great attention. However, do officers treat white
nity members with a greater degree of respect than th
to blacks?

We address this question by analyzing officers'
during vehicle stops of white and black community
Although many factors may shape these interactions, a
words are undoubtedly critical: Through them, the o
communicate respect and understanding of a citizen's
tive, or contempt and disregard for their voice. Furt
the language of those in positions of institutional pow
officers, judges, work superiors) has greater influence
course of the interaction than the language used by t
less power (12–16). Measuring officer language thus
a quantitative lens on one key aspect of the quality o
police–community interactions, and offers new opportu

- Had human raters code snippets of transcripts to generate labels of whether the interaction was "respectful" or not in a smaller sample of documents
- Generated features from the text using dictionary-based methods, e.g.
  - Informal titles: ["dude*", "bro*", "boss", "bud", "buddy", "champ", "man", "guy*", "guy", "brotha", "sista", "son", "sonny", "chief"]
  - Time minimizing: "(minute—min—second—sec—moment)s?—this[^ .,?!]+quick—right back"
- Built model to predict respect ratings using those features

Hands-on activity: using text mining to learn basic features of documents

Teaching Parents Of Kids With Disabilities To Fight Back

June 29, 2018 · 1:35 PM ET
Heard on All Things Considered


JOSEPH SHAPIRO

- Disability rights movement, prompted by scandals like Willowbrook where children were neglected in large institutions, emphasized (1) giving children rights to receive services in public schools, (2) giving parents rights to make sure their child gets the resources he or she needs

Hands-on activity: using text mining to learn basic features of documents

Teaching Parents Of Kids With Disabilities To Fight Back

June 29, 2018 · 1:35 PM ET
Heard on All Things Considered


JOSEPH SHAPIRO

- Disability rights movement, prompted by scandals like Willowbrook where children were neglected in large institutions, emphasized (1) giving children rights to receive services in public schools, (2) giving parents rights to make sure their child gets the resources he or she needs
- Schools are legally obligated to give "appropriate" resources, but have discretion over what counts as appropriate. Amidst budget pressure, they may use that discretion to cut corners/ration

Hands-on activity: using text mining to learn basic features of documents

Teaching Parents Of Kids With Disabilities To Fight Back

June 29, 2018 · 1:35 PM ET
Heard on All Things Considered


JOSEPH SHAPIRO

- Disability rights movement, prompted by scandals like Willowbrook where children were neglected in large institutions, emphasized (1) giving children rights to receive services in public schools, (2) giving parents rights to make sure their child gets the resources he or she needs
- Schools are legally obligated to give "appropriate" resources, but have discretion over what counts as appropriate. Amidst budget pressure, they may use that discretion to cut corners/ration
- Parents can monitor this and file formal complaints against the district; subset of those make it to a hearing that generates a written record of the parent's allegations of what the district was doing wrong and the district's defense of its decisions

Acquiring the data: web scraping

Hearing Officer Determinations

Below are Hearing Officer Determinations for calendar year 2009-2019:

2009

2010

2011

2012

2013

2014

2015

2016

2017

2018

Acquiring the data: web scraping

Listen

SHARE

2019 Hearing Officers Determinations

Below are the Hearing Officer Documents by month

[January 2019]

[February 2019]

[March 2019]

[April 2019]

[May 2019]

[June 2019]

[July 2019]

[August 2019]

[September 2019]

July 2019 Hearing Officers Determinations

Monday, August 26, 2019

Below are the July 2019 Hearing Officers Determinations:

Related Content: 2019 Hearing Officer Determinations

Attachment(s):

- HOD July 2019 (1).pdf - 746.2 KB (pdf)
- HOD July 2019 (2).pdf - 920.9 KB (pdf)
- HOD July 2019 (3).pdf - 732.0 KB (pdf)
- HOD July 2019 (4).pdf - 821.2 KB (pdf)
- HOD July 2019 (5).pdf - 832.9 KB (pdf)
- HOD July 2019 (6).pdf - 812.5 KB (pdf)

Acquiring the data: code

```
## functions
def parse_https_page(link):

    parsed_page = BeautifulSoup(urlopen(Request(link, headers={'User-Agent':
                'Mozilla/5.0'})).read(), "html.parser")
    return(parsed_page)

def extract_links_frompage(parsed_page, pattern_tosearch):

    empty_list = []
    for link in parsed_page.findAll('a', attrs={'href':
    re.compile(pattern_tosearch)}):
        empty_list.append(link.get('href'))

    return(empty_list)

## example application
one_month_page = parse_https_page(one_month)
pdfs_onpage = extract_links_frompage(parsed_page = one_month_page,
            pattern_tosearch = "\\.pdf")
hod_pdfs = [pdf for pdf in pdfs_onpage if "HOD" in pdf]
```

Outline

- Computational text analysis: why, and where from?
- **Computational text analysis: how? Two categories:**
  - Text mining/supervised: how can we search for (1) things we know in advance we want to look for and can (2) easily operationalize/define?
    - **Intro or review or basic `Python` data structures and control flow**
    - Workhorse tools for basic text mining: `re` (regular expressions); pandas str operations
  - Unsupervised: how can we more inductively discover themes/patterns in texts?
    - Preprocessing to prepare for a topic model: overview
    - Preprocessing: mechanics with nltk in Python
    - Topic modeling: concepts
    - Topic modeling: extension to structural topic model

Can follow along in script 00_basicpython

Python data structures relevant for text analysis: lists

- Example code to create:

```
angry_words = ["mad", "annoyed", "whyyyyyy", "seriously"]
formality = ["formal", "formal", "informal", "informal"]
```

Python data structures relevant for text analysis: lists

- Example code to create:

```
angry_words = ["mad", "annoyed", "whyyyyyy", "seriously"]
formality = ["formal", "formal", "informal", "informal"]
```

- Code to index and augment:

```
]: print(angry_words[0])
   new_angry_words = angry_words + ["forgot this angry word"]
   new_angry_words

   mad

]: ['mad', 'annoyed', 'whyyyyyy', 'seriously', 'forgot this angry word']
```

Python data structures relevant for text analysis: lists

- Example code to create:

```
angry_words = ["mad", "annoyed", "whyyyyyy", "seriously"]
formality = ["formal", "formal", "informal", "informal"]
```

- Code to index and augment:

```
]: print(angry_words[0])
   new_angry_words = angry_words + ["forgot this angry word"]
   new_angry_words

   mad

]: ['mad', 'annoyed', 'whyyyyyy', 'seriously', 'forgot this angry word']
```

- What for? Can represent a corpus (collection of documents) as a list of individual documents, where each element is a string with the full document text; Can represent a document as a list of individual words or tokens

Task: begin with the sentence 'whyyyyy did you do that' and filter to words contained in the list of angry_words.

Iterating over lists

1. Represent the sentence as a string

```
sentence = "whyyyyy did you do that"
```

Iterating over lists

1. Represent the sentence as a string
   `sentence = "whyyyyy did you do that"`
2. Split the string into words, iterate over the words, and only keep words that are in the list angry_words:

```
## split sentence
split_sentence = sentence.split()
print(type(sentence))
print(type(split_sentence))
print(sentence)

## iterate over sentence and filter to
## only angry words
only_angry = [word for word in sentence.split() if
              word in angry_words]
only_angry
```

```
<class 'str'>
<class 'list'>
whyyyyyy did you do that
```

```
['whyyyyyy']
```

Iterating over lists

1. Represent the sentence as a string
   `sentence = "whyyyyy did you do that"`
2. Split the string into words, iterate over the words, and only keep words that are in the list angry_words:

```
## split sentence
split_sentence = sentence.split()
print(type(sentence))
print(type(split_sentence))
print(sentence)

## iterate over sentence and filter to
## only angry words
only_angry = [word for word in sentence.split() if
              word in angry_words]
only_angry
```

```
<class 'str'>
<class 'list'>
whyyyyyy did you do that
```

```
['whyyyyyy']
```

3. Can re-join output into string
   ```
   only_angry_sentence = " ".join(only_angry)
   only_angry_sentence
   ```

Python data structures relevant for text analysis: dictionaries

- Why use? While elements of lists can be duplicated, keys in a dictionary are unique. It's useful thus for storing individual terms as keys, with the values being the count of the word; here, we just use as an intermediate step to turn our two lists into a dataframe.

```
5]: formality_dictionary = {}
    formality_dictionary['angry_words'] = angry_words
    formality_dictionary['formality'] = formality

6]: formality_dictionary

6]: {'angry_words': ['mad', 'annoyed', 'whyyyyyy', 'seriously'],
     'formality': ['formal', 'formal', 'informal', 'informal']}
```

Python data structures relevant for text analysis: dataframes

```python
import pandas as pd
import numpy as np

formality_df = pd.DataFrame(formality_dictionary)
formality_df.head()
formality_df['formality_binary'] = np.where(formality_df.formality ==
                                            "formal", 1, 0)
formality_df['word_and_formality'] = formality_df.angry_words + "_" + \
                                     formality_df.formality
formality_df.head()
```

	angry_words	formality	formality_binary	word_and_formality
0	mad	formal	1	mad_formal
1	annoyed	formal	1	annoyed_formal
2	whyyyyyy	informal	0	whyyyyyy_informal
3	seriously	informal	0	seriously_informal

Outline

- Computational text analysis: why, and where from?
- **Computational text analysis: how? Two categories:**
  - Text mining/supervised: how can we search for (1) things we know in advance we want to look for and can (2) easily operationalize/define?
    - Intro or review or basic `Python` data structures and control flow
    - **Workhorse tools for basic text mining: `re` (regular expressions); pandas str operations**
  - Unsupervised: how can we more inductively discover themes/patterns in texts?
    - Preprocessing to prepare for a topic model: overview
    - Preprocessing: mechanics with nltk in Python
    - Topic modeling: concepts
    - Topic modeling: extension to structural topic model

re package implements the text wrangling Ryan covered

- First search a string for a pattern: `re.search("(pattern to search for)", string to search within)`

```
]: import re

   ## Task: find the part of the string with AU
   strings = ["AmericanUniversity_datascience",
              "Americanuniversity_datascience"]

   ## manually search each item
   au_pattern = "(American[u|U]niversity)"
   search_result = re.search(au_pattern,
                             strings[0])
   print(type(search_result))
   print(search_result.group(1))
   search_result = re.search(au_pattern,
                             strings[1])
   print(search_result.group(1))

   ## do via list iteration
   all_results = [re.search(au_pattern, one_string).group(1)
                  for one_string in strings]
   all_results

   <class '_sre.SRE_Match'>
   AmericanUniversity
   Americanuniversity

]: ['AmericanUniversity', 'Americanuniversity']
```

re package implements the text wrangling Ryan covered

- First search a string for a pattern: `re.search("(pattern to search for)", string to search within)`
- Then extract matches (in the above case, we're only trying to match one group) and can combine with list iteration

```
]: import re

   ## Task: find the part of the string with AU
   strings = ["AmericanUniversity_datascience",
              "Americanuniversity_datascience"]

   ## manually search each item
   au_pattern = "(American[u|U]niversity)"
   search_result = re.search(au_pattern,
                             strings[0])
   print(type(search_result))
   print(search_result.group(1))
   search_result = re.search(au_pattern,
                             strings[1])
   print(search_result.group(1))

   ## do via list iteration
   all_results = [re.search(au_pattern, one_string).group(1)
                  for one_string in strings]
   all_results

   <class '_sre.SRE_Match'>
   AmericanUniversity
   Americanuniversity

]: ['AmericanUniversity', 'Americanuniversity']
```

re package implements the text wrangling Ryan covered

- First search a string for a pattern: `re.search("(pattern to search for)", string to search within)`
- Then extract matches (in the above case, we're only trying to match one group) and can combine with list iteration
- Other re functions: re.findall(), re.sub(), re.split(), etc.

```
]: import re

   ## Task: find the part of the string with AU
   strings = ["AmericanUniversity_datascience",
              "Americanuniversity_datascience"]

   ## manually search each item
   au_pattern = "(American[u|U]niversity)"
   search_result = re.search(au_pattern,
                             strings[0])
   print(type(search_result))
   print(search_result.group(1))
   search_result = re.search(au_pattern,
                             strings[1])
   print(search_result.group(1))

   ## do via list iteration
   all_results = [re.search(au_pattern, one_string).group(1)
                  for one_string in strings]
   all_results
```

```
<class '_sre.SRE_Match'>
AmericanUniversity
Americanuniversity
```

```
]: ['AmericanUniversity', 'Americanuniversity']
```

`str` functions within pandas also help you do basic tasks with a column of text data

- Basic structure: `df.stringvar.str.operation("pattern")`

`str` functions within pandas also help you do basic tasks with a column of text data

- Basic structure: `df.stringvar.str.operation("pattern")`
- Common operations: str.lower(); str.upper(); str.contains(); str.len()...

`str` functions within pandas also help you do basic tasks with a column of text data

- Basic structure: `df.stringvar.str.operation("pattern")`
- Common operations: str.lower(); str.upper(); str.contains(); str.len()...
- Task example: convert names of Airbnb listings to lowercase and search for word "cozy"

`str` functions within pandas also help you do basic tasks with a column of text data

```
## view head of data
rel_cols = ['name'] + [col for col in airbnb_nyc.columns if
                        "neighbourhood" in col]
airbnb_nyc[rel_cols].head()

## convert all words in the listing name to lowercase
airbnb_nyc['listing_lower'] = airbnb_nyc.name.str.lower()

## create a binary indicator for whether the review mentions
## the word "cozy"
airbnb_nyc['describes_cozy'] = np.where(airbnb_nyc.listing_lower.\
                                        str.contains("cozy"),
                                        1, 0)
```

	name	neighbourhood_group	neighbourhood

str functions within pandas also help you do basic tasks with a column of text data

```
: airbnb_nyc[['listing_lower'] + ['describes_cozy']].head()
```

:

	listing_lower	describes_cozy
0	clean & quiet apt home by the park	0
1	skylit midtown castle	0
2	the village of harlem....new york !	0
3	cozy entire floor of brownstone	1
4	entire apt: spacious studio/loft by central park	0

Can then do basic aggregation for tasks like: which neighborhoods have high rates of listings described as cozy?

- Count within each category: `df.varname.value_counts()`

Can then do basic aggregation for tasks like: which neighborhoods have high rates of listings described as cozy?

- Count within each category: `df.varname.value_counts()`
- Count by grouping variable: `pd.crosstab(df.catorbinaryvar, df.catorbinaryvar)`

Can then do basic aggregation for tasks like: which neighborhoods have high rates of listings described as cozy?

- Count within each category: `df.varname.value_counts()`
- Count by grouping variable: `pd.crosstab(df.catorbinaryvar, df.catorbinaryvar)`
- Proportion by grouping variable: add `normalize` argument to `pd.crosstab` and normalize by row or column

Can then do basic aggregation

```
# summarise proportion by neighborhood
irbnb_nyc.describes_cozy.value_counts()
ount_neigh = pd.crosstab(airbnb_nyc.neighbourhood_group,
                         airbnb_nyc.describes_cozy,
                         normalize = "index")
ount_neigh.columns = ["no_cozy", "cozy"]
ount_neigh.sort_values(by = "cozy", ascending = False)
```

```
    43768
     5127
ame: describes_cozy, dtype: int64
```

neighbourhood_group	no_cozy	cozy
Queens	0.861631	0.138369
Staten Island	0.873995	0.126005
Bronx	0.884510	0.115490
Brooklyn	0.897632	0.102368
Manhattan	0.902498	0.097502

Break for first part of activity in 02_preprocess_LDA

Outline

- Computational text analysis: why, and where from?
- Computational text analysis: how? Two categories:
  - Text mining/supervised: how can we search for (1) things we know in advance we want to look for and can (2) easily operationalize/define?
    - Intro or review or basic `Python` data structures and control flow
    - Workhorse tools for basic text mining: `re` (regular expressions); pandas str operations
  - **Unsupervised: how can we more inductively discover themes/patterns in texts?**
    - Preprocessing to prepare for a topic model: overview
    - Preprocessing: mechanics with nltk in Python
    - Topic modeling: concepts
    - Topic modeling: extension to structural topic model

Text mining of Airbnb listings versus topic modeling

- Suppose we were interested in looking at relationship between (1) what words people use to describe their airbnb listing and (2) neighborhood change (e.g., rapid demographic change, as measured through changes in ethnicity/income of those residing in the neighborhood)

Text mining of Airbnb listings versus topic modeling

- Suppose we were interested in looking at relationship between (1) what words people use to describe their airbnb listing and (2) neighborhood change (e.g., rapid demographic change, as measured through changes in ethnicity/income of those residing in the neighborhood)
- **Text mining approach**: build a dictionary of words or phrases we think signal gentrifying (*cute*; *safe*; *near cold brew*) and look at correlation with neighborhood change

Text mining of Airbnb listings versus topic modeling

- Suppose we were interested in looking at relationship between (1) what words people use to describe their airbnb listing and (2) neighborhood change (e.g., rapid demographic change, as measured through changes in ethnicity/income of those residing in the neighborhood)
- **Text mining approach**: build a dictionary of words or phrases we think signal gentrifying (*cute*; *safe*; *near cold brew*) and look at correlation with neighborhood change
- Drawbacks:

Text mining of Airbnb listings versus topic modeling

- Suppose we were interested in looking at relationship between (1) what words people use to describe their airbnb listing and (2) neighborhood change (e.g., rapid demographic change, as measured through changes in ethnicity/income of those residing in the neighborhood)
- **Text mining approach**: build a dictionary of words or phrases we think signal gentrifying (*cute*; *safe*; *near cold brew*) and look at correlation with neighborhood change
- Drawbacks:
  - Might be difficult to know in advance which words to include

Text mining of Airbnb listings versus topic modeling

- Suppose we were interested in looking at relationship between (1) what words people use to describe their airbnb listing and (2) neighborhood change (e.g., rapid demographic change, as measured through changes in ethnicity/income of those residing in the neighborhood)
- **Text mining approach**: build a dictionary of words or phrases we think signal gentrifying (*cute*; *safe*; *near cold brew*) and look at correlation with neighborhood change
- Drawbacks:
  - Might be difficult to know in advance which words to include
  - Lack of surprise: what if there's a pattern in the listings correlated with demographic change, but that we didn't anticipate?

Text mining of Airbnb listings versus topic modeling

- Suppose we were interested in looking at relationship between (1) what words people use to describe their airbnb listing and (2) neighborhood change (e.g., rapid demographic change, as measured through changes in ethnicity/income of those residing in the neighborhood)
- **Text mining approach**: build a dictionary of words or phrases we think signal gentrifying (*cute*; *safe*; *near cold brew*) and look at correlation with neighborhood change
- Drawbacks:
  - Might be difficult to know in advance which words to include
  - Lack of surprise: what if there's a pattern in the listings correlated with demographic change, but that we didn't anticipate?
- Therefore, rather than search for specific words or phrases, begin with *full text* of the document

Represent document as a bag of words

- Represent each document as a "bag of words", where order doesn't matter

Represent document as a bag of words

- Represent each document as a "bag of words", where order doesn't matter
- Examples:

```
['clean', '&', 'quiet', 'apt', 'home', 'by', 'the', 'park']
['skylit', 'midtown', 'castle']
['the', 'village', 'of', 'harlem', '....', 'new', 'york', '!']
['cozy', 'entire', 'floor', 'of', 'brownstone']
```

Represent document as a bag of words

- Represent each document as a "bag of words", where order doesn't matter
- Examples:

```
['clean', '&', 'quiet', 'apt', 'home', 'by', 'the', 'park']
['skylit', 'midtown', 'castle']
['the', 'village', 'of', 'harlem', '....', 'new', 'york', '!']
['cozy', 'entire', 'floor', 'of', 'brownstone']
```

- Notice that it contains a lot of extraneous information

Preprocess

"clean quiet apt home by the park"

"clean quiet apt home by the park"

Compound Words: With substantive justification, words can be combined or split to improve inference.

Preprocess

"clean quiet apt home by the park"

Compound Words: An analyst may want to combine words into a single term that can be analyzed.

Preprocess

"clean quiet apt home by the park"

Compound Words: An analyst may want to combine words into a single term that can be analyzed.

Preprocess

[clean], [] [quiet apt], [home], [by], [the], [park]

Compound Words: An analyst may want to combine words into a single term that can be analyzed.

[clean], [] [quiet apt], [home], [by], [the], [park]

Stopword Removal: Removing terms that are not related to what the author is studying from the text.

Preprocess

[clean], [], [quiet apt], [home], [by], [the], [park]

Stopword Removal: Removing terms that are not related to what the author is studying from the text.

Preprocess

[clean], [quiet apt], [home], [park]

Stopword Removal: Removing terms that are not related to what the author is studying from the text.

[clean], [quiet apt], [home], [park]

Stemming: Takes the ends off conjugated verbs or plural nouns, leaving just the “stem.”

Preprocess

[clean], [quiet apt], [home], [park]

Stemming: Takes the ends off conjugated verbs or plural nouns, leaving just the “stem.”

Preprocess

[clean], [quiet apt], [hom], [park]

Stemming: Takes the ends off conjugated verbs or plural nouns, leaving just the “stem.”

Repeat with each document and then represent as document-term matrix

doc	1br	apartment	apt	area	backyard	bdrm …
1	0	0	1	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	1	0	0	0
⋮						

Outline

- Computational text analysis: why, and where from?
- Computational text analysis: how? Two categories:
  - Text mining/supervised: how can we search for (1) things we know in advance we want to look for and can (2) easily operationalize/define?
    - Intro or review or basic `Python` data structures and control flow
    - Workhorse tools for basic text mining: `re` (regular expressions); pandas str operations
  - Unsupervised: how can we more inductively discover themes/patterns in texts?
    - Preprocessing to prepare for a topic model: overview
    - **Preprocessing: mechanics with nltk in Python**
    - Topic modeling: concepts
    - Topic modeling: extension to structural topic model

Step one: create stopword list to filter out

Why do this early? Especially if you want to create your own list of stopwords for your context—for instance, in the airbnb reviews, you might want to remove the words apartment or apt—it's easier to do that before stemming

```
## call the specific module
from nltk.corpus import stopwords

## call a specific set of stopwords from package
list_stopwords = stopwords.words('english')

## augment with your own
list_stopwords_new = list_stopwords + ['apartment', 'apt']
```

Step two: convert words to lowercase and filter out stopwords

```
## example with one string
one_listing = airbnb_nyc.listing_lower[5]
one_listing
```

```
'large cozy 1 br apartment in midtown east'
```

```
## tokenize
one_listing_tokenized = wordpunct_tokenize(one_listing)
one_listing_tokenized
```

```
['large', 'cozy', '1', 'br', 'apartment', 'in', 'midtown', 'east']
```

```
## filter out stopwords
one_listing_nostop = [token for token in
                      one_listing_tokenized if
                      token not in list_stopwords_new]
one_listing_nostop
```

```
['large', 'cozy', '1', 'br', 'midtown', 'east']
```

```
## recombine into string for further processing
one_listing_nostop_string = " ".join(one_listing_nostop)
one_listing_nostop_string
```

```
'large cozy 1 br midtown east'
```

And so on with other preprocessing steps....

```
## filter out stopwords
one_listing_nostop = [token for token in
                      one_listing_tokenized if
                      token not in list_stopwords_new]
one_listing_nostop
```

```
['large', 'cozy', '1', 'br', 'midtown', 'east']
```

```
from nltk.stem.porter import PorterStemmer
porter = PorterStemmer()
onelisting_nostop_nopunct_stemmed_onlywords = [porter.stem(token)
                            for token in one_listing_nostop
                            if token.isalpha() and
                            len(token) > 2]
onelisting_nostop_nopunct_stemmed_onlywords
```

```
['larg', 'cozi', 'midtown', 'east']
```

Repeat over all documents, and combine into a document-term matrix

- **More manual way**: basically, need to find union of all words; can do it by (1) creating an empty dictionary; (2) looping over the documents; (3) when a document contains a new term, it gets added to dictionary as a key; (4) when a document contains a term already in the dictionary, we start counting how many times the term appears in the doc

Repeat over all documents, and combine into a document-term matrix

- **More manual way**: basically, need to find union of all words; can do it by (1) creating an empty dictionary; (2) looping over the documents; (3) when a document contains a new term, it gets added to dictionary as a key; (4) when a document contains a term already in the dictionary, we start counting how many times the term appears in the doc
- **More automatic way**: uses `sklearn` function; code is in the `create_dtm` helper function in your activity

Extensions thus far to topics I'm not covering

- Sentiment analysis. Two approaches:

Extensions thus far to topics I'm not covering

- Sentiment analysis. Two approaches:
    1. Dictionary-based method: `vader lexicon` and other imports within `nltk` have options for scoring a string along different sentiment dimensions; should think carefully about preprocessing (e.g., we removed punctuation but that might be very relevant!)

Extensions thus far to topics I'm not covering

- Sentiment analysis. Two approaches:
  1. Dictionary-based method: `vader lexicon` and other imports within `nltk` have options for scoring a string along different sentiment dimensions; should think carefully about preprocessing (e.g., we removed punctuation but that might be very relevant!)
  2. Classification methods: have training set of documents labeled as positive/negative; use document-term matrix (filtered based on metrics like tf-idf to reduce dimensionality) to build model to predict those labels

Extensions thus far to topics I'm not covering

- Sentiment analysis. Two approaches:
  1. Dictionary-based method: `vader lexicon` and other imports within `nltk` have options for scoring a string along different sentiment dimensions; should think carefully about preprocessing (e.g., we removed punctuation but that might be very relevant!)
  2. Classification methods: have training set of documents labeled as positive/negative; use document-term matrix (filtered based on metrics like tf-idf to reduce dimensionality) to build model to predict those labels
- N-grams: similar setup but modify `create_dtm` to use bigrams.

Extensions thus far to topics I'm not covering

- Sentiment analysis. Two approaches:
  1. Dictionary-based method: `vader lexicon` and other imports within `nltk` have options for scoring a string along different sentiment dimensions; should think carefully about preprocessing (e.g., we removed punctuation but that might be very relevant!)
  2. Classification methods: have training set of documents labeled as positive/negative; use document-term matrix (filtered based on metrics like tf-idf to reduce dimensionality) to build model to predict those labels
- N-grams: similar setup but modify `create_dtm` to use bigrams.
- Word embeddings: bag of words treats all text in document as an unordered collection. But we might think the context of the word (e.g., cozy brownstone versus spacious brownstone) matters, and we want to have more flexible windows than n-grams. Basic idea behind embeddings is to (1) predict a focal word (e.g., "cozy"); (2) predict its context within some window (e.g., "brownstone", "tiny", "cramped"). Each word is then represented as a vector, and vectors can be related to each other—e.g., "doctor:nurse"; "female:male." Good intro here.

Extensions thus far to topics I'm not covering

- Sentiment analysis. Two approaches:
  1. Dictionary-based method: `vader lexicon` and other imports within `nltk` have options for scoring a string along different sentiment dimensions; should think carefully about preprocessing (e.g., we removed punctuation but that might be very relevant!)
  2. Classification methods: have training set of documents labeled as positive/negative; use document-term matrix (filtered based on metrics like tf-idf to reduce dimensionality) to build model to predict those labels
- N-grams: similar setup but modify `create_dtm` to use bigrams.
- Word embeddings: bag of words treats all text in document as an unordered collection. But we might think the context of the word (e.g., cozy brownstone versus spacious brownstone) matters, and we want to have more flexible windows than n-grams. Basic idea behind embeddings is to (1) predict a focal word (e.g., "cozy"); (2) predict its context within some window (e.g., "brownstone", "tiny", "cramped"). Each word is then represented as a vector, and vectors can be related to each other—e.g., "doctor:nurse"; "female:male." Good intro here.
  - For description: can look at relationships between words over time

Extensions thus far to topics I'm not covering

- Sentiment analysis. Two approaches:
  1. Dictionary-based method: `vader lexicon` and other imports within `nltk` have options for scoring a string along different sentiment dimensions; should think carefully about preprocessing (e.g., we removed punctuation but that might be very relevant!)
  2. Classification methods: have training set of documents labeled as positive/negative; use document-term matrix (filtered based on metrics like tf-idf to reduce dimensionality) to build model to predict those labels
- N-grams: similar setup but modify `create_dtm` to use bigrams.
- Word embeddings: bag of words treats all text in document as an unordered collection. But we might think the context of the word (e.g., cozy brownstone versus spacious brownstone) matters, and we want to have more flexible windows than n-grams. Basic idea behind embeddings is to (1) predict a focal word (e.g., "cozy"); (2) predict its context within some window (e.g., "brownstone", "tiny", "cramped"). Each word is then represented as a vector, and vectors can be related to each other—e.g., "doctor:nurse"; "female:male." Good intro here.
  - For description: can look at relationships between words over time
  - Classification: if using words to predict some outcome (e.g., respectful

Outline

- Computational text analysis: why, and where from?
- Computational text analysis: how? Two categories:
  - Text mining/supervised: how can we search for (1) things we know in advance we want to look for and can (2) easily operationalize/define?
    - Intro or review or basic `Python` data structures and control flow
    - Workhorse tools for basic text mining: `re` (regular expressions); pandas str operations
  - **Unsupervised: how can we more inductively discover themes/patterns in texts?**
    - Preprocessing to prepare for a topic model: overview
    - Preprocessing: mechanics with nltk in Python
    - **Topic modeling: concepts**
    - Workhorse tool for topic modeling in `R`: `STM` package

Latent Dirichlet Allocation

- Idea: documents exhibit each topic in some proportion. This is an admixture.

Latent Dirichlet Allocation

- Idea: documents exhibit each topic in some proportion. This is an admixture.
- Each document is a mixture over topics. Each topic is a mixture over words.

Latent Dirichlet Allocation

- Idea: documents exhibit each topic in some proportion. This is an admixture.
- Each document is a mixture over topics. Each topic is a mixture over words.
- Latent Dirichlet Allocation estimates:

Latent Dirichlet Allocation

- Idea: documents exhibit each topic in some proportion. This is an admixture.
- Each document is a mixture over topics. Each topic is a mixture over words.

- Latent Dirichlet Allocation estimates:
  - The distribution over words for each topic.

Latent Dirichlet Allocation

- Idea: documents exhibit each topic in some proportion. This is an admixture.
- Each document is a mixture over topics. Each topic is a mixture over words.
- Latent Dirichlet Allocation estimates:
  - The distribution over words for each topic.
  - The proportion of a document in each topic, for each document.

Latent Dirichlet Allocation

- Idea: documents exhibit each topic in some proportion. This is an admixture.
- Each document is a mixture over topics. Each topic is a mixture over words.

- Latent Dirichlet Allocation estimates:
  - The distribution over words for each topic.
  - The proportion of a document in each topic, for each document.

Latent Dirichlet Allocation

- Idea: documents exhibit each topic in some proportion. This is an admixture.
- Each document is a mixture over topics. Each topic is a mixture over words.

- Latent Dirichlet Allocation estimates:
  - The distribution over words for each topic.
  - The proportion of a document in each topic, for each document.

Maintained assumptions: Bag of words/fix number of topics ex ante.

From: Stewart, LDA

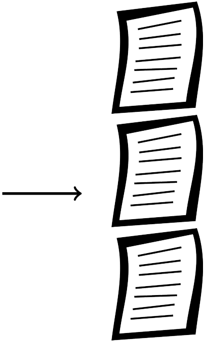
What this means in pictures

Say you have
a lot of people.

What this means in pictures

Say you have
a lot of people.

Each writes
some texts



What this means in pictures

Say you have
a lot of people.

$\longrightarrow$

Each writes
some texts

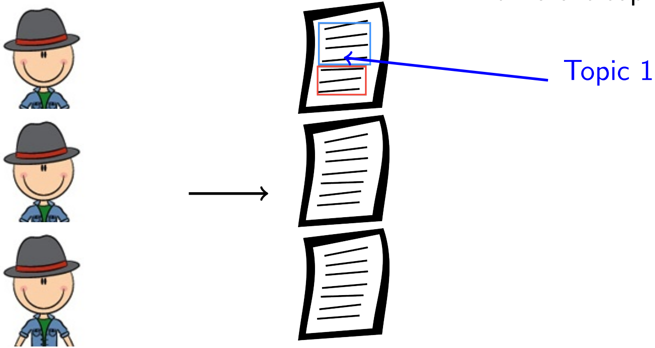
That discuss a few
different topics

What this means in pictures

Say you have
a lot of people.

Each writes
some texts

That discuss a few
different topics

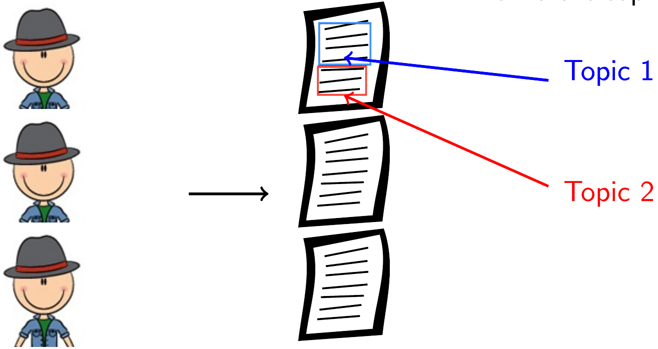


What this means in pictures

Say you have
a lot of people.

Each writes
some texts

That discuss a few
different topics

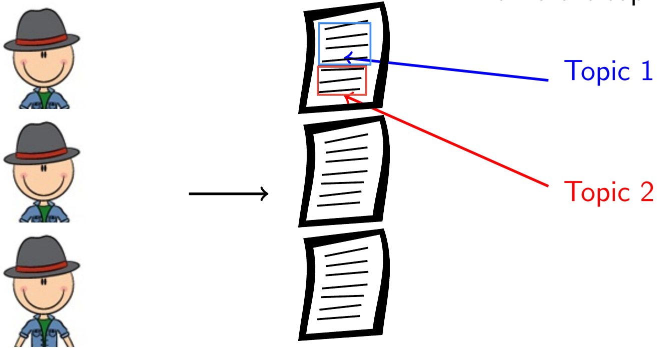


What this means in pictures

Say you have
a lot of people.

Each writes
some texts

That discuss a few
different topics



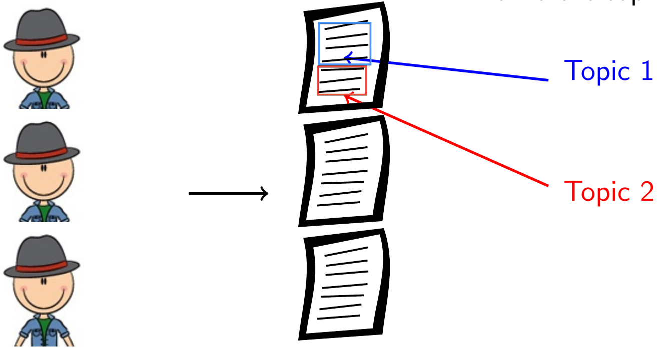
The Latent Dirichlet Allocation estimates:

What this means in pictures

Say you have a lot of people.

Each writes some texts

That discuss a few different topics



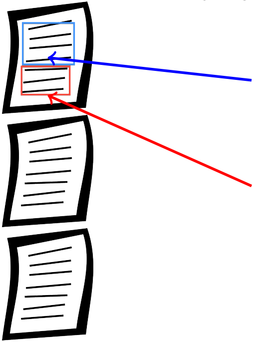
The Latent Dirichlet Allocation estimates:

1. The topics- each is a distribution over words

What this means in pictures

Say you have
a lot of people.

Each writes
some texts



That discuss a few
different topics

Politics

congress, nations, power, votes, agreement, bargaining

Statistics

estimator, data, analysis, variance, model, inference

The Latent Dirichlet Allocation estimates:

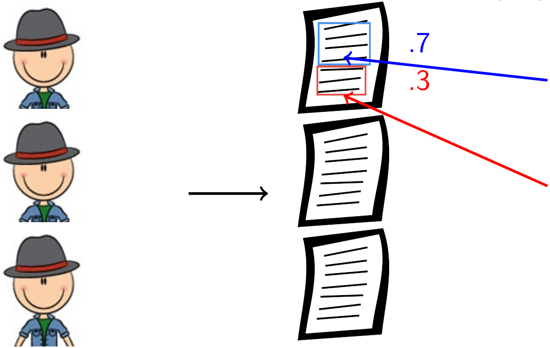
1. The topics- each is a distribution over words

What this means in pictures

Say you have a lot of people.

Each writes some texts

That discuss a few different topics



Politics

congress, nations, power, votes, agreement, bargaining

Statistics

estimator, data, analysis, variance, model, inference

The Latent Dirichlet Allocation estimates:

1. The topics- each is a distribution over words
2. The proportion of each document in each topic

Why does this work $\rightsquigarrow$ Co-occurrence

Where's the information for each word's topic?

Why does this work $\rightsquigarrow$ Co-occurrence

Where's the information for each word's topic?

Reconsider document-term matrix

Why does this work $\rightsquigarrow$ Co-occurrence

Where's the information for each word's topic?

Reconsider document-term matrix

	Word_1	Word_2	$\ldots$	Word_J
Doc_1	0	1	$\ldots$	0
Doc_2	2	0	$\ldots$	3
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
Doc_N	0	1	$\ldots$	1

Why does this work $\rightsquigarrow$ Co-occurrence

Where's the information for each word's topic?

Reconsider document-term matrix

	Word_1	Word_2	$\ldots$	Word_J
Doc_1	0	1	$\ldots$	0
Doc_2	2	0	$\ldots$	3
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
Doc_N	0	1	$\ldots$	1

We are learning the pattern of what words occur together.

Why does this work $\rightsquigarrow$ Co-occurrence

Where's the information for each word's topic?

Reconsider document-term matrix

	Word_1	Word_2	$\dots$	Word_J
Doc_1	0	1	$\dots$	0
Doc_2	2	0	$\dots$	3
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
Doc_N	0	1	$\dots$	1

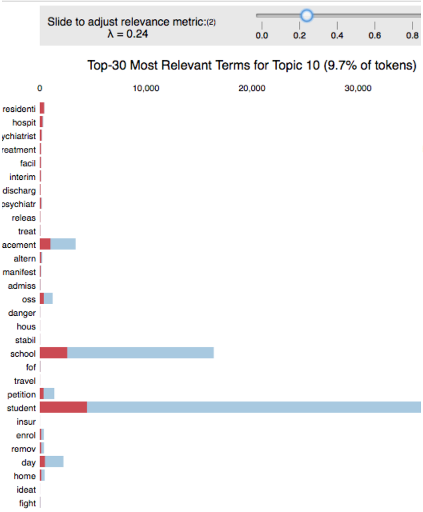
We are learning the pattern of what words occur together.

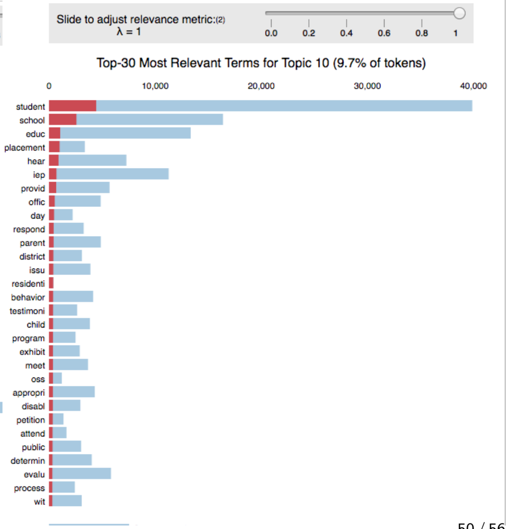
The model wants a topic to contain as few words as possible, but a document to contain as few topics as possible. This tension is what makes the model work.

From: Stewart, LDA

Break for second part of activity in 02_preprocess_LDA

In our case, the metric used to pull “top words” for a topic matters





Outline

- Computational text analysis: why, and where from?
- Computational text analysis: how? Two categories:
  - Text mining/supervised: how can we search for (1) things we know in advance we want to look for and can (2) easily operationalize/define?
    - Intro or review or basic `Python` data structures and control flow
    - Workhorse tools for basic text mining: `re` (regular expressions); pandas str operations
  - **Unsupervised: how can we more inductively discover themes/patterns in texts?**
    - Preprocessing to prepare for a topic model: overview
    - Preprocessing: mechanics with nltk in Python
    - Topic modeling: concepts
    - **Topic modeling: extension to structural topic model**

Motivation

- Previous task should have returned topics representing concepts like (1) institutionalization/psychiatric care, (2) speech issues, and more

Motivation

- Previous task should have returned topics representing concepts like (1) institutionalization/psychiatric care, (2) speech issues, and more
- What if we want to descriptively explore how attributes of a complaint–e.g., which parent was involved; when the complaint was filed– are related to these topics?

Motivation

- Previous task should have returned topics representing concepts like (1) institutionalization/psychiatric care, (2) speech issues, and more
- What if we want to descriptively explore how attributes of a complaint–e.g., which parent was involved; when the complaint was filed– are related to these topics?
- Various extensions of LDA to incorporate document "metadata" or covariates. Focus on one: structural topic models (STM)

STM = LDA + Contextual Information

STM = LDA + Contextual Information

- STM provides two ways to include contextual information

STM = LDA + Contextual Information

- STM provides two ways to include contextual information

STM = LDA + Contextual Information

- STM provides two ways to include contextual information
  - Topic prevalence can vary by metadata

STM = LDA + Contextual Information

- STM provides two ways to include contextual information
  - Topic prevalence can vary by metadata

STM = LDA + Contextual Information

- STM provides two ways to include contextual information
  - Topic prevalence can vary by metadata
    - e.g. city papers cover protests more than provincial papers

STM = LDA + Contextual Information

- STM provides two ways to include contextual information
  - Topic prevalence can vary by metadata
    - e.g. city papers cover protests more than provincial papers

STM = LDA + Contextual Information

- STM provides two ways to include contextual information
  - Topic prevalence can vary by metadata
    - e.g. city papers cover protests more than provincial papers
  - Topic content can vary by metadata

STM = LDA + Contextual Information

- STM provides two ways to include contextual information
  - Topic prevalence can vary by metadata
    - e.g. city papers cover protests more than provincial papers
  - Topic content can vary by metadata

STM = LDA + Contextual Information

- STM provides two ways to include contextual information
  - Topic prevalence can vary by metadata
    - e.g. city papers cover protests more than provincial papers
  - Topic content can vary by metadata
    - e.g. city papers talk about protests differently

STM = LDA + Contextual Information

- STM provides two ways to include contextual information
  - Topic prevalence can vary by metadata
    - e.g. city papers cover protests more than provincial papers
  - Topic content can vary by metadata
    - e.g. city papers talk about protests differently

STM = LDA + Contextual Information

- STM provides two ways to include contextual information
  - Topic prevalence can vary by metadata
    - e.g. city papers cover protests more than provincial papers
  - Topic content can vary by metadata
    - e.g. city papers talk about protests differently
- Including context improves the model:

STM = LDA + Contextual Information

- STM provides two ways to include contextual information
  - Topic prevalence can vary by metadata
    - e.g. city papers cover protests more than provincial papers
  - Topic content can vary by metadata
    - e.g. city papers talk about protests differently
- Including context improves the model:

STM = LDA + Contextual Information

- STM provides two ways to include contextual information
  - Topic prevalence can vary by metadata
    - e.g. city papers cover protests more than provincial papers
  - Topic content can vary by metadata
    - e.g. city papers talk about protests differently
- Including context improves the model:
  - more accurate estimation

STM = LDA + Contextual Information

- STM provides two ways to include contextual information
  - Topic prevalence can vary by metadata
    - e.g. city papers cover protests more than provincial papers
  - Topic content can vary by metadata
    - e.g. city papers talk about protests differently
- Including context improves the model:
  - more accurate estimation

STM = LDA + Contextual Information

- STM provides two ways to include contextual information
  - Topic prevalence can vary by metadata
    - e.g. city papers cover protests more than provincial papers
  - Topic content can vary by metadata
    - e.g. city papers talk about protests differently
- Including context improves the model:
  - more accurate estimation
  - better qualitative interpretability

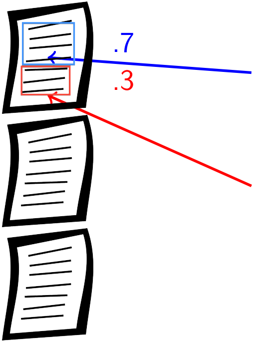
From: Stewart, LDA

STM: What this means in pictures

Say you have
a lot of people.

⟶

Each writes
some text



that discuss a few
different topics

Politics

congress, nations, power, votes, agreement, bargaining

Statistics

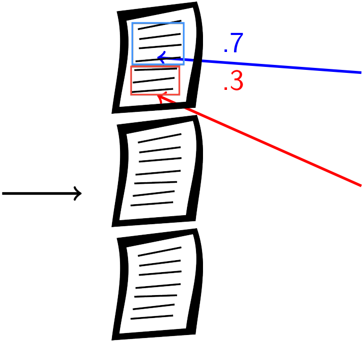
estimator, data, analysis, variance, model, inference

The STM Allows for:

STM: What this means in pictures

Say you have
a lot of people.

Each writes
some text



that discuss a few
different topics

Politics

congress, nations, power, votes, agreement, bargaining

Statistics

estimator, data, analysis, variance, model, inference

The STM Allows for:

1. The words in each topic to vary by gender

STM: What this means in pictures

Say you have a lot of people.

Each writes some text

that discuss a few different topics

Politics

congress, nations, power, votes, agreement, bargaining

Statistics

estimator, data, analysis, variance, model, inference

The STM Allows for:

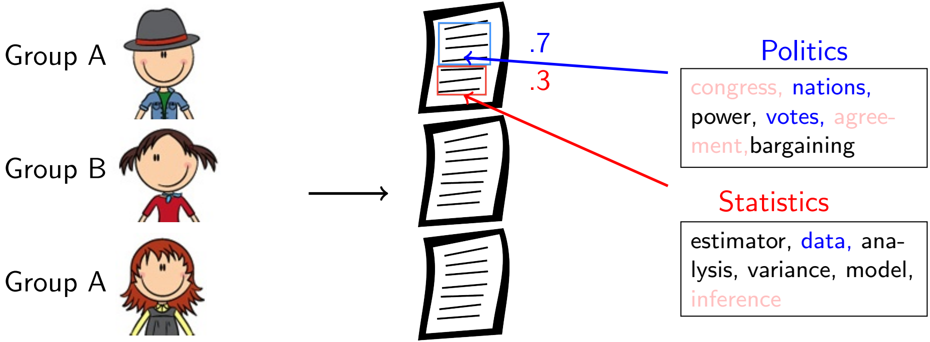
1. The words in each topic to vary by gender

STM: What this means in pictures

Say you have a lot of people.

Each writes some text

that discuss a few different topics



The STM Allows for:

1. The words in each topic to vary by gender
2. The topic proportions to vary by group

STM: What this means in pictures

Say you have
a lot of people.

Each writes
some text

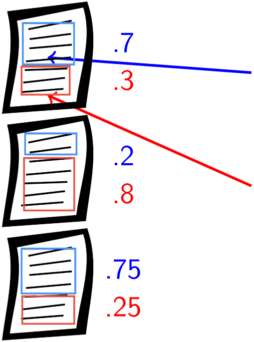
that discuss a few
different topics

Group A

Group B

Group A

→



Politics

congress, nations, power, votes, agreement, bargaining

Statistics

estimator, data, analysis, variance, model, inference

The STM Allows for:

1. The words in each topic to vary by gender
2. The topic proportions to vary by group

Break for activity in 03_estimateviz_STM

Thanks!

raj2@princeton.edu

scholar.princeton.edu/rebeccajohnson