Social Media Analytics Workshop Series with R

Text as Data (Part 3)

Ryan Wesslen
UNC Charlotte / Project Mosaic

July 27, 2017

Text as Data Paradigm

Dictionaries and why Sentiment Analysis is Hard

Supervised Machine Learning for Text Classification

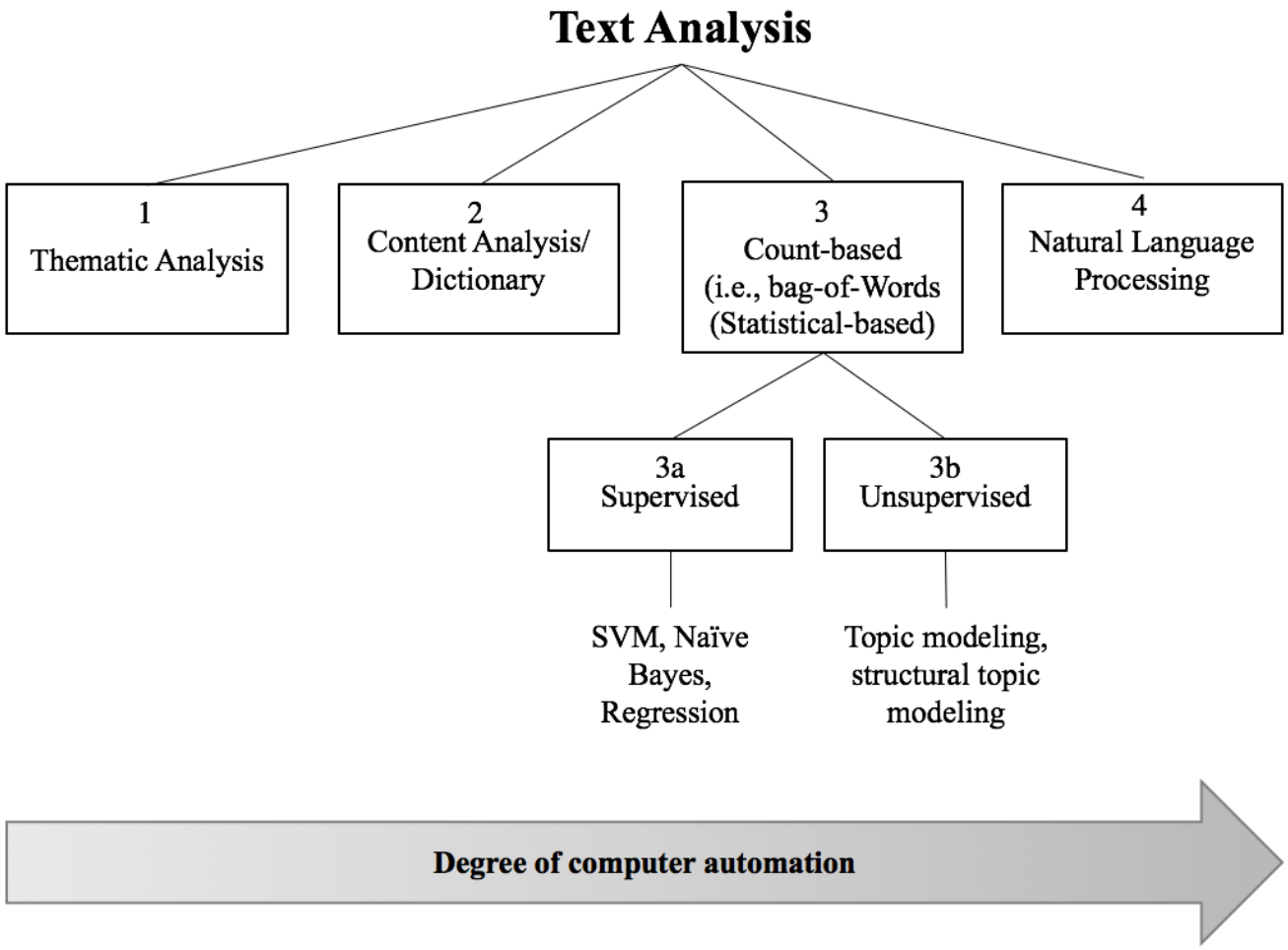
Text Analysis for the Real (Unsupervised) World: Topic Modeling

Text Pre-Processing: Why It's More Important Than You Think

To Infinity and Beyond: Word Embedding & Deep Learning

Text as Data Paradigm

Text Analysis Methods



Note. This categorization is an overgeneralization for illustrative purposes (e.g., omits word embedding, deep learning)

Credit: Haley Woznyj

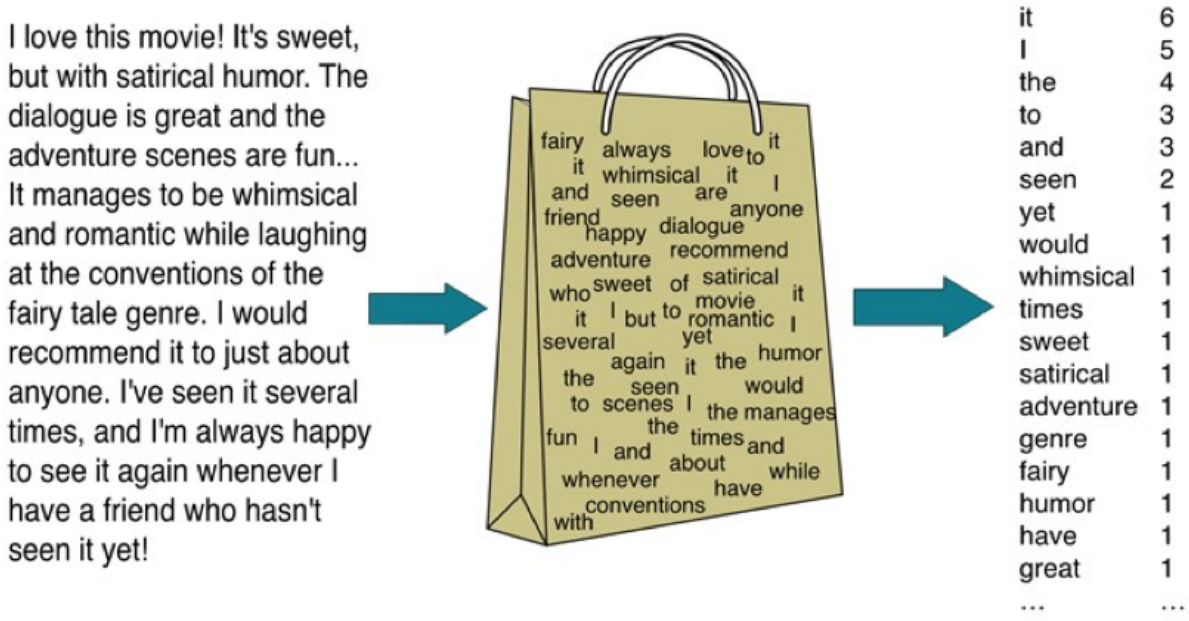
Question: How to quantify Text?

Problem of High Dimensionality

"A sample of 30-word Twitter messages that use only the 1,000 most common words in the English language, for example, has roughly *as many dimensions as there are atoms in the universe*."

Gentzkow, Kelly and Taddy (2017)

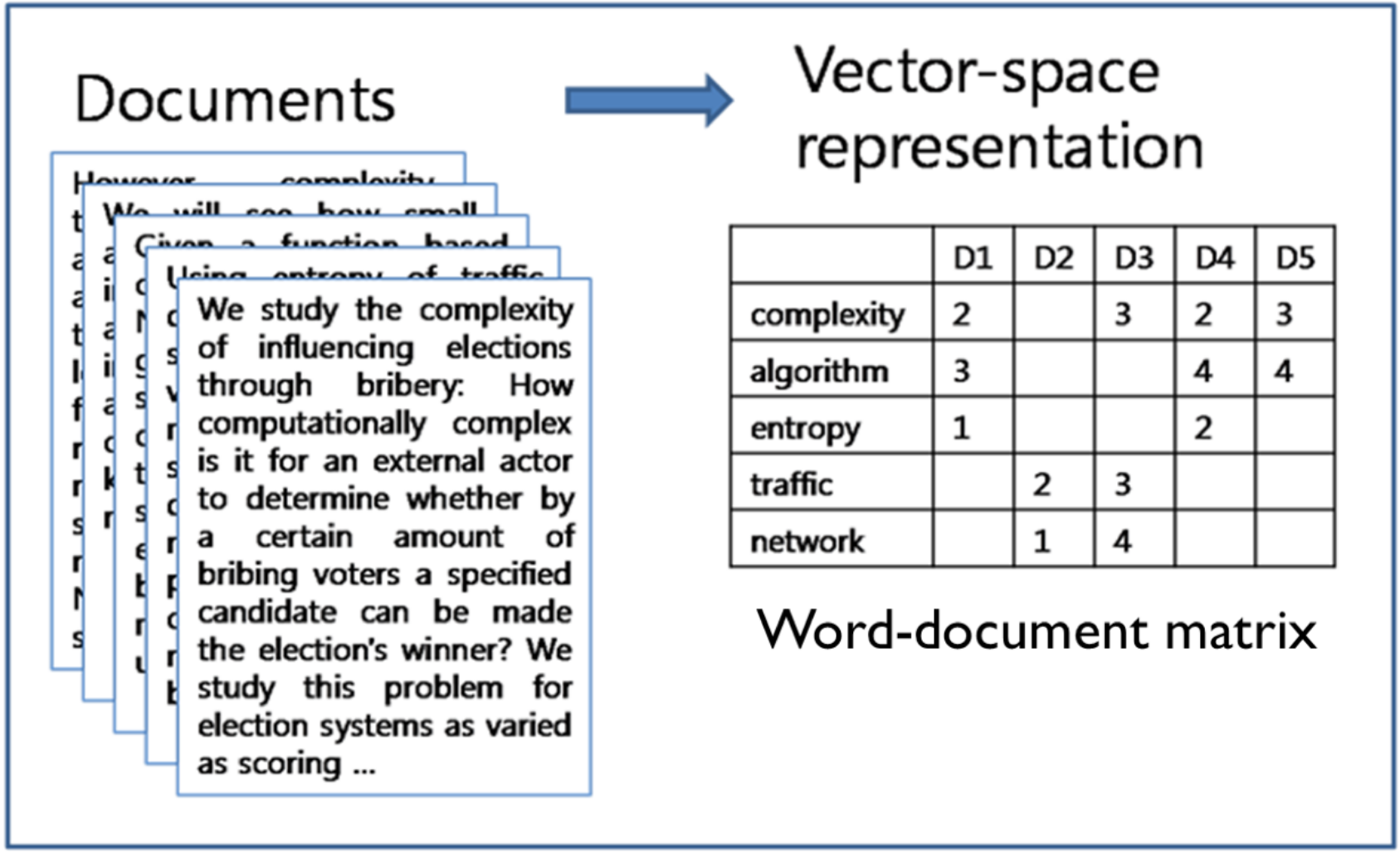
Bag of Words: Simplest Approach



Credit: Chris Manning

- Count the number of words in each document.
- Simplicity trade-off for correctness; ignores word order.
- Good at classification; poor at semantic meaning.

Document Term Matrix



	D1	D2	D3	D4	D5
complexity	2		3	2	3
algorithm	3			4	4
entropy	1			2	
traffic		2	3		
network		1	4		

Pre-processing

Raw text (strings) require pre-processing to convert the text into a pre-determined data structure.

After tokenization seven common pre-processing steps include:

* Remove Punctuation (P)
* Remove Numbers (N)
* Lowercasing (L)
* Stemming (S)
* Stopword Removal (W)
* n-Gram Inclusion (3)
* Remove Infrequent Words (I)

Citation	Steps	Cites
Slapin and Proksch (2008)	P-S-L-N-W	427
Grimmer (2010)	L-P-S-I-W	258
Quinn et al. (2010)	P-L-S-I	275
Grimmer and King (2011)	L-P-S-I	109
Roberts et al. (2014)	P-L-S-W	117

Table 1: Preprocessing steps taken/suggested in recent notable papers that deal with unsupervised learning methods. The cite total is taken from Google Scholar at the time of writing. In the case of Slapin and Proksch (2008), we consulted their Wordfish manual (version 1.3). In the case of Roberts et al. (2014), the authors suggest further steps might be appropriate for a given application.

Denny & Spirling, 2017

Text as Data Paradigm

Text as Data: The Promise and Pitfalls of Automatic Content Analysis Methods for Political Texts

Justin Grimmer
Department of Political Science, Stanford University, Encina Hall West 616 Serra Street, Stanford, CA 94305
e-mail: jgrimmer@stanford.edu (corresponding author)

Brandon M. Stewart
Department of Government and Institute for Quantitative Social Science, Harvard University, 1737 Cambridge Street, Cambridge, MA 02138
e-mail: bstewart@fas.harvard.edu

Edited by R. Michael Alvarez

Politics and political conflict often occur in the written and spoken word. Scholars have long recognized this, but the massive costs of analyzing even moderately sized collections of texts have hindered their use in political science research. Here lies the promise of automated text analysis: it substantially reduces the costs of analyzing large collections of text. We provide a guide to this exciting new area of research and show how, in many instances, the methods have already obtained part of their promise. But there are pitfalls to using automated methods—they are no substitute for careful thought and close reading and require extensive and problem-specific validation. We survey a wide range of new methods, provide guidance on how to validate the output of the models, and clarify misconceptions and errors in the literature. To conclude, we argue that for automated text methods to become a standard tool for political scientists, methodologists must contribute new methods and new methods of validation.

http://stanford.edu/~jgrimmer/tad2.pdf

Four Principles of Text as Data Methods

1. All quantitative models of language are wrong – but some are useful.
2. Quantitative methods for text *amplify* human abilities, **not replace them**.
3. There is no globally best method for text analysis.
4. Validate, validate, validate.

Grimmer and Stewart, 2013

Text as Data Methods

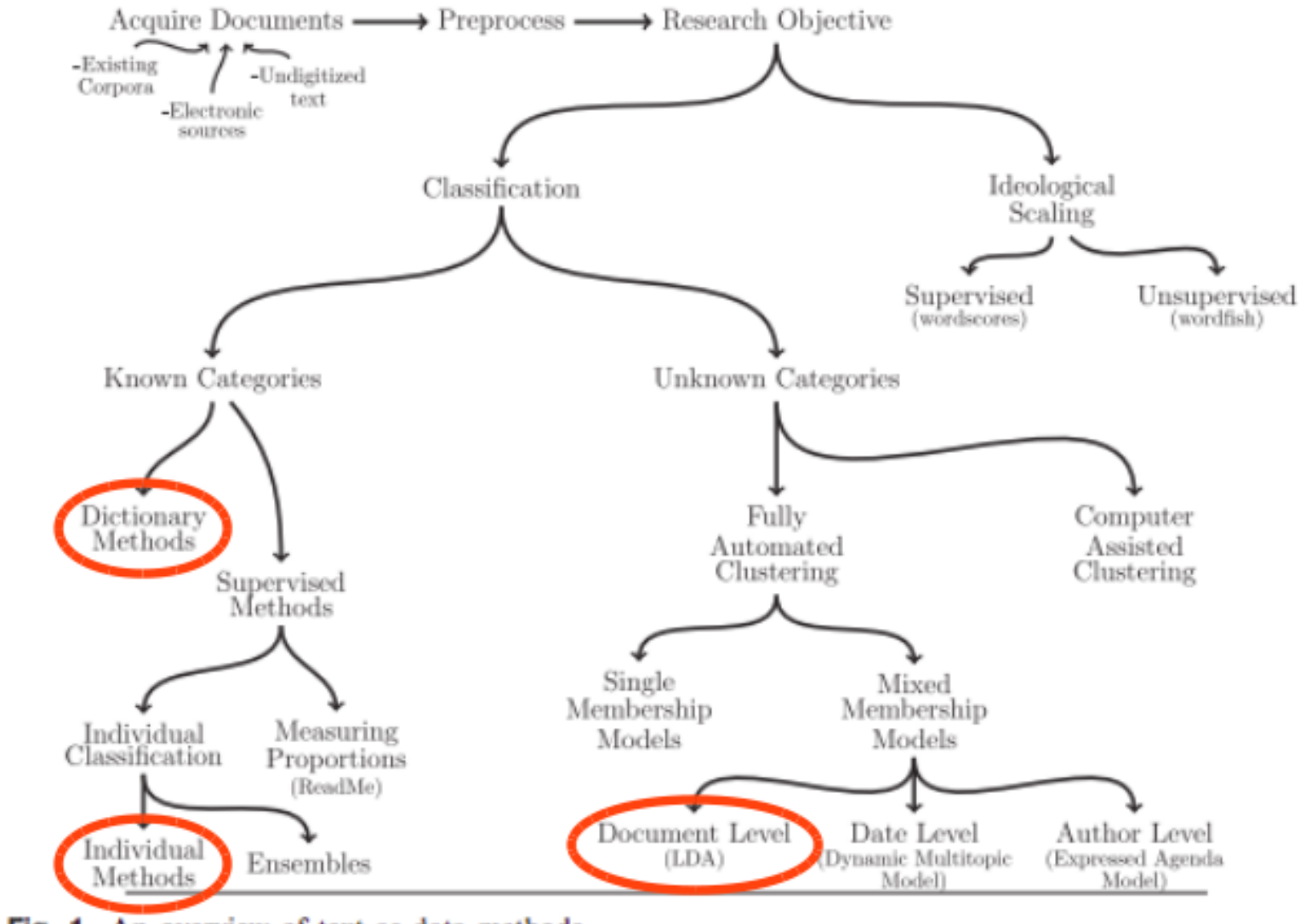


Fig. 1 An overview of text as data methods.

Cost/Benefits of Methods

TABLE 1 A Summary of Common Assumptions and Relative Costs Across Different Methods of Discrete Text Categorization

	Method				
A. Assumptions	***Reading***	***Human Coding***	***Dictionaries***	***Supervised Learning***	***Topic Model***
Categories are known	No	Yes	Yes	Yes	No
Category nesting, if any, is known	No	Yes	Yes	Yes	No
Relevant text features are known	No	No	Yes	Yes	Yes
Mapping is known	No	No	Yes	No	No
Coding can be automated	No	No	Yes	Yes	Yes
B. Costs					
Preanalysis Costs					
Person-hours spent conceptualizing	Low	High	High	High	Low
Level of substantive knowledge	Moderate/High	High	High	High	Low
Analysis Costs					
Person hours spent per text	High	High	Low	Low	Low
Level of substantive knowledge	Moderate/High	Moderate	Low	Low	Low
Postanalysis Costs					
Person-hours spent interpreting	High	Low	Low	Low	Moderate
Level of substantive knowledge	High	High	High	High	High

Quinn et al., 2010

Dictionaries and why Sentiment Analysis is Hard

Dictionary or Lexicon Based Analysis

Use a pre-defined list of words; requires knowing them in advance.

Lexicons can be theoretically-motivated (linguistics) or manually created (or both)

- Sentiment (Polarity): Hu and Liu, 2004
- Moral Foundations Created by Social Psychologists
- Biased Language: Recasens, Danescu-Niculescu-Mizil, and Jurafsky, 2013
- Subjectivity: Wilson, Weibe, and Hoffmann, 2005

Works ok for "aggregated" analysis (e.g., user-level)...

But works poorly on tweet/post-level.

Simple Sentiment Analysis: Load Lexicon

```
# loading lexicon of positive and negative words
lexicon <- readr::read_csv("../data/lexicon.csv")
pos.words <- lexicon$word[lexicon$polarity=="positive"]
neg.words <- lexicon$word[lexicon$polarity=="negative"]
```

```
# random sample of positive and negative words
sample(pos.words, 6)
```

```
## [1] "adventure"   "tranquility" "smilingly"   "pious"       "
## [6] "evocative"
```

```
sample(neg.words, 6)
```

```
## [1] "anti-proliferation" "heathen"                "scandals"
## [4] "massacre"           "mistrust"               "contrived"
```

Simple Sentiment Analysis: Load Lexicon & Score

```
tweets <- readr::read_csv("../data/pres_tweets.csv")

library(quanteda) #install.packages("quanteda")
twcorpus <- corpus(tweets$body)

# first we construct a dictionary object
mydict <- dictionary(list(negative = neg.words,
                          positive = pos.words))
# apply it to our corpus
sent <- dfm(twcorpus, dictionary = mydict)
# and add it as a new variable
tweets$score <- as.numeric(sent[,2]) - as.numeric(sent[,1])
```

Simple Sentiment Analysis: Most Positive Tweets

```
sample <- tweets[order(tweets$score, decreasing = T),]

cat(as.character(sample[1,"displayName"]))
```

```
## Ted Cruz
```

```
strwrap(as.character(sample[1,"body"]), width=50)
```

```
## [1] "We will restore our spirit. We will free our"
## [2] "minds &amp;amp; imagination. We will create a better"
## [3] "world. We will bring back jobs, freedom &amp;amp;"
## [4] "security"
```

```
cat(as.character(sample[1,"score"]))
```

```
## 9
```

Simple Sentiment Analysis: Most Negative Tweets

```
sample <- tweets[order(tweets$score, decreasing = F),]

cat(as.character(sample[1,"displayName"]))
```

```
## Ted Cruz
```

```
strwrap(as.character(sample[1,"body"]), width=50)
```

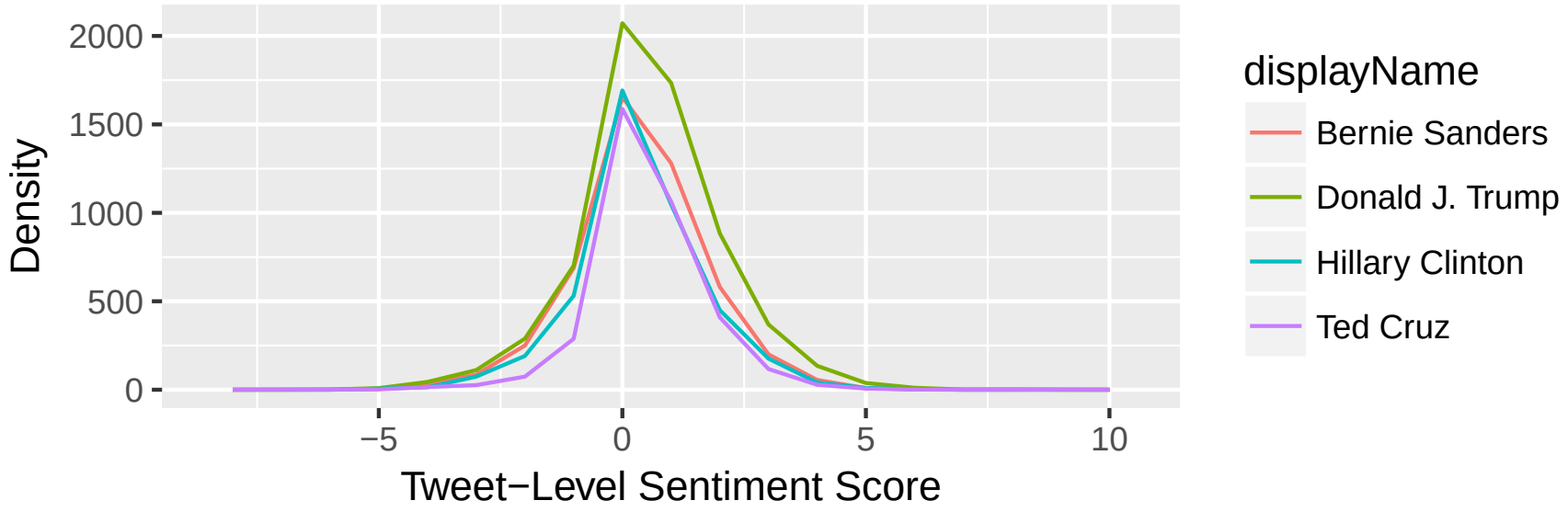
```
## [1] "We can't defeat radical Islamic terrorism as long"
## [2] "as we have a Commander-in-Chief unwilling to"
## [3] "utter the words \"radical Islamic terrorism.\""
```

```
cat(as.character(sample[1,"score"]))
```

```
## -7
```

Simple Sentiment Analysis: Distribution

```
library(ggplot2)
ggplot(tweets, aes(x = score, color = displayName)) +
  geom_freqpoly(binwidth = 1) +
  xlab("Tweet-Level Sentiment Score") +
  ylab("Density") +
  theme(legend.position="right")
```



Dictionaries: BEWARE!

← Information theory stuff

Memorizing small tables →

Be careful with dictionary-based text analysis

Posted on October 5, 2011

OK, everyone loves to run dictionary methods for sentiment and other text analysis — counting words from a predefined lexicon in a big corpus, in order to explore or test hypotheses about the corpus. In particular, this is often done for sentiment analysis: count positive and negative words (according to a sentiment polarity lexicon, which was derived from human raters or previous researchers' intuitions), and then proclaim the output yields sentiment levels of the documents. More and more papers come out every day that do this. I've done this myself. It's interesting and fun, but it's easy to get a bunch of meaningless numbers if you don't carefully validate what's going on. There are certainly good studies in this area that do further validation and analysis, but it's hard to trust a study that just presents a graph with a few overly strong speculative claims as to its meaning. This happens more than it ought to.

I was happy to see a similarly critical view in a nice working paper by Justin Grimmer and Brandon Stewart, Text as Data: The Promise and Pitfalls of Automatic Content Analysis Methods for Political Texts.

`https://brenocon.com/blog/2011/10/be-careful-with-dictionary-based-text-analysis/`

Ways to Improve Dictionary-Based Methods

- Validate: Choose x% and manually review for validity
- Customize Lexicons by adding/removing words
- Use to supplement or feature for other text methods

Supervised Machine Learning for Text Classification

Types of Machine Learning Algorithms

Supervised: know y and x

- Uses: Regression, Classification

Unsupervised: only know x (y is not known)

- Uses: Clustering, Dimensionality Reduction

Neural Networks

Reinforcement Learning

Supervised Learning: Text Classification Problem

- Classification models predict class labels (class labels = categories)
  - binary (yes or no)
  - ordinal (high, medium, low)
  - nominal (dog, cat, kangaroo)
- Classification models use supervised algorithms as the class labels ("y variables") are **known** (observed).
- There are many different models (algorithms) that can be used for classification problems.
- Examples: Naïve Bayes, Decision Tree, Support Vector Machine, Logistic Regression

Ridge Regression

- Ridge regression is based on regression framework but with regularization parameters.
- Regularization = add in penalty for the number of X variables
- Regularization is performed to reduce overfitting given the large number of X variables (words).

StackExchange on why Ridge works well for Text Classification

StackExchange on Ridge Regression Intuition

Ridge Regression

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

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

Problem:

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

Source: Grimmer, 2014, "Text as Data" course week 15

Ridge 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
- $\lambda \rightsquigarrow$ penalty parameter

Source: Grimmer, 2014, "Text as Data" course week 15

Ridge Regression Pro & Con

Pro:

- Prevents overfitting with many features (x variables)
- Interpretable coefficients (compared to Naïve Bayes or SVM)

Con:

- Decision of the lambda value (need of cross-validation)
- Abandons unbiased estimator

Patrick Breheny's lecture notes on Ridge Regression

Example: Predicting Trump vs. Cruz

Let's use a dataset of tweets between June 2015-June 2016 of Presidential Candidates tweets.

See Day 3 example. Code originally from Pablo Barbera.

```
library(tidyverse)
tweets <- read_csv("../data/pres_tweets.csv")

candidates <- c("Donald J. Trump","Ted Cruz")
tweets <- filter(tweets, displayName %in% candidates)

# create label: Trump = 0, Cruz = 1
trump <- "Donald J. Trump"
tweets$trump <- ifelse(tweets$displayName==trump, 0, 1)

# remove handles to avoid overfitting
tweets$body <- gsub('@[0-9_A-Za-z]+', '@', tweets$body)
```

quanteda preprocessing

```
library(quanteda)
twcorpus <- corpus(tweets$body)

# additional stop words
extraStop <- c("t.co", "https", "rt",
               "amp", "http", "t.c", "can")

twdfm <- dfm(twcorpus,
             remove=c(stopwords("english"), extraStop),
             remove_numbers = TRUE,
             remove_symbols = TRUE,
             remove_url = TRUE
             )

# remove sparse terms (trim)
twdfm <- dfm_trim(twdfm, min_count = 10)
```

Create Training vs Test Datasets

We'll separate the data into:

- training (80% of data)
- test (20% of data)

```
set.seed(123)
training <- sample(1:nrow(tweets),
                   floor(.80 * nrow(tweets)))

test <- (1:nrow(tweets))[1:nrow(tweets) %in% training == F]
```

Run 5-fold CV Ridge Regression

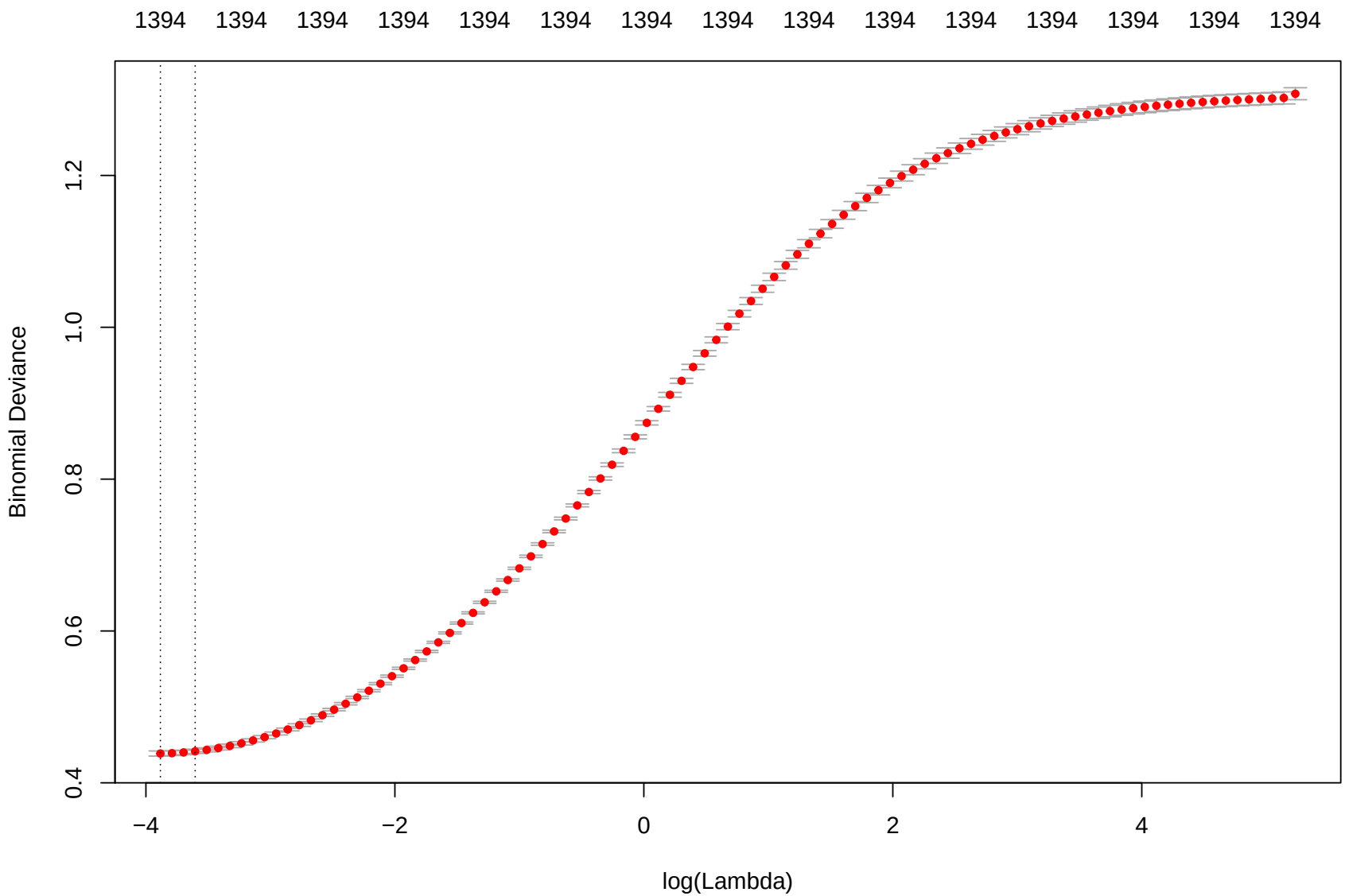
```
library(glmnet)
require(doMC)

registerDoMC(cores=3) # assumes have at least 4 cores

ridge <- cv.glmnet(twdfm[training,],
                   tweets$trump[training],
                   family="binomial",
                   alpha=0,
                   nfolds=5,
                   parallel=TRUE,
                   type.measure="deviance")
```

Plot Lambda Value

```
plot(ridge)
```



Make Predictions for the Test Dataset

Let's use the percent Trump/Cruz tweets from the training dataset as the threshold.

```
# 36.0439%
thres <- mean(tweets$trump[training])

preds <- predict(ridge,
                 twdfm[test,],
                 type="response") > thres
```

Model Accuracy on Test Data

```
# confusion matrix
table(preds, tweets$trump[test])
```

```
##
## preds       0    1
##   FALSE 1179   53
##   TRUE    95  678
```

```
accuracy <- accuracy(preds, tweets$trump[test])
precision <- precision(preds, tweets$trump[test])
recall <- recall(preds, tweets$trump[test])

knitr::kable(data.frame(accuracy, precision, recall))
```

accuracy	precision	recall
0.9261845	0.8771022	0.9274966

Explore why: Find most predictive words

To do this, we need to find the best lambda value (model).

```
best.lambda <- which(ridge$lambda==ridge$lambda.min)
beta <- ridge$glmnet.fit$beta[,best.lambda]
```

And save these coefficients.

```
## identifying predictive features
df <- data.frame(coef = as.numeric(beta),
                 word = names(beta), stringsAsFactors=F)
```

Trump's most predictive words

```
df <- df[order(df$coef),]
head(df[,c("coef", "word")], n=10)
```

```
##             coef                                word
## 812  -2.946053                       entrepreneurs
## 1119 -2.780045                             stopped
## 1291 -2.431947 #trump2016#makeamericagreatagain
## 1195 -2.422461                          #wiprimary
## 1265 -2.413748                            register
## 1289 -2.365488                             muslims
## 13   -2.244013          #makeamericagreatagain
## 1124 -2.224513                                 son
## 1365 -2.211081                           louisiana
## 1099 -2.162391                            desperate
```

Cruz's most predictive words

```
df <- df[order(df$coef, decreasing=TRUE),]
head(df[,c("coef", "word")], n=10)
```

```
##           coef            word
## 932  2.833202           admin
## 54   2.791958    #choosecruz
## 749  2.789096           heidi
## 1378 2.740852          prayer
## 1287 2.705626     #cruzcrowd
## 374  2.696764 #atimefortruth
## 1329 2.670115           human
## 129  2.666027 #caucusforcruz
## 1343 2.497244          ensure
## 979  2.492870         houston
```

Text Analysis for the Real (Unsupervised) World: Topic Modeling

Supervised Learning

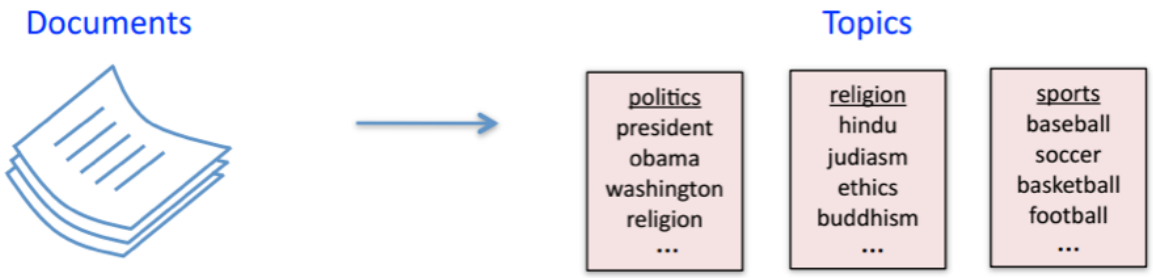
Credit: Matthew Denny

Unsupervised Learning

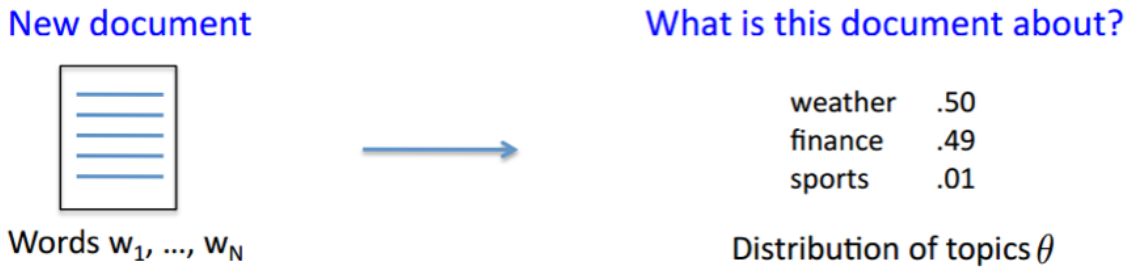
Credit: Matthew Denny

Topic Modeling

- **Topic models** are powerful tools for exploring large data sets and for making inferences about the content of documents



- Many applications in information retrieval, document summarization, and classification



- LDA is one of the simplest and most widely used topic models

Source: Pablo Barbera

Topic Modeling in R

```
tweets <- read_csv('../data/CharlotteTweets20Sample.csv')
source('../day3/functions.R')

library(quanteda)
twcorpus <- corpus(tweets$body)
docvars(twcorpus, "actor.id") <- as.character(tweets$actor.id)

stopWords <- c("t.co", "https", "rt", "amp",
               "http", "t.c", "can", "u")
twdfm <- dfm(twcorpus,
             groups = "actor.id",
             remove = c(stopwords("english"), stopWords),
             remove_punct = TRUE,
             remove_numbers = TRUE,
             remove_symbols = TRUE,
             remove_url = TRUE,
             ngrams= 1L)

twdfm <- dfm_trim(twdfm, min_docfreq = 3)
```

Running LDA

```
library(topicmodels)

# convert dfm to data structure to work with topicmodels
dtm <- convert(twdfm, to="topicmodels")

# estimate LDA with K topics
K <- 20
lda <- LDA(dtm, k = K, method = "Gibbs",
                control = list(verbose=25L,
                               seed = 123,
                               burnin = 100,
                               iter = 500))
```

```
## K = 20; V = 9399; M = 9705
## Sampling 600 iterations!
## Iteration 25 ...
## Iteration 50 ...
## Iteration 75 ...
## Iteration 100 ...
## Iteration 125 ...
```

Five Sample Topics

```
term <- terms(lda, 10)
colnames(term) <- paste("Topic",1:K)

knitr::kable(term[,1:5])
```

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
like	#nc	time	just	game
just	bank	home	posted	panthers
really	see	arena	photo	win
one	#realestate	warner	school	bowl
right	america	cable	day	super
people	stadium	work	high	team
still	looking	game	church	man
think	check	#charlotte	#repost	play
oh	hot	hair	sc	good
know	drive	studio	rock	season

Five More Sample Topics

```
knitr::kable(term[,6:10])
```

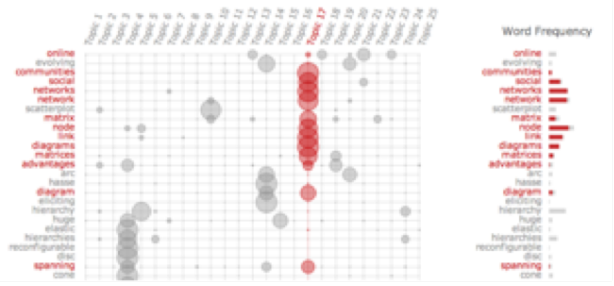
Topic 6	Topic 7	Topic 8	Topic 9	Topic 10
gt	will	drinking	great	#charlotte
now	never	#photo	go	#clt
f	get	beer	got	rd
wind	just	chicken	back	#traffic
weather	people	coffee	better	w
charlotte	like	ipa	thanks	accident
panthers	know	brewery	awesome	n
will	first	good	today	serious
mph	god	try	sure	st
pressure	man	company	good	south

Difficulty with Unsupervised Learning

- How to interpret clusters (topics)?
- How many clusters (topics)?
- How to validate?
- How to avoid overfitting?

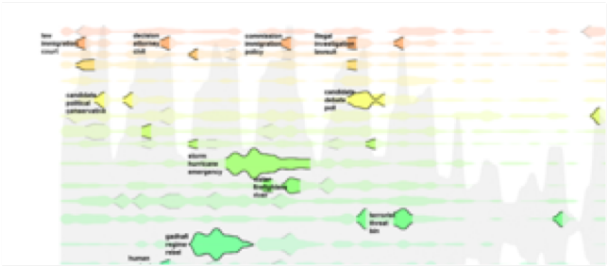
Visualizing Topic Models

Topic-Oriented



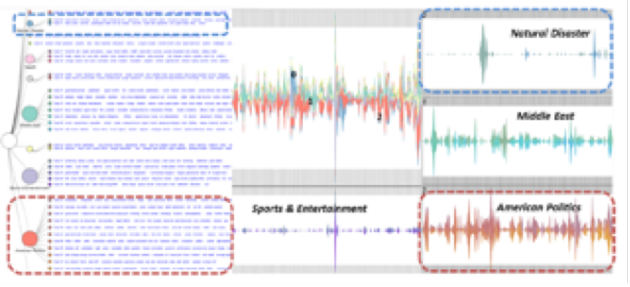
Termite (Chuang et al 2012)

Time-Oriented



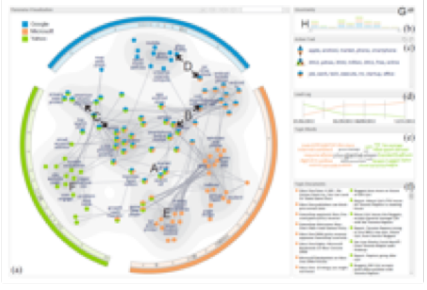
LeadLine (Dou et al 2012)

Hierarchical



HierarchicalTopics (Dou et al 2013)

Graph (Relational) Based



TopicPanorama (Wang et al 2016)

Given topic models are unsupervised, visualizations are **critical** to interpretating topic models.

Text Pre-Processing: Why It's More Important Than You Think

Importance of Pre-Processing

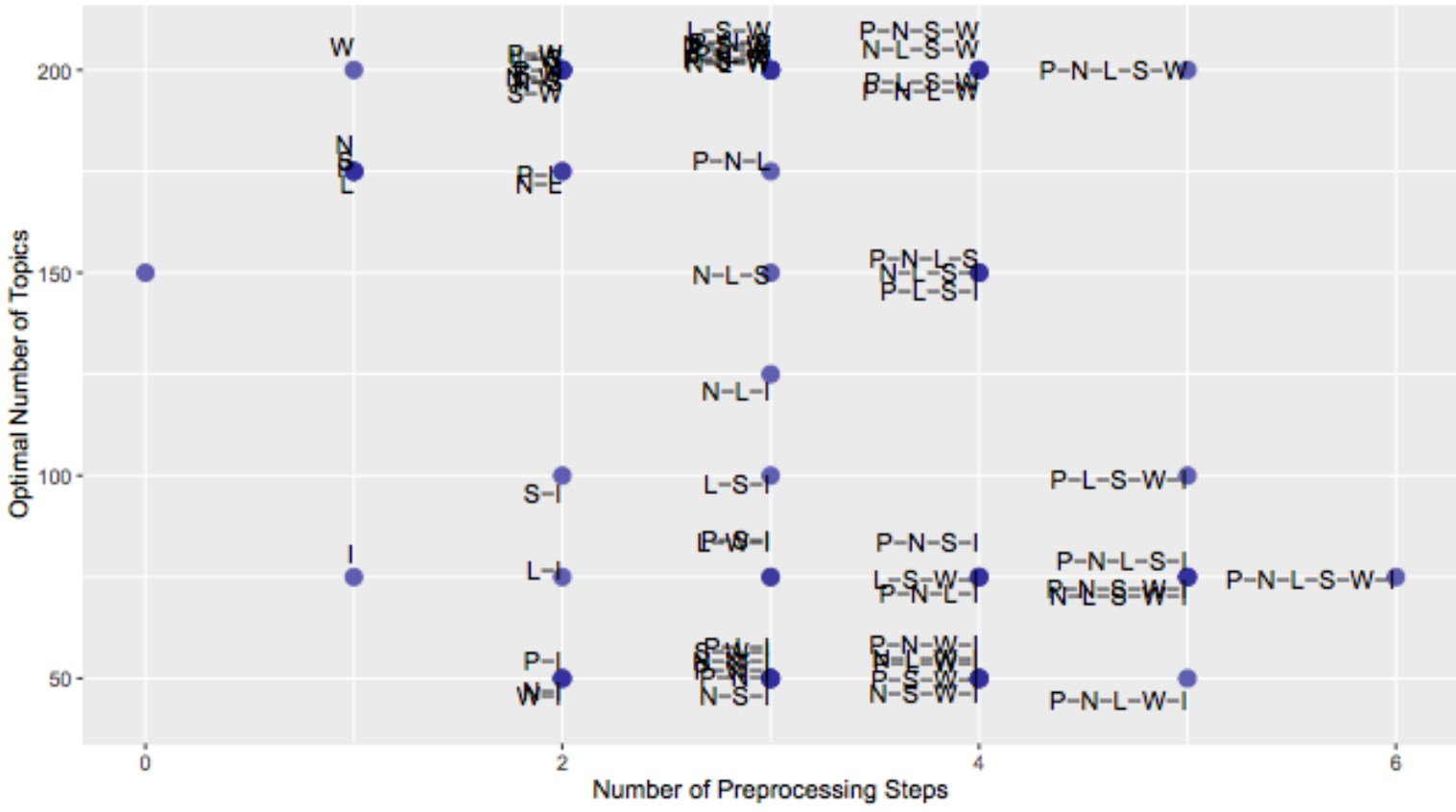


Figure 2: Plot depicting the optimal number of topics (as selected via perplexity) for each of 64 preprocessing specifications not including trigrams. On the x-axis is the number of preprocessing steps, and the y-axis is the number of topics. Each point is labeled according to its specification.

Denny & Spirling, 2017

Importance of Pre-Processing

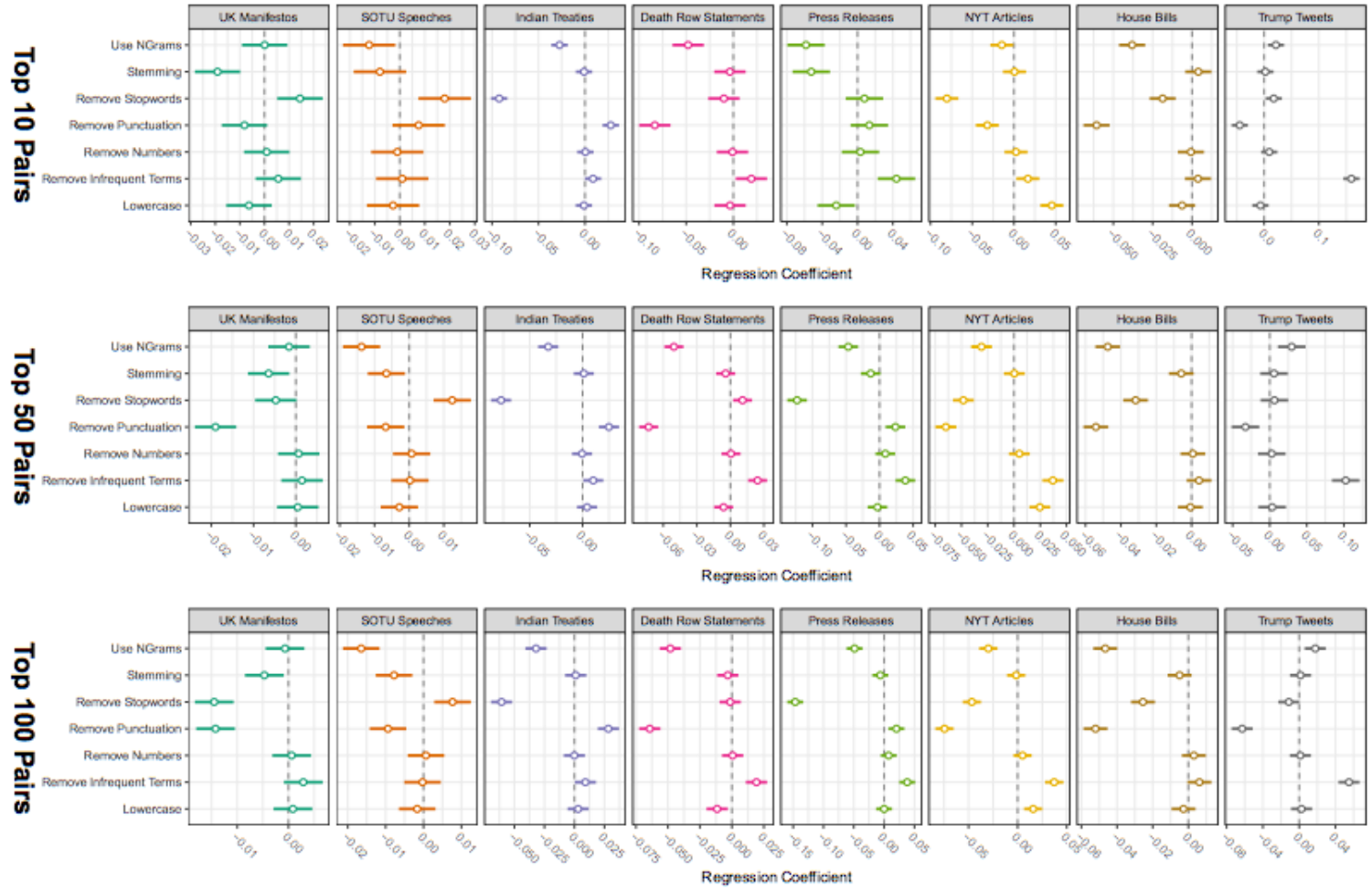


Figure 5: Regression results depicting the effects of each of the seven preprocessing steps on the **preText** score for that preprocessing combination.

http://www.mjdenny.com/getting_started_with_preText.html

To Infinity and Beyond: Word Embedding & Deep Learning

Word Embedding

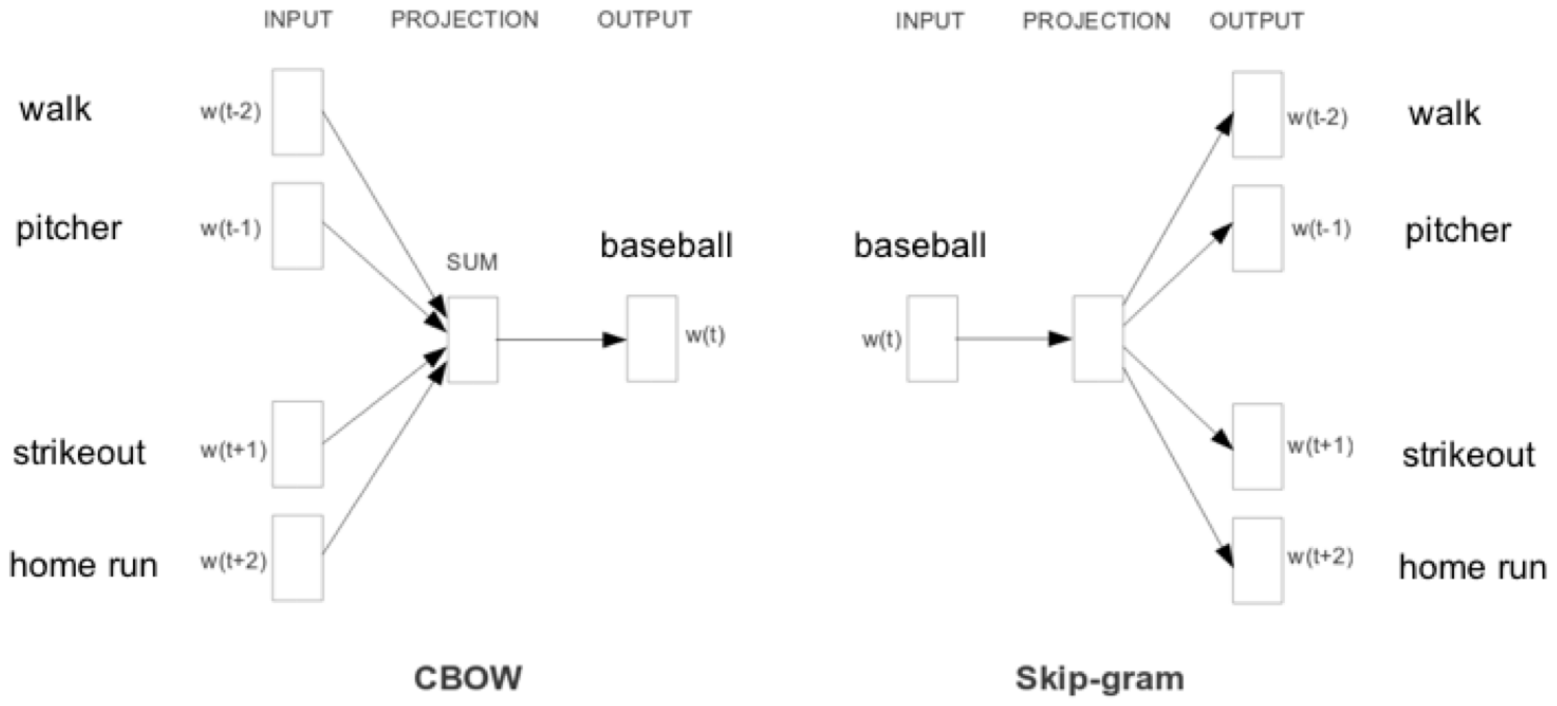
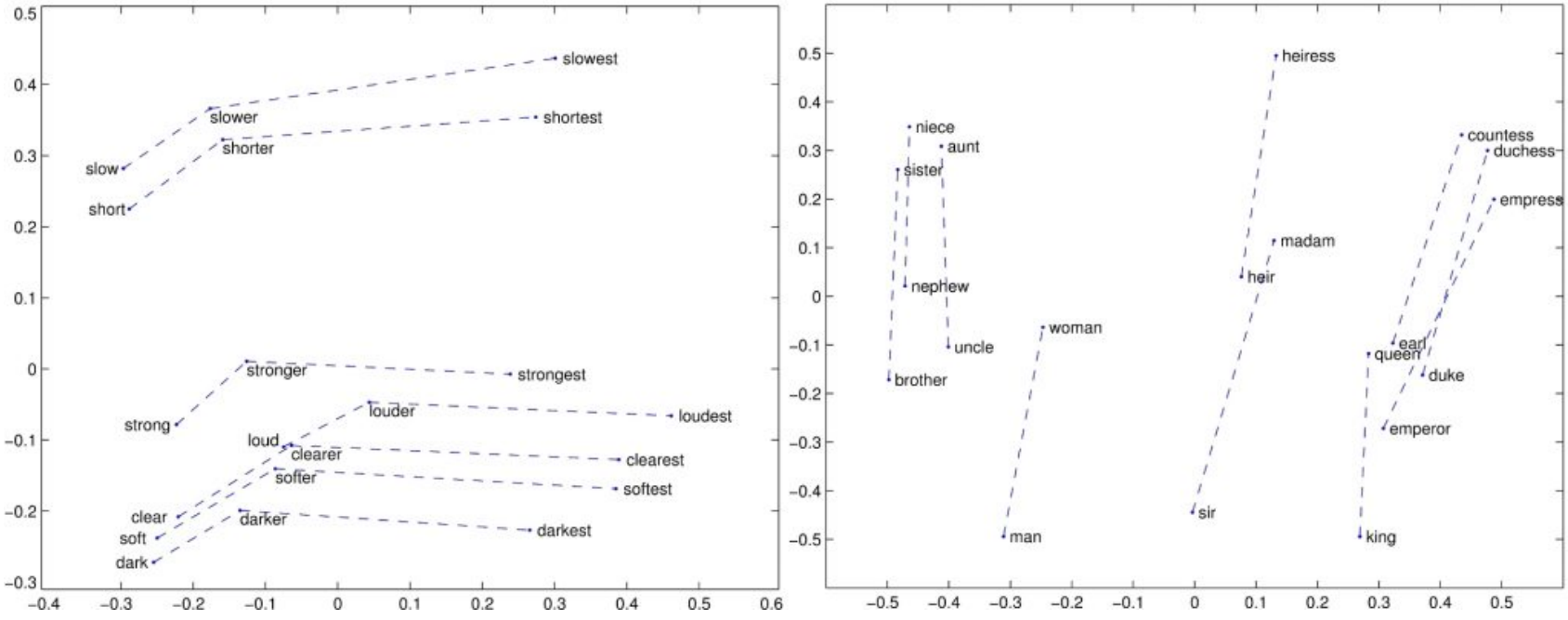


Figure 1: New model architectures. The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word.

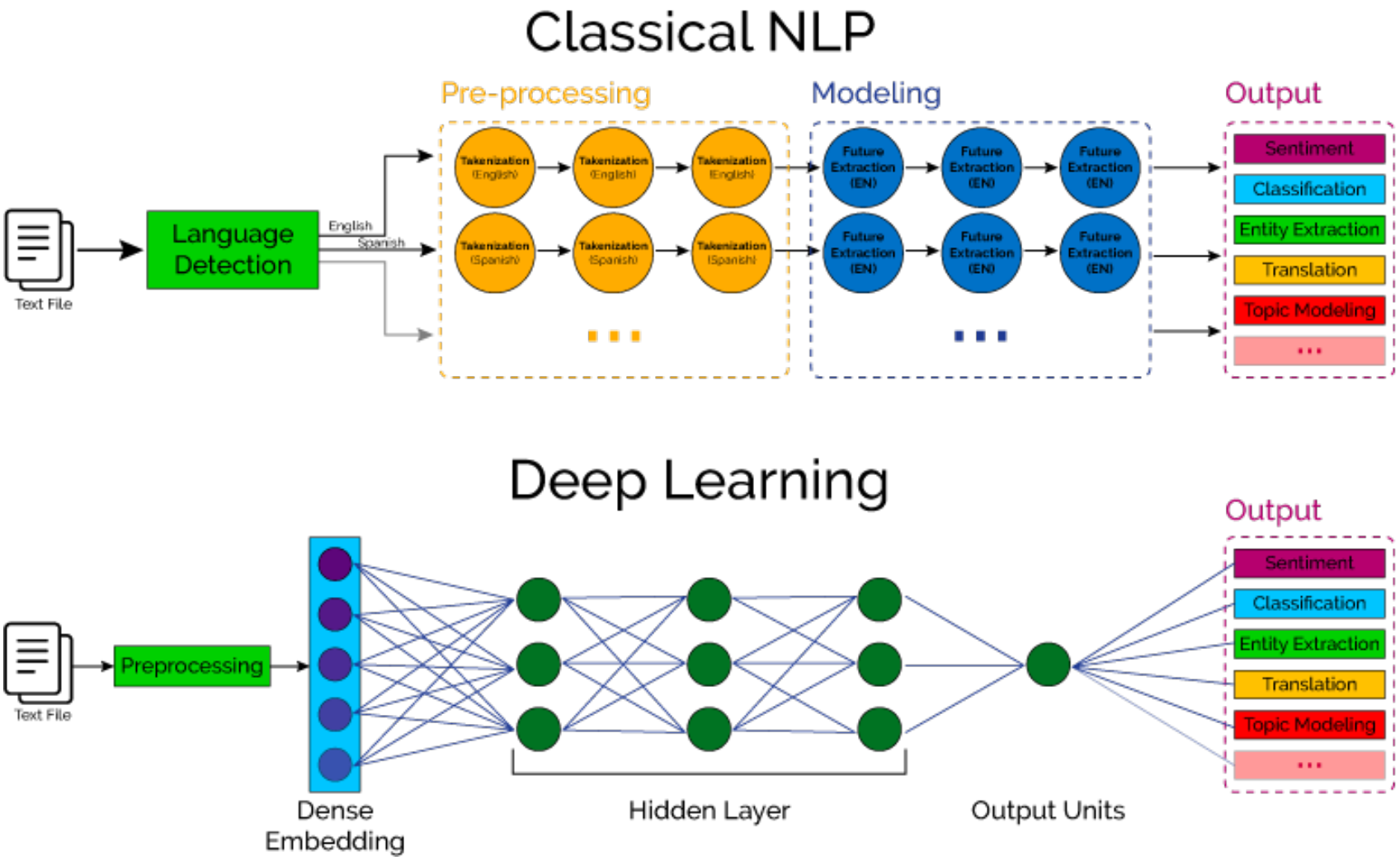
Mikolov et al., 2013

GloVe Model



Pennington, Socher, and Manning, 2014

Deep Learning



Credit: blog.aylien.com