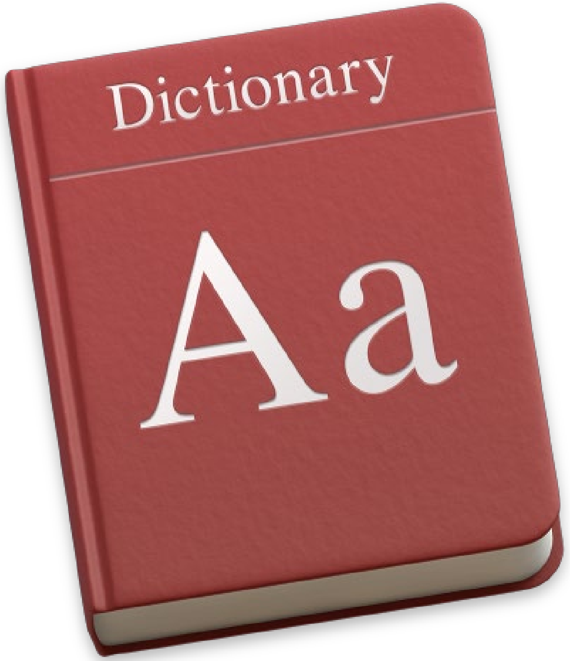
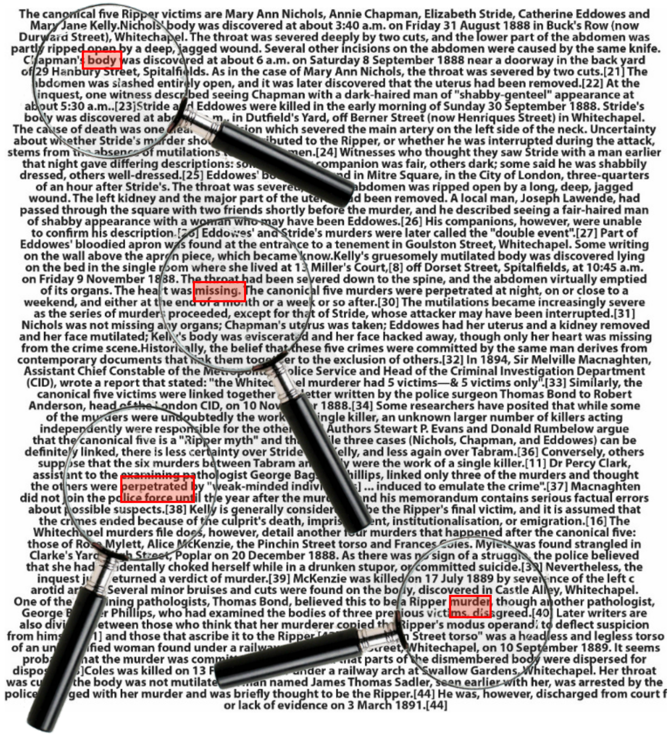
KEDGE Faculty Training

Automated Text Analysis – DAY 2

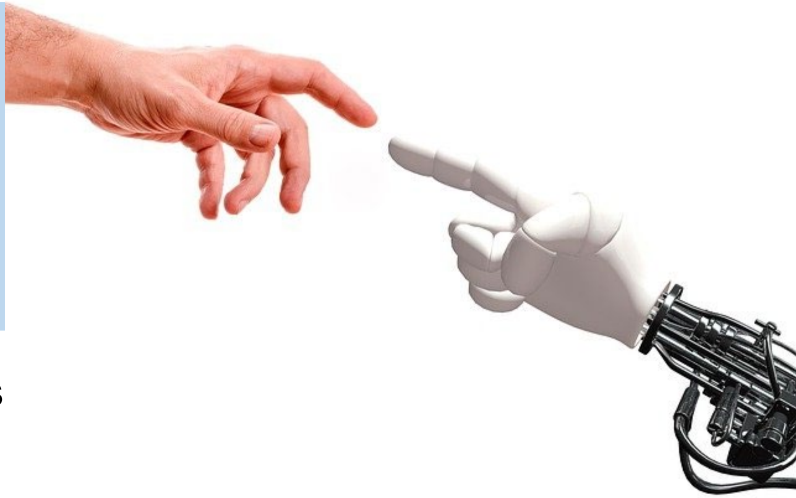
Prof. Dr. Dennis Herhausen, KEDGE Business School

4) Classification with Dictionaries





Bridging Qualitative and Quantitative Text Analysis



- A **hybrid procedure** between qualitative and quantitative text classification
  - ✓ "Qualitative" since it involves identification of the concepts and associated keys (categories), and the textual features associated with each key (category)
  - ✓ Dictionary construction involves contextual interpretation and qualitative judgment
  - ✓ "Quantitative" since perfect reliability in the actual text analysis procedure
- A dictionary is really a **thesaurus**: a "key" associated with a list of equivalent synonyms
  - ✓ self = I, me, my, mine, myself, …
  - ✓ selves = we, us, our, ours, ourselves, …
- Rather than count words that occur, **pre-define words** associated with specific meanings
- Involves stemming and lemmatization: transformation to the **dictionary look-up form**
- Many programs will return an **intensity score**: This is calculated as the sum words or expressions related to a category divided by the sum of total words used in a document).
  - ✓ Using intensities or proportions overcomes the **problem with simple counts**, where longer documents would be more likely to include more occurrences of every entity

Examples for Standardized Dictionaries

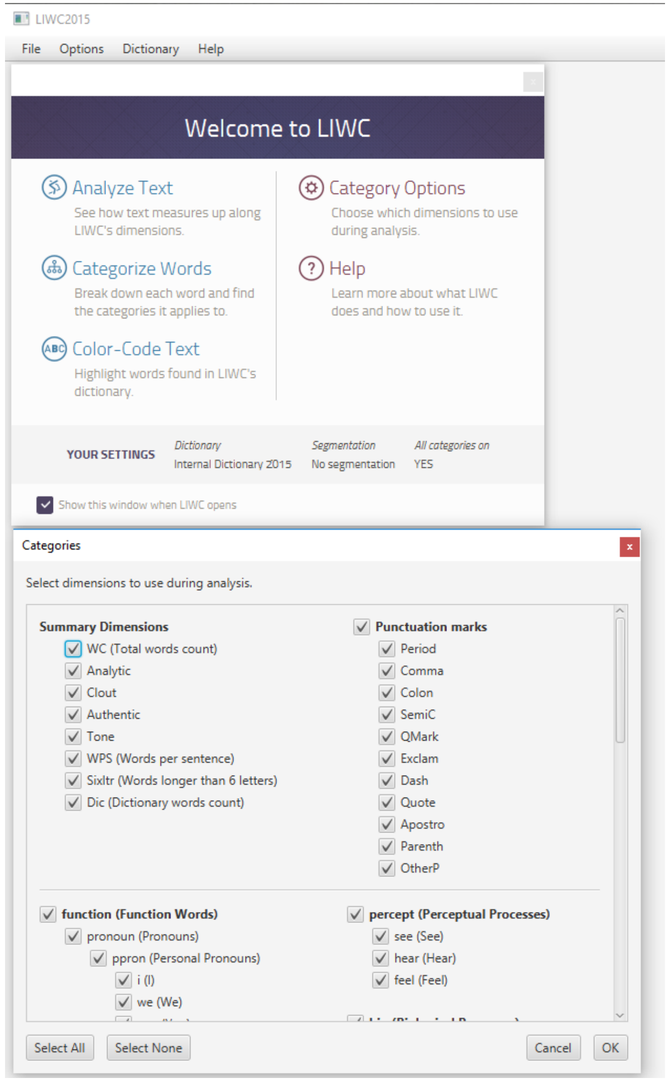
1. **LIWC:** A broad dictionary to measure parts of speech, but also psychological and social categories and processes (https://liwc.wpengine.com/)
2. **VADER:** A lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media (https://cran.r-project.org/web/packages/vader/index.html)
3. **The Evaulative Lexicon:** A computational linguistic tool that quantifies the emotionality, valence, and extremity of individuals' evaluations in natural text (http://www.evaluativelexicon.com/)
4. **SentiStrenght:** Estimates the strength of positive and negative sentiment in short texts, even for informal language (http://sentistrength.wlv.ac.uk/)
5. **Whissell Dictionary of Affect in Language**: A tool for the statistical analysis of individual words according to the way they ‘feel’ (https://www.god-helmet.com/wp/whissel-dictionary-of-affect/index.htm)
6. **WordNet:** A large lexical dictionary that categories a variety of objects, feelings, and processes (https://wordnet.princeton.edu/)
7. **Concreteness:** A weighted word list to measure concreteness based on 4,000 participants’ ratings of the concreteness of many common words (https://cran.r-project.org/web/packages/doc2concrete/)

→ See Humphreys and Wang (2018) for more dictionaries

LIWC = Linguistic Inquiry and Word Count

Created by James Pennebaker
(https://liberalarts.utexas.edu/psychology/faculty/pennebak)

- Uses a dictionary to calculate the percentage of words in the text that match each of up to **82 language dimensions**
- Consists of about **4,500 words and word stems**, each defining one or more word categories or sub-dictionaries
  - ✓ For example, the word "cried" is part of five word categories: sadness, negative emotion, overall affect, verb, and past tense verb.
  - ✓ So observing the token "cried" causes each of these five sub-dictionary scale scores to be incremented (the word is counted in each sub-dictionary)
- **Hierarchical:** the category "anger" is part of an emotion category and a negative emotion subcategory
- Two good summaries on the interpretation of LIWC dimensions:
  - ✓ Pennebaker (2011). *The Secret Life of Pronouns: What Our Words Say About Us*.
  - ✓ Tausczik & Pennebaker (2010). The psychological meaning of words: LIWC and computerized text analysis methods. *Journal of Language and Social Psychology*, 29, 24-54.



http://www.liwc.net

Some Examples from LIWC

Category	Examples
Indefinite pronouns	It, it's, those
Articles	A, an, the
Common verbs	Walk, went, see
Auxiliary verbs	Am, will, have
Past tense	Went, ran, had
Present tense	Is, does, hear
Future tense	Will, gonna
Adverbs	Very, really, quickly
Prepositions	To, with, above
Conjunctions	And, but, whereas
Negations	No, not, never
Quantifiers	Few, many, much
Numbers	Second, thousand
Swear words	Damn, piss, fuck
Psychological processes	
Social processes	Mate, talk, they, child
Family	Daughter, husband
Friends	Buddy, friend, neighbor
Humans	Adult, baby, boy
Affective processes	Happy, cried, abandon

Category	Examples
Positive emotion	Love, nice, sweet
Negative emotion	Hurt, ugly, nasty
Anxiety	Worried, nervous
Anger	Hate, kill, annoyed
Sadness	Crying, grief, sad
Cognitive processes	Cause, know, ought
Insight	Think, know, consider
Causation	Because, effect, hence
Discrepancy	Should, would, could
Tentative	Maybe, perhaps, guess
Certainty	Always, never
Inhibition	Block, constrain, stop
Inclusive	And, with, include
Exclusive	But, without, exclude

Category	Examples
Perceptual processes	Observing, heard, feeling
See	View, saw, seen
Hear	Listen, hearing
Feel	Feels, touch
Biological processes	Eat, blood, pain
Body	Cheek, hands, spit
Health	Clinic, flu, pill
Sexual	Horny, love, incest
Ingestion	Dish, eat, pizza
Relativity	Area, bend, go
Motion	Arrive, car, go
Space	Down, in, thin
Time	End, until, season
Personal concerns	
Work	Job, majors, xerox
Achievement	Earn, hero, win
Leisure	Cook, chat, movie
Home	Apartment, kitchen, family
Money	Audit, cash, owe
Religion	Altar, church, mosque
Death	Bury, coffin, kill
Spoken categories	
Assent	Agree, OK, yes
Nonfluencies	Er, hm, umm
Fillers	Blah, Imean, yaknow

If a Standard Dictionary exists, use it!

Choosing one (or more) standardized dictionaries versus creating a custom dictionary:

- Similar to existing scales, standardized dictionaries have been **validated with a large and varied number of text corpora**
- Because operationalization does not change, standard dictionaries enable **comparison across research**
- For this reason, if a standard dictionary exists, you should use it if at all possible to enhance the **replicability of your study**.
- If you wish to create a **new dictionary for an existing construct**, you should run and compare the new dictionary to any existing dictionary for the construct, just as one would with a newly developed scale

???

➢ Reviewers often ask to use a different, related dictionary for a robustness test

➢ Sometimes the use of existing dictionaries is questioned because of the context (e.g., important contextual words are missing) or the data type (e.g., “Tweets are too short”)

➢ Carefully argue for your approach upfront and do all necessary robustness tests

Building your Own Dictionary

The ideal dictionary associates all and only the relevant words to each category in a perfectly valid scheme.

A **good dictionary** ensures that all documents that match the dictionary contain the desired concept, and that all documents that contain the concept are matched.

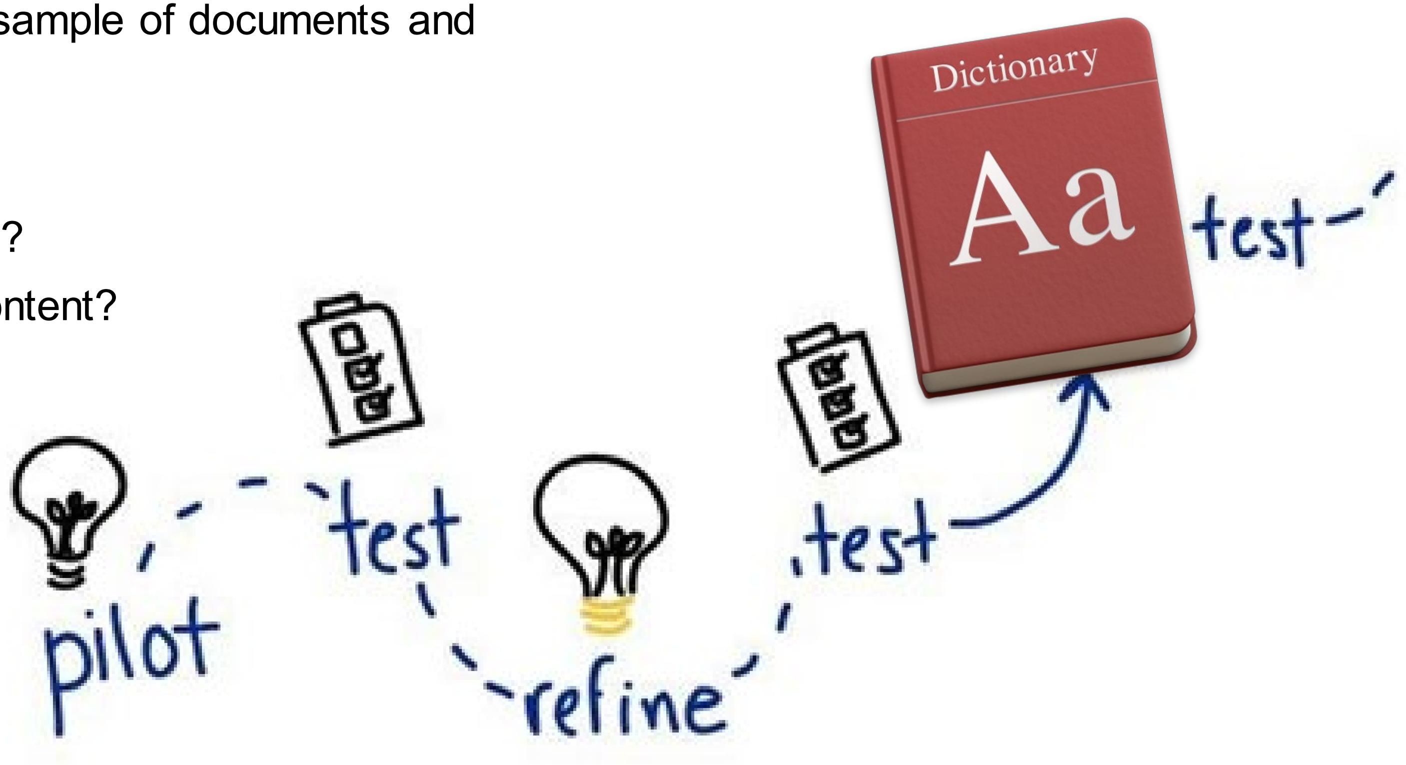
To check this, you can **manually annotate or code** a sample of documents and compare the score with the dictionary hits.

Three key issues:

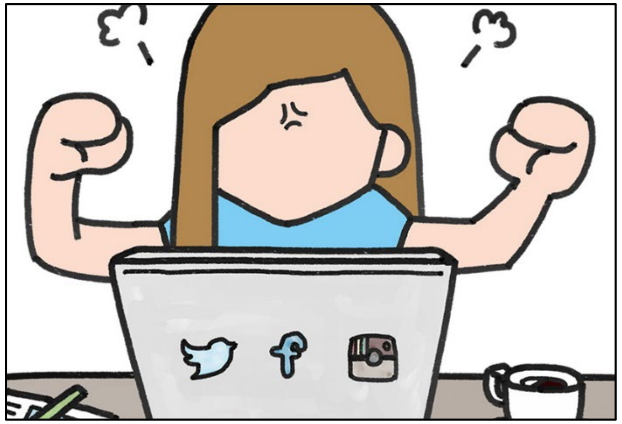
1. **Validity:** Is the dictionary's category scheme valid?
2. **Sensitivity:** Does this dictionary identify all my content?
3. **Specificity:** Does it identify only my content?

Imagine two logical extremes:

- including all words (too sensitive)
- including just one word (too specific)



Building a Dictionary for “disgust”



<u>**Detecting, Preventing, and Mitigating Online Firestorms in Brand Communities**</u>

Initial submission: Low arousal emotions (sadness) and high arousal emotions (fear/anxiety, anger)

Reviewer comment: “*The authors build their arguments on emotional contagion theory. Unfortunately, some important works on emotional contagion and WOM transmission have been largely neglected. Accordingly the specific negative emotions that are supposedly contagious in the negative WOM transmission are not clear at all, and whether the nature of these negative emotions such as disgust is important in predicting the potential firestorms has been overlooked.*”

Additional measurement of disgust with a **newly developed dictionary**:

- Three independent coders populated a **list of words unambiguously indicative of disgust**
- Initial list produced **134 words**. Each coder indicated whether each word reflected disgust (yes or no)
- The coders achieved a **Krippendorff’s α** of 96% (and disagreements were resolved in a discussion)

aberration, abhor, abject, abnormal, abominat*, appall*, averse, bloated, constipat*, contaminate, contaminated, crap, cringeworthy, decay*, degrad*, deteriorated, detestation, dirt, dirty, disgust*, distast*, distorted, dung, entrails, excrement, execration, feces, filth, filthy, flabby, flatulence, gaby, gag, garbage, grime, grimy, gross, grotesque, gruesome, gutter, heretic, herpes*, hideous, incest*, infestation, latrines, lewd, loathing, loo, maggot, mess, messy, mildew, mire, muck, muddy, musty, mutilated, nause*, nauseous, obscene, ooze, perver*, pig*, pollut*, puk*, pungen*, purgator*, rags, rancid, regurgitation, repel*, repelling, repugnan*, repuls*, revuls*, rot, rotting, rubbish, scum, sewage, sewer, sewerage, shabby, sicken*, sickness, slime, slimy, slop, sloth, sludge, spew, spit, squirm, stain, sticky, stink, stinking, swig, tasteless, toad, trash, trashy, ugly, unclean, untidy, unwashed, vomit, vulgar, wart, weird, withered*

Herhausen et al. 2019

Create and Use a Simple Dictionary

```
# Create a new dictionary with four categories
dict <- dictionary(list(terrorism = 'terror*',
        economy = c('econom*', 'tax*', 'job*'),
        military = c('army','navy','military','airforce','soldier'),
        freedom = c('freedom','liberty')))

# Use the dictionary to lookup categories in the speeches
dict_dtm <- dfm_lookup(dtm_inaugural, dict, exclusive=TRUE)

# Convert the output into a dataframe
df_inaugural <- convert(dict_dtm, to="data.frame")

# Inspect the dataframe with the output
View(df_inaugural)

# Convert the dataframe into a .csv file
write.csv(df_inaugural,'df_inaugural.csv')

# Convert the dataframe into a .xlxs file
write.xlsx(df_inaugural,'df_inaugural.xlsx')
```

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function | Addins

Teaching R - 03 simple dictionary.R | df | Teaching R - 02 preprocess.R

Filter

	document	terrorism	economy	military	freedom
1	1789-Washington	0	1	0	0
2	1793-Washington	0	0		
3	1797-Adams	1	0		
4	1801-Jefferson	0	1		
5	1805-Jefferson	0	3		
6	1809-Madison	0	1		
7	1813-Madison	0	0		
8	1817-Monroe	0	2		
9	1821-Monroe	0	6		
10	1825-Adams	0	3		
11	1829-Jackson	0	1		
12	1833-Jackson	0	2		
13	1837-VanBuren	0	1		
14	1841-Harrison	0	0		
15	1845-Polk	0	8		
16	1849-Taylor	0	1		
17	1853-Pierce	0	1		
18	1857-Buchanan	0	3		
19	1861-Lincoln	0	0		
20	1865-Lincoln	0	0		

Showing 1 to 20 of 58 entries

	A	B	C	D	E
1	document	terrorism	economy	military	freedom
40	1941-Roosevelt	1	0	0	6
41	1945-Roosevelt	0	0	0	1
42	1949-Truman	0	7	0	12
43	1953-Eisenhower	0	4	3	10
44	1957-Eisenhower	0	1	0	11
45	1961-Kennedy	2	0	0	4
46	1965-Johnson	0	0	0	2
47	1969-Nixon	0	1	0	2
48	1973-Nixon	0	0	0	4
49	1977-Carter	0	0	0	4
50	1981-Reagan	2	14	0	8
51	1985-Reagan	0	15	0	14
52	1989-Bush	0	1	0	6
53	1993-Clinton	0	4	0	3
54	1997-Clinton	2	3	0	2
55	2001-Bush	0	3	0	5
56	2005-Bush	0	1	2	27
57	2009-Obama	1	7	0	3
58	2013-Obama	0	7	2	7
59	2017-Trump	1	6	1	1

https://quanteda.io/

Using an Existing Dictionary

laver-garry.cat is a dictionary that contains political left-right ideology keywords:

https://raw.githubusercontent.com/quanteda/tutorials.quanteda.io/master/content/dictionary/laver-garry.cat

```
# Use the laver-garry.cat dictionary
dict_lg <- dictionary(file = "C:/01 Teaching/KEDGE # Text Mining/laver-garry.cat",
             encoding = "UTF-8")
###Use your own folder!!!###

# Use the dictionary to lookup categories in the speeches
dict_dtm_lg <- dfm_lookup(dtm_inaugural, dict_lg, exclusive=TRUE)

# Convert the output into a dataframe
df_inaugural_lg <- convert(dict_dtm_lg, to="data.frame")

# Inspect the dataframe with the output
View(df_inaugural_lg)

# Convert the dataframe into a .xlxs file
write.xlsx(df_inaugural_lg,'df_inaugural_lg.xlsx')
```

Estimating Policy Positions from Political Texts

Michael Laver Trinity College Dublin
John Garry Trinity College Dublin

The analysis of policy-based party competition will not make serious progress beyond the constraints of (a) the unitary actor assumption and (b) a static approach to analyzing party competition between elections until a method is available for deriving reliable and valid time-series estimates of the policy positions of large numbers of political actors. Retrospective estimation of these positions in past party systems will require a method for estimating policy positions from political texts.

Previous hand-coding content analysis schemes deal with policy emphasis rather than policy positions. We propose a new hand-coding scheme for policy positions, together with a new English language computer-coding scheme that is compatible with this. We apply both schemes to party manifestos from Britain and Ireland in 1992 and 1997 and cross validate the resulting estimates with those derived from quite independent expert surveys and with previous manifesto analyses.

There is a high degree of cross validation between coding methods, including computer coding. This implies that it is indeed possible to use computer-coded content analysis to derive reliable and valid estimates of policy positions from political texts. This will allow vast volumes of text to be coded, including texts generated by individuals and other internal party actors, allowing the empirical elaboration of dynamic rather than static models of party competition that move beyond the unitary actor assumption.

Deriving reliable and valid estimates of the policy positions of key actors is fundamental to the analysis of political competition. Various systematic methods have been used to do this, including surveys of voters, politicians, and political scientists, and the content analysis of policy documents. Each method has advantages and disadvantages but, for both theoretical and pragmatic reasons, policy documents represent a core source of information about the policy positions of political actors.

We explore various ways to extract information about policy positions from political texts. We are particularly interested in using computer-coding techniques to derive reliable and valid estimates of the policy positions of political actors. This is not mere laziness on our part, a lack of stomach for the hard graft of expert coding. If analyses of party competition are to move beyond both static models and a view of political parties as unitary actors, this requires information on the policy positions of actors inside political parties and on the development of these over time and between elections. The laborious expert "hand-coding" of text is simply not a viable method for estimating the policy positions of huge numbers of political actors, for example, all members of a legislature. Any serious attempt to operationalize a model of internal party policy competition, or of dynamic policy-based party competition or coalition government between elections, implies using computer-coding for estimating the policy positions of key political actors.

We first review existing methods for estimating policy positions from political texts. These have for the most part concentrated on the expert coding of party manifestos. We then suggest ways to improve these, dealing with both expert- and computer-coded content analysis. We then explore the impact of our suggestions upon estimates of party policy positions derived from British and Irish manifestos issued during the 1992 and 1997 general elections in each country, positions for which a range of

Michael Laver is a Professor of Political Science and Director of the Policy Institute, Trinity College, Dublin 2, Ireland (mlaver@tcd.ie). John Garry is a Ph.D. student of Political Science, Trinity College, Dublin 2, Ireland (garryj@tcd.ie).

Versions of this paper have been presented at the ECPR workshop on "Empirical rational choice theory," Warwick, April 1998 and the ECPR workshop on "Estimating the policy positions of political actors," Mannheim, March 1999. The authors are grateful to Ken Benoit, Ian Budge, Miranda de Vries, Matt Gabel, Daniela Giannetti, John Huber, Jan Kleinnijenhuis, Michael Marsh, Michael McDonald, Leonard Ray, and many other participants at these conferences and workshops, as well as three anonymous journal referees for their helpful and constructive comments.

American Journal of Political Science, Vol. 44, No. 3, July 2000, Pp. 619–634

©2000 by the Midwest Political Science Association

https://quanteda.io/

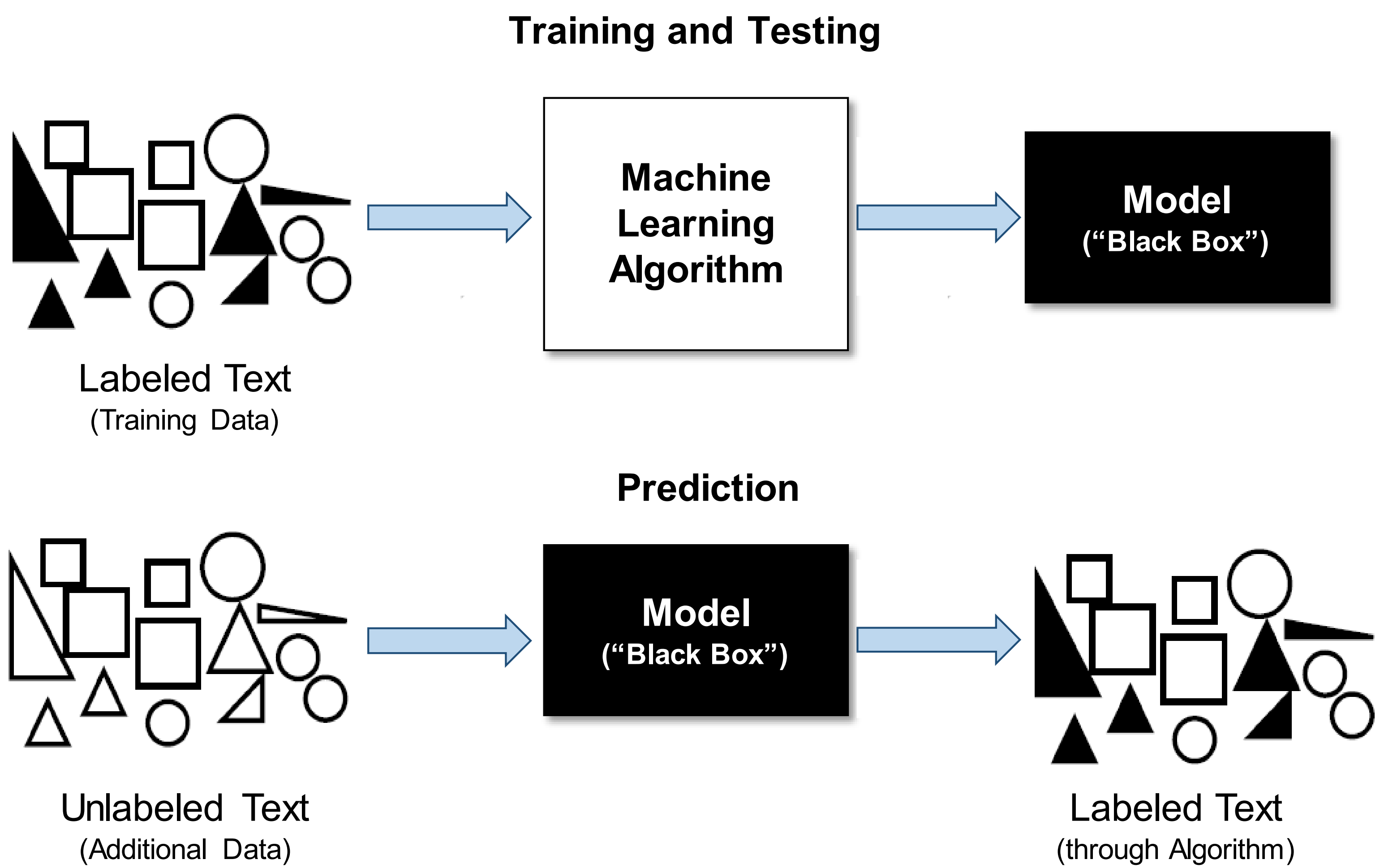
Exercise with Your Own Data

- **Use your own dictionary**
    1. Build your own simple dictionary
        - ✓ at least three categories with five words or word stems each
    2. Use your dictionary to analyze your documents
    3. Export the results of your analysis

- **Use an existing dictionary**
    1. Choose an existing dictionary for analysis (https://quanteda.io/reference/dictionary.html)
    2. Run the analysis and export the results

Try to use and adapt the code from the examples!

5) Classification with Supervised Machine Learning



Classification of Documents into Pre-Existing Categories

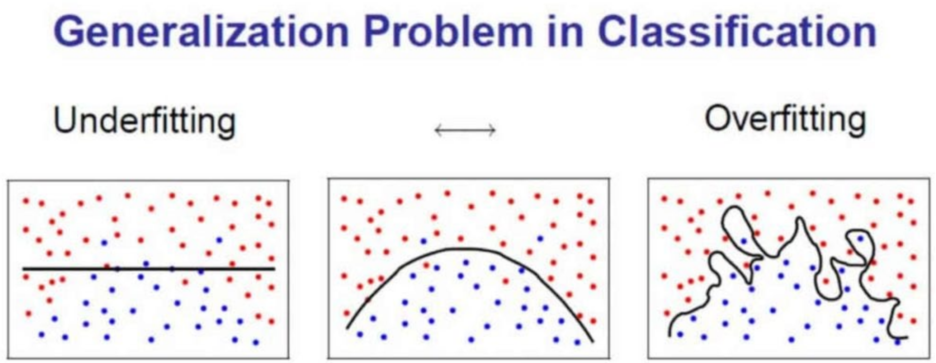
Dictionary methods

- Advantage: not corpus-specific, cost to apply to a new corpus is trivial
- Disadvantage: not corpus-specific, so performance on a new corpus is unknown (domain shift)

Supervised learning can be conceptualized as a generalization of dictionary methods, where features associated with each category are learned from the data. They will **outperform dictionary methods** in classification tasks as long as the training sample is large enough.

What we need:

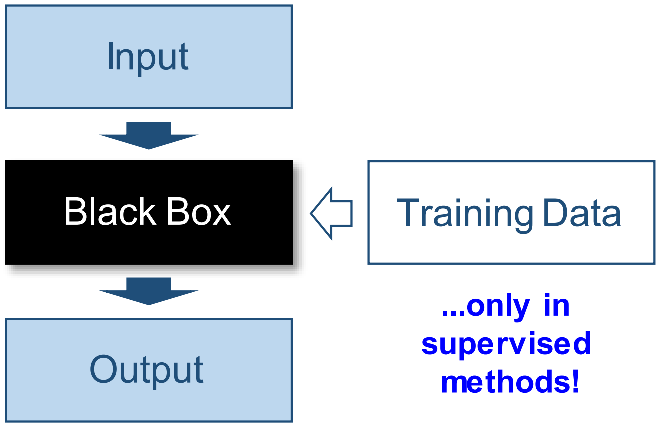
- Hand-coded dataset (labeled), to be split into:
  - ✓ **Training set** used to train the classifier
  - ✓ **Validation/Test set** used to validate the classifier
- Method to extrapolate from hand coding to unlabeled documents (**classifier**):
  - ✓ **Naive Bayes**, SVM, K-nearest neighbor, regularized regression, ...
- Approach to validate classifier: **cross-validation**
- **Performance metric** to choose best classifier and avoid overfitting
  - ✓ Confusion matrix, accuracy, precision, recall...



https://quanteda.io/

Choice of Supervised vs. Unsupervised Methods

The goal in both methods is to **differentiate documents from one another**, treating them as bags-of-words (i.e., depending on the words in each document).



<u>Different approaches:</u>

- Supervised methods require a **training set** that exemplify **contrasting classes**, identified by the **researcher**
- Unsupervised methods scale documents based on **patterns of similarity** from the **term-document matrix** (**no training step**)

Relative advantage of supervised methods: You already know the dimension being scaled, because you set it in the training stage.

Relative disadvantage of supervised methods: You must already know the dimension being scaled, because you have to feed it good sample documents in the training stage.

➢ Depending on research question and research approach

➢ Unsupervised methods are rather explorative and need post-hoc interpretation

https://quanteda.io/

Basic Principle of Supervised Learning

Generalization goal: A classifier algorithm learns to correctly predict output from given inputs not only in previously seen samples but also in previously unseen samples

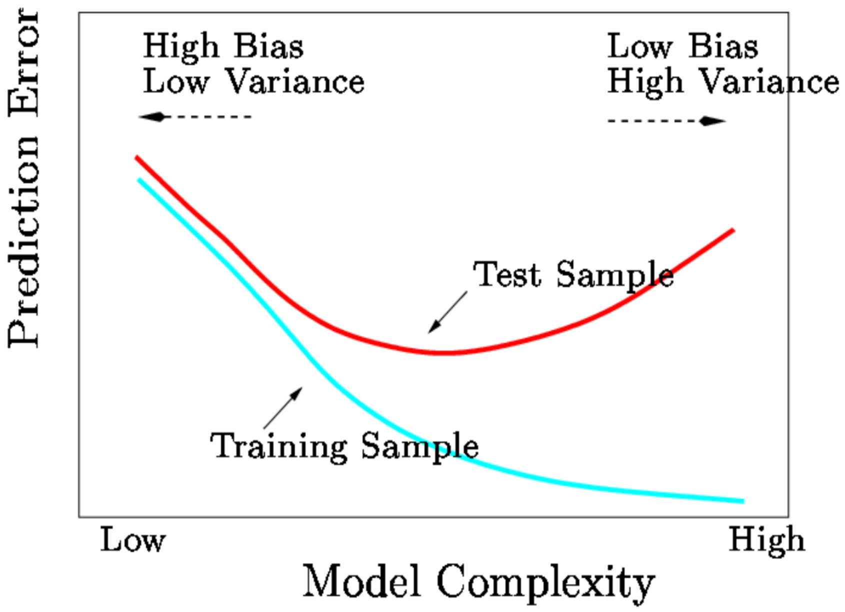
Classifier algorithm is trained to maximize **in-sample performance** but generally we want to apply method to new data

- ***Underfitting:*** A classifier algorithm fails to
- ***Overfitting:*** A classifier algorithm learns to correctly predict output from given inputs in previously seen samples but fails to do so in previously unseen samples. Model is too complex, describes noise rather than signal (Bias-Variance trade-off)

Both causes **poor prediction and generalization** to new data!

➢ Solutions include randomly split dataset into training and test set and cross-validation of methods

➢ Combination with other machine learning packages in R such as caret (https://cran.r-project.org/web/packages/caret/vignettes/caret.html)



https://quanteda.io/

Obtaining Training Data

<u>How to create a labeled set?</u>

External sources of annotation

- Stars or ratings in product reviews *(dichotomized if necessary)*
- "Helpfulness" ratings in online communities
- etc...

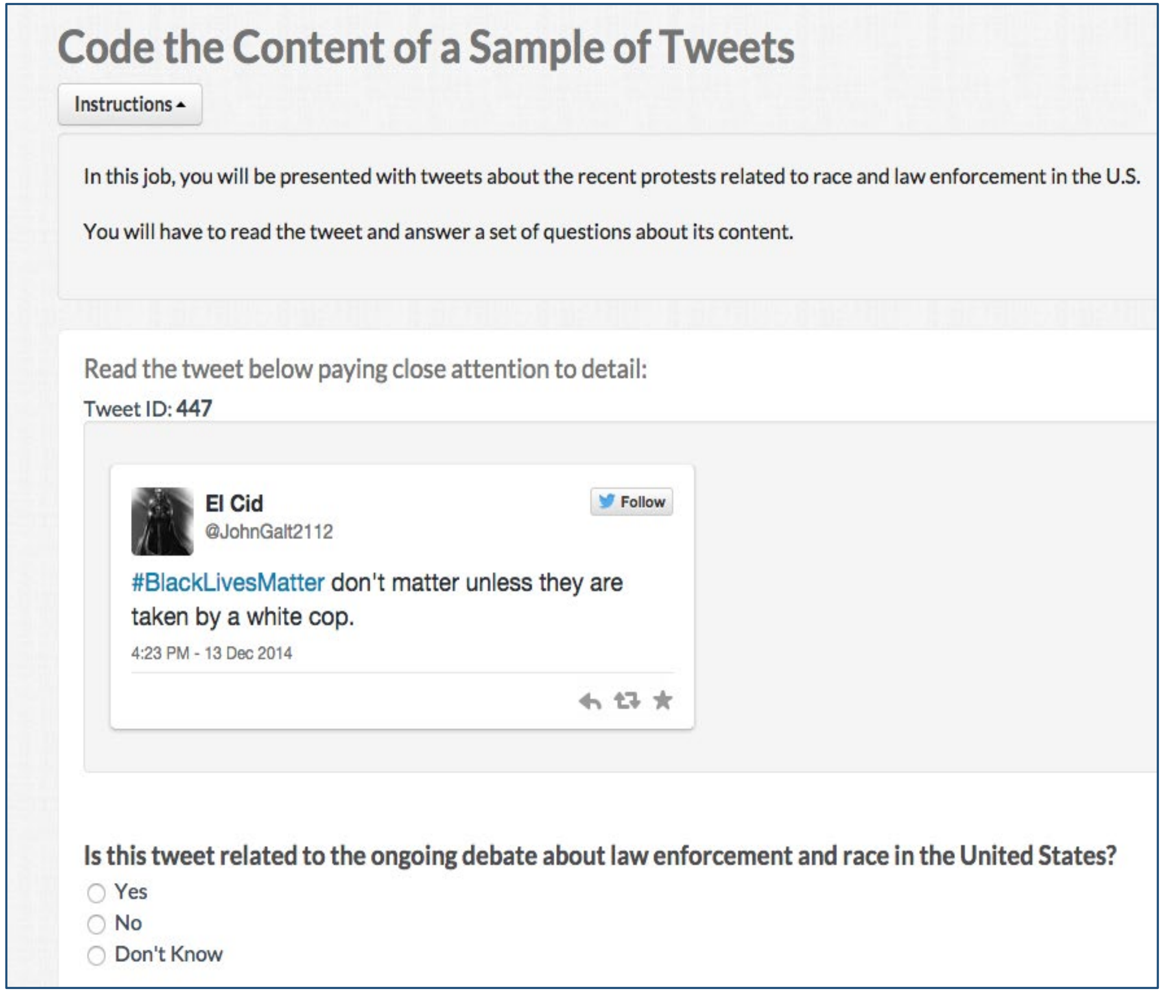
Expert annotation

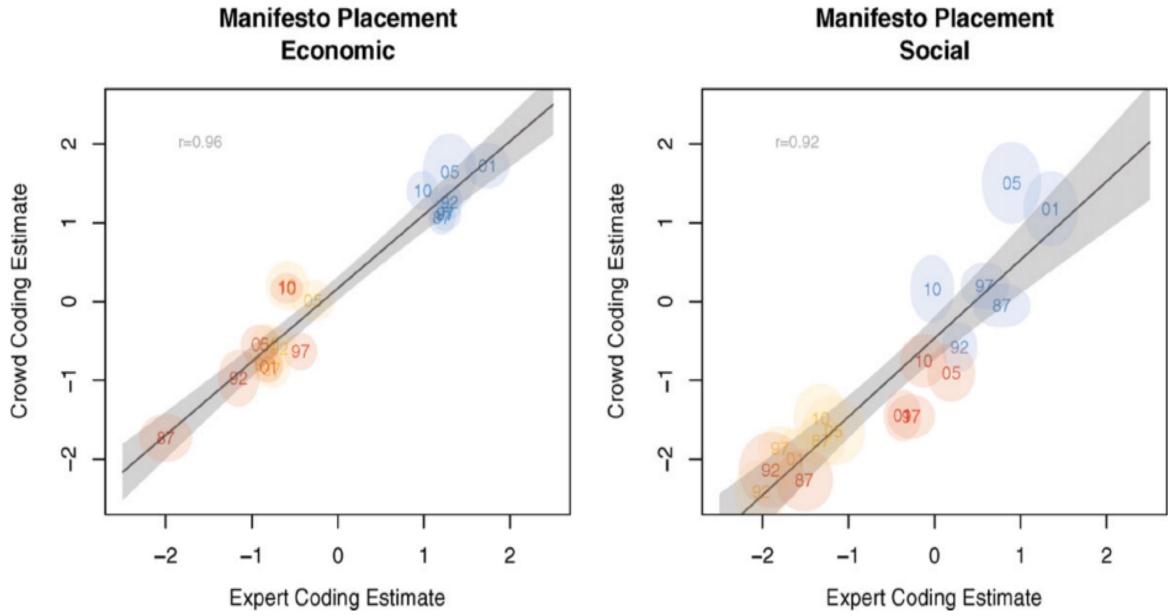
- Labeling (only) through the research team *(not suggested!)*
- In most projects → undergraduate students *(expertise comes from training)*

Crowd-sourced coding

- Wisdom of crowds: aggregated judgments of non-experts converge to judgments of experts at much lower cost *(e.g., Benoit et al 2016)*
- Easy to implement with MTurk or similar websites *(costs are quite low)*

Validity of Crowd-Sourced Coding





Benoit et al. 2016

Evaluating the Quality of a Labeled Set

Any labeled set should be tested and reported for its reliability:

Percent agreement

- Very simple: (number of agreeing ratings) / (total ratings) * 100%

Correlation

- Pearson's r or ordinal such as Spearman's rho or Kendall's tau-b, range is [0,1]

Agreement measures

- Not only observed agreement, but also agreement that would have occurred by chance (Krippendorf's α, range is [0,1])

Article:	1	2	3	4	5	6	7	8	9	10
Coder A	1	1	0	0	0	0	0	0	0	0
Coder B	0	1	1	0	0	1	0	1	0	0

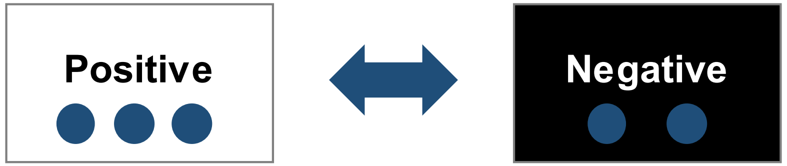
- ➢ A and B agree on 60% of the articles: 60% agreement
- ➢ Correlation *r* is (approximately) .10
- ➢ Observed disagreement = 4, Expected disagreement (by chance) = 4.4211, Krippendorf's α = .095 $(1 - \frac{OD}{ED})$

Methods to Extrapolate Manual Coding

Main models for classification (class prediction):

Naive Bayes classifier for texts: Fit a multinomial or Bernoulli Naive Bayes model, given a document-feature matrix and some training labels.

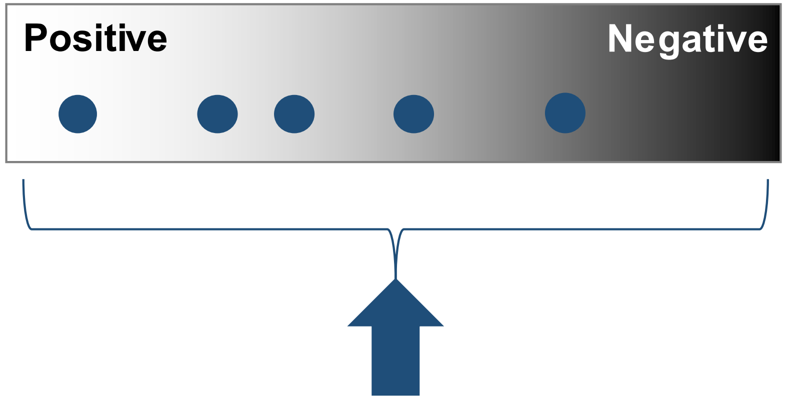
Linear Support Vector Model classifier for texts: Fit a fast linear Support Vector Model classifier for texts, using the LiblineaR package (https://cran.r-project.org/web/packages/LiblineaR/index.html)



Models for scaling (derive latent positions from text data):

For some problems it is more interesting to estimate the **positions of a text on a certain dimension** that is specified a priori.

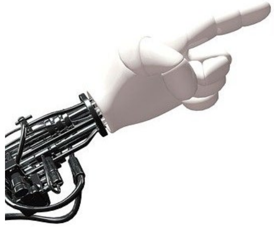
- Obtain reference texts with a priori known positions
- Generate word scores from reference texts
- Score each virgin text using word scores



➢ Not considered today! See https://tutorials.quanteda.io/machine-learning/wordscores/

Cross-Validation with Confusion Matrix

		Labeling	
		Positive	Negative
Prediction	Positive	True Positive	False Positive (Type I error)
	Negative	False Negative (Type II error)	True Negative



Common Performance Metrics

$$\text{Precision: } \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

$$\text{Recall: } \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

$$\text{Accuracy: } \frac{\text{Correctly classified}}{\text{Total number of cases}}$$

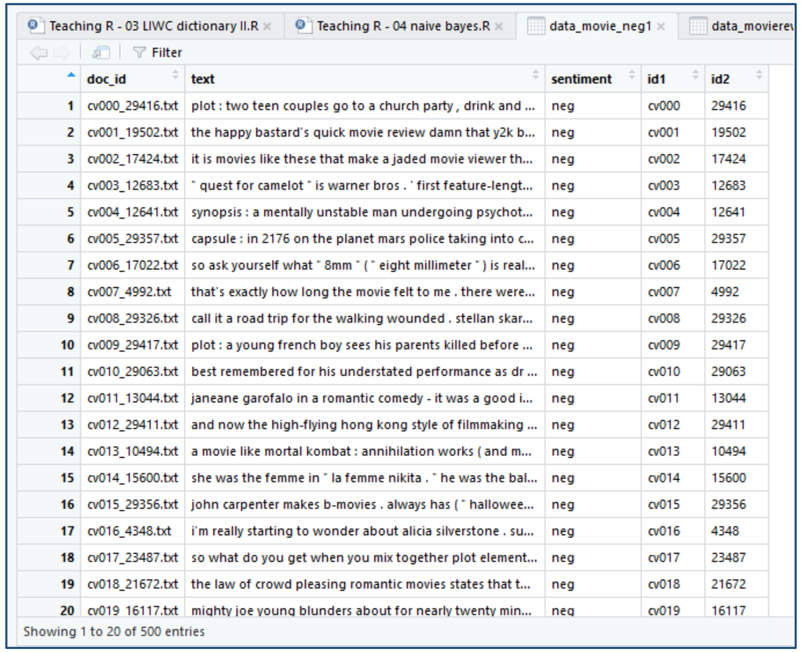
$$F1 = 2\,\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

(the harmonic mean of precision and recall)

- For all: The higher the better!
- But no common cut-offs, depending on the task and context

Naive Bayes Classifier (1/3)

```
# 1. Training set to train the classifier
# 2. Validation/Test set to validate the classifier
# 3. Additional data to use the classifier

# Load the packages
library(quanteda)
library(quanteda.textmodels)
library(caret)
library(openxlsx)

# Convert the corpus of 2000 movie reviews into a dataframe
data_moviereviews <- convert(data_corpus_moviereviews, to="data.frame")

# Split the dataframe
data_movie_neg1 <- data_moviereviews[1:500, ]
data_movie_neg2 <- data_moviereviews[501:1000, ]
data_movie_pos1 <- data_moviereviews[1001:1500, ]
data_movie_pos2 <- data_moviereviews[1501:2000, ]

# Combine the dataframes
data_movie1 <- rbind(data_movie_neg1, data_movie_pos1)
data_movie2_wr <- rbind(data_movie_neg2, data_movie_pos2)

# Remove sentiment rating from the second dataset
data_movie2 = subset(data_movie2_wr, select = -c(sentiment))

# Convert dataframes into two corpi
corp_movies1 <- corpus(data_movie1)
corp_movies2 <- corpus(data_movie2)
```



	doc_id	text	sentiment	id1	id2
1	cv000_29416.txt	plot : two teen couples go to a church party , drink and ...	neg	cv000	29416
2	cv001_19502.txt	the happy bastard's quick movie review damn that y2k b...	neg	cv001	19502
3	cv002_17424.txt	it is movies like these that make a jaded movie viewer th...	neg	cv002	17424
4	cv003_12683.txt	" quest for camelot " is warner bros . ' first feature-lengt...	neg	cv003	12683
5	cv004_12641.txt	synopsis : a mentally unstable man undergoing psychot...	neg	cv004	12641
6	cv005_29357.txt	capsule : in 2176 on the planet mars police taking into c...	neg	cv005	29357
7	cv006_17022.txt	so ask yourself what " 8mm " (" eight millimeter ") is real...	neg	cv006	17022
8	cv007_4992.txt	that's exactly how long the movie felt to me . there were...	neg	cv007	4992
9	cv008_29326.txt	call it a road trip for the walking wounded . stellan skar...	neg	cv008	29326
10	cv009_29417.txt	plot : a young french boy sees his parents killed before ...	neg	cv009	29417
11	cv010_29063.txt	best remembered for his understated performance as dr ...	neg	cv010	29063
12	cv011_13044.txt	janeane garofalo in a romantic comedy - it was a good i...	neg	cv011	13044
13	cv012_29411.txt	and now the high-flying hong kong style of filmmaking ...	neg	cv012	29411
14	cv013_10494.txt	a movie like mortal kombat : annihilation works (and m...	neg	cv013	10494
15	cv014_15600.txt	she was the femme in " la femme nikita . " he was the bal...	neg	cv014	15600
16	cv015_29356.txt	john carpenter makes b-movies . always has (" hallowee...	neg	cv015	29356
17	cv016_4348.txt	i'm really starting to wonder about alicia silverstone . su...	neg	cv016	4348
18	cv017_23487.txt	so what do you get when you mix together plot element...	neg	cv017	23487
19	cv018_21672.txt	the law of crowd pleasing romantic movies states that t...	neg	cv018	21672
20	cv019_16117.txt	mighty joe young blunders about for nearly twenty min...	neg	cv019	16117

Showing 1 to 20 of 500 entries

https://quanteda.io/

Naive Bayes Classifier (2/3)

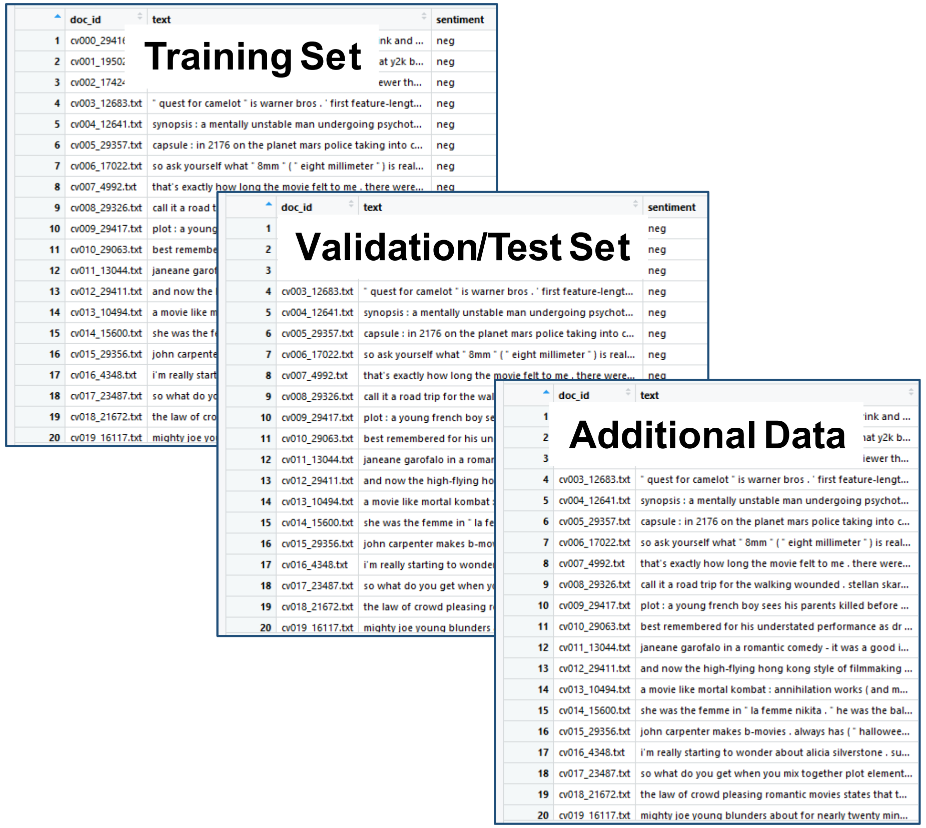
The variable “Sentiment” indicates whether a movie review was classified as positive or negative.

1. **Training set:** We use 500 reviews to build a Naive Bayes classifier
2. **Validation/Test set:** We predict the sentiment for the remaining 500 reviews
3. **Additional data:** We predict the sentiment for the remaining 1000 reviews without rating

Since the first 500 reviews are negative and the remaining 500 reviews are classified as positive in the hand-coded data set, we need to draw a random sample of the documents.

```
# Generate 500 numbers without replacement
id_train <- sample(1:1000, 500, replace = FALSE)
head(id_train, 10)

# Create docvar with ID
corp_movies1$id_numeric <- 1:ndoc(corp_movies1)

# Get training set from corp_movies1
dfmat_training <- corpus_subset(corp_movies1, id_numeric %in% id_train) %>%
                  dfm(remove = stopwords("english"), stem = TRUE)

# Get test set set from corp_movies1 (documents not in id_train)
dfmat_test <- corpus_subset(corp_movies1, !id_numeric %in% id_train) %>%
              dfm(remove = stopwords("english"), stem = TRUE)

# Get additional data from corp_movies2
dfm_add <- dfm(corp_movies2, remove = stopwords("english"), stem = TRUE)
```

➢ Random sampling is always advised!



https://quanteda.io/

Naive Bayes Classifier (3/3)

```
# Train the naive Bayes classifier
tmod_nb <- textmodel_nb(dfmat_training, dfmat_training$sentiment)
summary(tmod_nb)

# Only consider features that occur in the training set and in the other sets
# Make the features identical using dfm_match()
dfmat_matched <- dfm_match(dfmat_test, features = featnames(dfmat_training))
dfm_matched <- dfm_match(dfm_add, features = featnames(dfmat_training))

# Inspect how well the classification worked
actual_class <- dfmat_matched$sentiment
predicted_class <- predict(tmod_nb, newdata = dfmat_matched)
tab_class <- table(actual_class, predicted_class)
tab_class

# Use the confusion matrix to assess the performance of the classification
confusionMatrix(tab_class, mode = "everything")

# Use the trained naive Bayes classifier on a new data set
prediction_add <- predict(tmod_nb, newdata = dfm_matched)

# Add prediction to new data set
corp_movies2$predicted_sentiment <- prediction_add

# Convert the corpus into data frame and inspect it
data_movie_add <- convert(corp_movies2, to="data.frame")
View(data_movie_add)
```

```
            predicted_class
actual_class neg pos
         neg 209  47
         pos  52 192

               Accuracy : 0.802
                 95% CI : (0.7643, 0.8361)
    No Information Rate : 0.522
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.6036

 Mcnemar's Test P-Value : 0.6877

            Sensitivity : 0.8008
            Specificity : 0.8033
         Pos Pred Value : 0.8164
         Neg Pred Value : 0.7869
              Precision : 0.8164
                 Recall : 0.8008
                     F1 : 0.8085
             Prevalence : 0.5220
         Detection Rate : 0.4180
   Detection Prevalence : 0.5120
      Balanced Accuracy : 0.8021
```

Filter

	doc_id	text	id1	id2	predicted_sentiment
1	cv500_10722.txt	you always have to be careful with the first official studi...	cv500	10722	neg
2	cv501_12675.txt	synopsis : easily-angered , chainsmoking architect david ...	cv501	12675	neg
3	cv502_10970.txt	the blues brothers was a wonderful film , a hilarious co...	cv502	10970	neg
4	cv503_11196.txt	michael crichton has had a long career of writing novels...	cv503	11196	neg
5	cv504_29120.txt	american pie 2 is filled with laughs . but they are mostly ...	cv504	29120	pos
6	cv505_12926.txt	well , what are you going to expect ? it's a movie about ...	cv505	12926	neg
7	cv506_17521.txt	this film is extraordinarily horrendous and i'm not going...	cv506	17521	neg

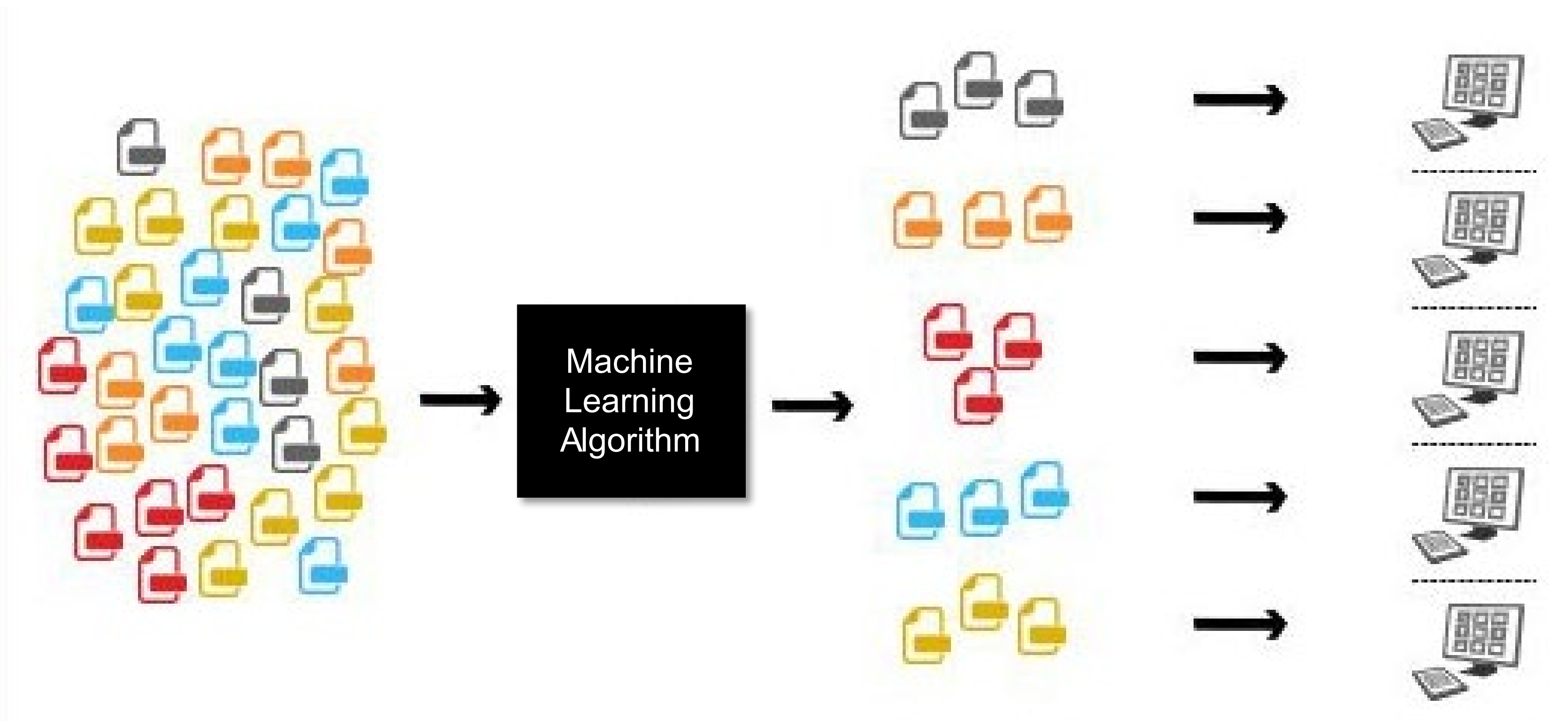
https://quanteda.io/

Exercise with Amazon Reviews of Office Products

- **Implement a Naive Bayes Classifier on Amazon Reviews**
  1. Import the necessary datasets
     - ✓ *df_amazon_train.xlxs* (with labeling)
     - ✓ *df_amazon_add.xlxs* (without labeling)
  2. Generate a training and validation/test set
  3. Train the naive Bayes classifier and inspect its performance
  4. Change properties of the document-feature matrix and inspect the performance
  5. Use the classifier on the new data set and inspect the results

Try to use and adapt the code from the examples!

6) Clustering and Topic Discovery



Raw Data
(unlabeled)

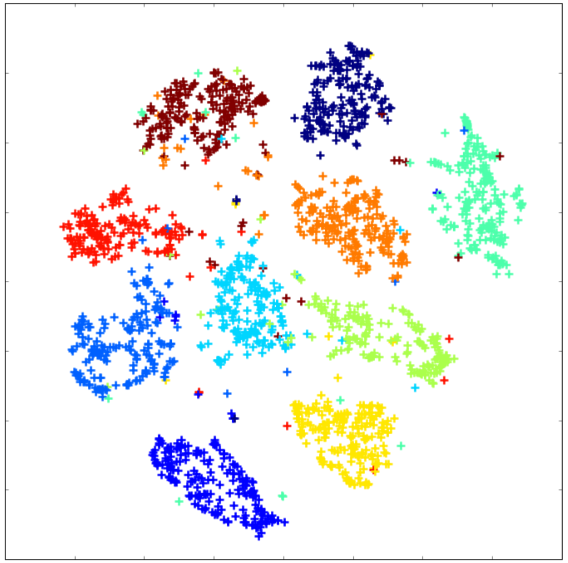
Model
(to be trained)

Automated Cluster
(labeled data)

Manual Review
(Interpretation)

Application of Clustering and Topic Discovery

- Topics models are unsupervised **document classification techniques**
- Topic models **discover the main themes** that pervade a large and otherwise unstructured collection of documents
- By modeling distributions of topics over words and words over documents, topic models identify the **most discriminatory groups of documents** automatically
- Topic modeling algorithms can be applied to **massive collections of documents**
- Topic modeling algorithms can be adapted to **many kinds of data**. E.g., to find patterns in genetic data, images, and social networks…



See for more information:

topicmodels: An R Package for Fitting Topic Models (https://www.jstatsoft.org/article/view/v040i13)

➢ Due to memory requirements *topicmodels* will only work for a reasonably large document-feature matrix!

➢ Use parallel computing, work stations, or the cloud to fit a more complex document-feature matrix

Latent Dirichlet Allocation (LDA)

Latent Dirichlet allocation is one of the most common algorithms for topic modeling guided by two principles:

- **Every document is a mixture of topics.** We imagine that each document may contain words from several topics in particular proportions. For example, in a two-topic model we could say "Document 1 is 90% topic A and 10% topic B, while Document 2 is 30% topic A and 70% topic B."
- **Every topic is a mixture of words.** For example, we could imagine a two-topic model of American news, with one topic for "politics" and one for "entertainment." The most common words in the politics topic might be "president", and "government", while the entertainment topic may be made up of words such as "movies" and "actor". Importantly, words can be shared between topics; a word like "budget" might appear in both equally.

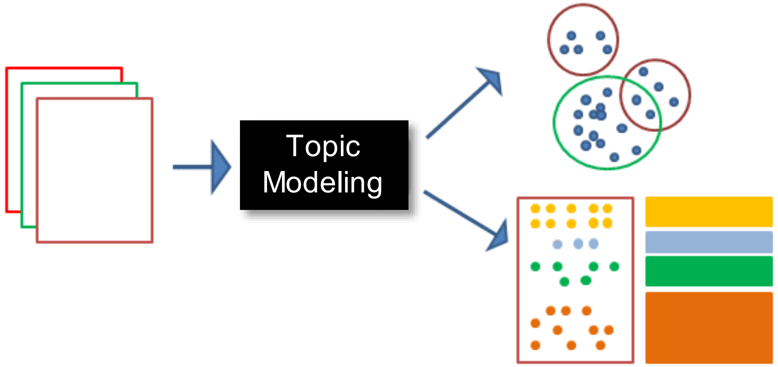
LDA is a mathematical method for **estimating both at the same time**: finding the mixture of words that is associated with each topic, while also determining the mixture of topics that describes each document.

Challenges in Applying LDA:

✓ How many topics (*k*)?

✓ How to interpret (label) the topics?

✓ Should we expect all topics to make sense?



Finding the “Best” Number of Topic Clusters (*k*)

Topic Coherence

Examine the words in topics, decide if they make sense (interpretability of cluster solution)

Quantitative Measures

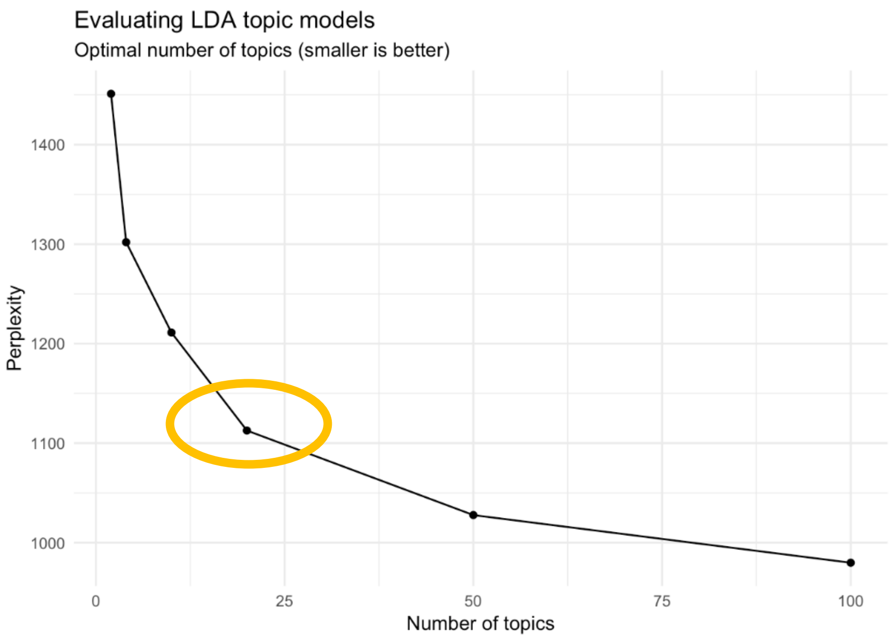
Log-likelihood measures “how plausible model parameters are given the data”

- All values are negative, numerical optimization (search for the largest log-likelihood)
- Can be used to calculate common fit indices
  - ✓ *Akaike Information Criterion* (*AIC = -2/N * LL + 2 * k/N*)
  - ✓ *Bayesian Information Criterion* (*BIC = -2 * LL + log(N) * k*)

Finding the best k

1) Fit the model for several values of *k*
2) Plot the values
3) Pick the one where improvements are small

➢ Similar to “elbow plot” in k-means clustering



Fitting a Topic Model: A Simple Illustration

```r
# Load the packages
library(quanteda)
library(topicmodels)

# Build a corpus
text <- c("Due to bad loans, the bank agreed to pay the fines",
        "If you are late to pay off your loans to the bank, you will face fines",
        "A new restaurant opened in downtown",
        "There is a new restaurant that just opened",
        "How will you pay off the fines for the bank?")

# Create the dfm
dtm <-dfm(text, tolower=TRUE, remove=stopwords("en"),
    remove_punct=TRUE, remove_numbers = TRUE)
dtm

# Fit LDAs with two clusters
LDA_fit_2 <- convert(dtm, to = "topicmodels") %>%
          LDA(k = 2, method="Gibbs")

# Get top five terms per topic
get_terms(LDA_fit_2, 5)

# Get top topic per document
get_topics(LDA_fit_2)
```

```
> dtm
Document-feature matrix of: 5 documents, 14 features (65.7% sparse).
       features
docs     due bad loans bank agreed pay fines late face new
  text1   1   1     1    1      1   1     1    0    0   0
  text2   0   0     1    1      0   1     1    1    1   0
  text3   0   0     0    0      0   0     0    0    0   1
  text4   0   0     0    0      0   0     0    0    0   1
  text5   0   0     0    1      0   1     1    0    0   0
[ reached max_nfeat ... 4 more features ]
>
> # Fit LDAs with two clusters
> LDA_fit_2 <- convert(dtm, to = "topicmodels") %>%
+             LDA(k = 2, method="Gibbs")
>
> # Get top five terms per topic
> get_terms(LDA_fit_2, 5)
     Topic 1  Topic 2
[1,] "bank"   "new"
[2,] "pay"    "restaurant"
[3,] "fines"  "due"
[4,] "opened" "bad"
[5,] "loans"  "loans"
>
> # Get top topic per document
> get_topics(LDA_fit_2)
text1 text2 text3 text4 text5
    1     1     2     2     1
> 
```

Latent Dirichlet Allocation (1/2)

```
# Load additional packages
library(quanteda.textmodels)
library(tidytext)
library(ggplot2)
library(dplyr)

# Create and reduce the dfm
dfm_movie <- dfm(data_corpus_moviereviews,
                 tolower=TRUE, remove=stopwords("en"),
                 remove_punct=TRUE, remove_numbers = TRUE)
dfm_movie <- dfm_trim(dfm_movie, min_termfreq = 50, max_docfreq = 50)

# Check the size of the dfm
dfm_movie

# Fit LDAs with 5 and 10 clusters
LDA_fit_5 <- convert(dfm_movie, to = "topicmodels") %>%
                LDA(k = 5, method="Gibbs")
LDA_fit_10 <- convert(dfm_movie, to = "topicmodels") %>%
                LDA(k = 10, method="Gibbs")

# Get top ten terms per topic
get_terms(LDA_fit_5, 10)
get_terms(LDA_fit_10, 10)
```

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
[1,]	"titanic"	"="	"trek"	"toy"	"chan"
[2,]	"lucas"	"mars"	"godzilla"	"douglas"	"carter"
[3,]	"cage"	"truman"	"simon"	"species"	"vampire"
[4,]	"phantom"	"tarzan"	"vegas"	"troopers"	"julie"
[5,]	"angels"	"carrey"	"jay"	"spice"	"witch"
[6,]	"rising"	"christmas"	"tarantino"	"starship"	"blade"
[7,]	"rose"	"hanks"	"soldiers"	"anderson"	"travolta"
[8,]	"jedi"	"babe"	"spielberg"	"gibson"	"blair"
[9,]	"cruise"	"patch"	"dude"	"mulan"	"hong"
[10,]	"spawn"	"jane"	"affleck"	"horse"	"arnold"

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8	Topic 9	Topic 10
[1,]	"truman"	"godzilla"	"mars"	"anderson"	"toy"	"witch"	"="	"lucas"	"simon"	"vampire"
[2,]	"carrey"	"tarzan"	"titanic"	"reeves"	"julie"	"douglas"	"trek"	"phantom"	"cage"	"blade"
[3,]	"christmas"	"travolta"	"chan"	"babe"	"tarantino"	"blair"	"carter"	"spice"	"kelly"	"vegas"
[4,]	"spielberg"	"apes"	"hong"	"boogie"	"mulan"	"jay"	"arnold"	"tommy"	"princess"	"vampires"
[5,]	"soldiers"	"hanks"	"kong"	"gangster"	"dvd"	"angels"	"troopers"	"jedi"	"kate"	"baldwin"
[6,]	"dude"	"jane"	"species"	"jimmy"	"vincent"	"affleck"	"starship"	"rising"	"austin"	"virus"
[7,]	"gibson"	"japanese"	"martial"	"football"	"ray"	"urban"	"brooks"	"ford"	"annie"	"hunt"
[8,]	"mel"	"beast"	"arts"	"seagal"	"sarah"	"roberts"	"devil"	"helen"	"myers"	"shakespeare"
[9,]	"rocky"	"nuclear"	"damme"	"sandler"	"joan"	"spawn"	"washington"	"larry"	"snake"	"cruise"
[10,]	"ghost"	"jungle"	"ripley"	"porn"	"cusack"	"damon"	"x-files"	"grant"	"campbell"	"las"

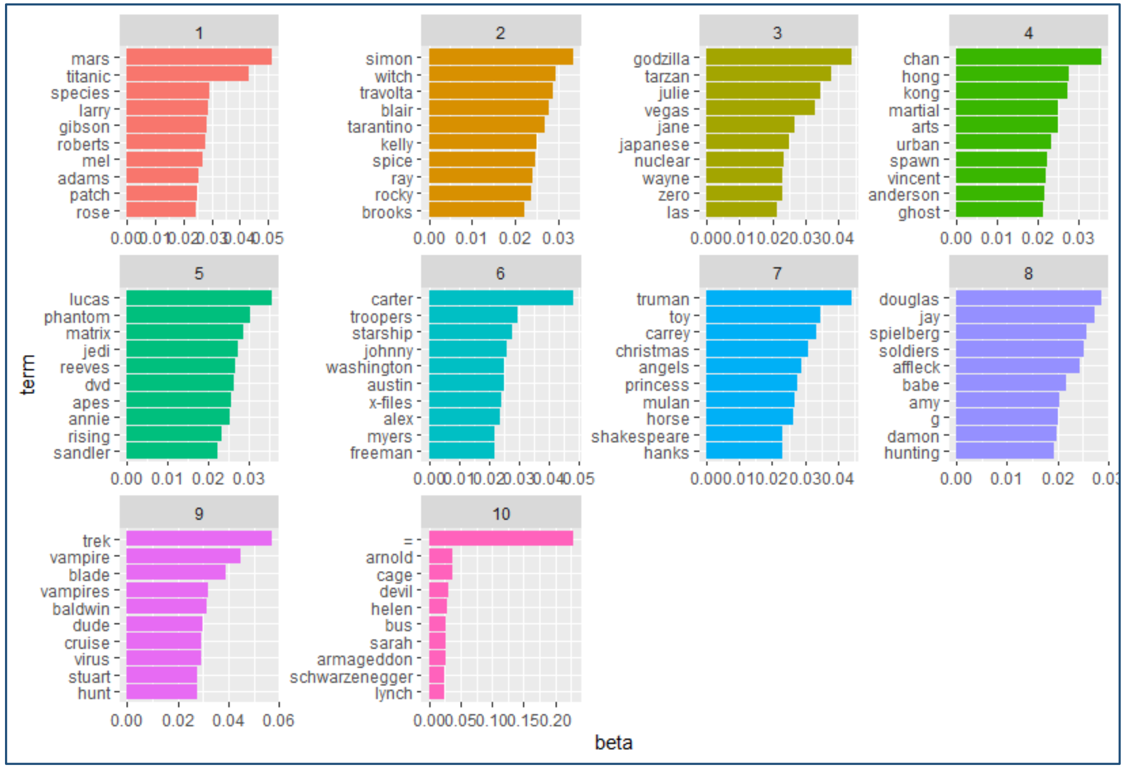
https://quanteda.io/

Latent Dirichlet Allocation (2/2)

```
# Cross validation with log-likelihood
logLik(LDA_fit_5)
logLik(LDA_fit_10)
```

➢ larger log-likelihood is better (i.e., closer to zero)

```
# Extract the per-topic-per-word probabilities
ap_topics <- tidy(LDA_fit_10, matrix = "beta")

# Select the top ten terms per topic
ap_top_terms <- ap_topics %>%
  group_by(topic) %>%
  top_n(10, beta) %>%
  ungroup() %>%
  arrange(topic, -beta)

# Display the top ten terms per topic
ap_top_terms %>%
  mutate(term = reorder_within(term, beta, topic)) %>%
  ggplot(aes(term, beta, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free") +
  coord_flip() +
  scale_x_reordered()
```



Choosing the "best" number of clusters:
➢ Topic coherence - examine the words in topics
➢ Measure model parameters given the data

https://quanteda.io/

Exercise (with Your Own Data)

- **Discover topic clusters in your data**
    1. Create and reduce the document-feature matrix
        - ✓ keep in mind that “topicmodels” will only work for reasonably large corpora
    2. Fit Latent Dirichlet Allocation with different cluster solutions
    3. Choose the “best” cluster solutions

- **Inspect the “best” cluster solution**
    1. Extract the per-topic-per-word probabilities
    2. Display the per-topic-per-word probabilities

Try to use and adapt the code from the examples!

Further Questions? And then: Go out and play!

Further Reading on Automated Text Analysis

Aggarwal, C. C., & Zhai, C. (Eds.). (2012). *Mining text data*. Springer Science & Business Media.

Balducci, B., & Marinova, D. (2018). Unstructured data in marketing. *Journal of the Academy of Marketing Science*, 46(4), 557-590.

Benoit, K., Watanabe, K., Wang, H., Nulty, P., Obeng, A., Müller, S., & Matsuo, A. (2018). *quanteda: An R package for the quantitative analysis of textual data*. Journal of Open Source Software, 3(30), 774.

Berger, J., Humphreys, A., Ludwig, S., Moe, W. W., Netzer, O., & Schweidel, D. A. (2020). Uniting the tribes: Using text for marketing insight. *Journal of Marketing*, 84(1), 1-25.

Feldman, R., & Sanger, J. (2007). *The text mining handbook: advanced approaches in analyzing unstructured data.* Cambridge university press.

Hartmann, J., Huppertz, J., Schamp, C., & Heitmann, M. (2019). Comparing automated text classification methods. *International Journal of Research in Marketing*, 36(1), 20-38.

Kwartler, T. (2017). *Text mining in practice with R*. John Wiley & Sons.

Humphreys, A., & Wang, R. J. H. (2018). Automated text analysis for consumer research. *Journal of Consumer Research*, 44(6), 1274-1306.

McKenny, A. F., Aguinis, H., Short, J. C., & Anglin, A. H. (2018). What doesn't get measured does exist: Improving the accuracy of computer-aided text analysis. *Journal of Management*, 44(7), 2909-2933.

Ordenes, F. V., & Zhang, S. (2019). From words to pixels: text and image mining methods for service research. *Journal of Service Management*.

Silge, J., & Robinson, D. (2017). *Text mining with R: A tidy approach*. O'Reilly Media.

Tausczik, Y. R., & Pennebaker, J. W. (2010). The psychological meaning of words: LIWC and computerized text analysis methods. *Journal of language and social psychology*, 29(1), 24-54.

Further Reading on Web Scraping

Articles and Books on Web Scraping

Aydin, O. (2018). *R Web Scraping Quick Start Guide: Techniques and tools to crawl and scrape data from websites*. Packt Publishing Ltd.

Bradley, A., & James, R. J. (2019). Web scraping using R. *Advances in Methods and Practices in Psychological Science*, 2(3), 264-270.

Landers, R. N., Brusso, R. C., Cavanaugh, K. J., & Collmus, A. B. (2016). A primer on theory-driven web scraping: Automatic extraction of big data from the Internet for use in psychological research. *Psychological methods*, 21(4), 475.

Munzert, S., Rubba, C., Meißner, P., & Nyhuis, D. (2014). *Automated data collection with R: A practical guide to web scraping and text mining*. John Wiley & Sons.

Tutorials on Web Scraping with R

https://www.freecodecamp.org/news/an-introduction-to-web-scraping-using-r-40284110c848/

https://www.datacamp.com/community/tutorials/r-web-scraping-rvest

https://towardsdatascience.com/tidy-web-scraping-in-r-tutorial-and-resources-ac9f72b4fe47

https://www.r-bloggers.com/2019/04/practical-introduction-to-web-scraping-in-r/

https://www.scrapingbee.com/blog/web-scraping-r/

R Packages for Web Scraping

httr: Tools for Working with URLs and HTTP (https://cran.r-project.org/web/packages/httr/)

rvest: Easily Harvest (Scrape) Web Pages (https://cran.r-project.org/web/packages/rvest/)

RSelenium: R Bindings for 'Selenium WebDriver‘ (https://cran.r-project.org/web/packages/RSelenium/)

Packages for Data Collection in R

twitteR is an R package which provides access to the Twitter API. Most functionality of the API is supported, with a bias towards API calls that are more useful in data analysis as opposed to daily interaction: https://www.rdocumentation.org/packages/twitteR/versions/1.1.9

tuber allows you to Get comments posted on YouTube videos, information on how many times a video has been liked, search for videos with particular content, and much more. You can also scrape captions from videos: https://www.rdocumentation.org/packages/tuber/versions/0.9.9

edgar is a tool for the U.S. SEC EDGAR retrieval and parsing of corporate filings. The EDGAR database automated system collects all the different necessary filings and makes it publicly available. This package facilitates retrieving, storing, searching, and parsing of all the available filings on the EDGAR server: https://cran.r-project.org/web/packages/edgar/index.html

Scraping Amazon Reviews in R: https://martinctc.github.io/blog/vignette-scraping-amazon-reviews-in-r/
Good tutorial but requires a bit more coding...

- Note: As a data collection activity, web-scraping may have legal implications depending on your country. For most countries, as a general rule you can legally web-scrape anything out there that is in the public domain, but it is recommended that you obtain the site owner's permission if you are reporting on the data or using the data for commercial use!

Contact

Prof. Dr. Dennis Herhausen

Associate Professor of Marketing
KEDGE Business School
Domaine de Luminy
Rue Antoine Bourdelle
13009 Marseille, France
dennis.herhausen@kedgebs.com

To connect: LinkedIn