Choosing the right Method for the Task: **Bottom-Up and Top-Down Approaches to Automated Content Analysis**

Damian Trilling

d.c.trilling@uva.nl
@damian0604
www.damiantrilling.net

Department of Communication Science
Universiteit van Amsterdam

Summer School Computational Social Science
July 30th – August 4th, 2018, Los Angeles, USA

If you are interested in the whole course on which this talk is based, please find

- all slides of an 8-weeks course
- a 130 page PDF teaching you the basics of Python and how to code up all techniques discussed
- some additional Jupyter Notebooks

at:
`https://github.com/damian0604/bdaca/`

Today

Types of Automated Content Analysis

Let's start with reflecting on the role of theory in CSS...

Three types of epistemologies with regard to Big Data:

1. (Reborn) empiricism: purely inductive, correlation is enough

Kitchin, R. (2014). Big Data, new epistemologies and paradigm shifts. *Big Data & Society, 1*(1), 1–12. doi:10.1177/2053951714528481

Let's start with reflecting on the role of theory in CSS...

Three types of epistemologies with regard to Big Data:

1. (Reborn) empiricism: purely inductive, correlation is enough
2. Data-driven science: knowledge discovery guided by theory

Kitchin, R. (2014). Big Data, new epistemologies and paradigm shifts. *Big Data & Society, 1*(1), 1–12. doi:10.1177/2053951714528481

Let's start with reflecting on the role of theory in CSS. . .

Three types of epistemologies with regard to Big Data:

1. (Reborn) empiricism: purely inductive, correlation is enough
2. Data-driven science: knowledge discovery guided by theory
3. Computational social science and digital humanities: employ Big Data research within existing epistemologies
   - DH: descriptive statistics, visualizations
   - CSS: prediction and simulation

Kitchin, R. (2014). Big Data, new epistemologies and paradigm shifts. *Big Data & Society, 1*(1), 1–12. doi:10.1177/2053951714528481

Things to rember

Before designing an Automated Content Analysis, decide whether you

- **A** want to inductively explore data or find things you did not define in advance;
  or
- **B** have theoretically well-defined concepts you want to operationalize and measure.

Things to rember

Before designing an Automated Content Analysis, decide whether you

- **A** want to inductively explore data or find things you did not define in advance;
  or
- **B** have theoretically well-defined concepts you want to operationalize and measure.

A is a *bottom up* approach, B is *top down*.

Automated Content Analysis

	Methodological approach		
	Counting and Dictionary	*Supervised Machine Learning*	*Unsupervised Machine Learning*
Typical research interests and content features	visibility analysis sentiment analysis subjectivity analysis	frames topics gender bias	frames topics
Common statistical procedures	string comparisons counting	support vector machines naive Bayes	principal component analysis cluster analysis latent dirichlet allocation semantic network analysis

deductive → **inductive**

Boumans, J.W., & Trilling, D. (2016). Taking stock of the toolkit: An overview of relevant automated content analysis approaches and techniques for digital journalism scholars. *Digital Journalism*, *4*, 1. 8–23.

Examples

	bottom-up	top-down
event detection	unusual spikes and (word) distributions	finding pre-defined events
topics	words co-occurance patterns; unknown which topics exist (number may be known/set)	finding pre-defined topics (well-defined and knwon)
frames	same as above	same as above

Examples

	bottom-up	top-down
event detection	You want to see when people suddenly start tweeting about something new	you want to know when "your" topic receives a lot of attention
topics	You want to know what people write about on review sites	You want to know whether these reviews are (a) complaints, (b) praise, or (c) neutral
frames	You want to know which different topic-specific frames are used to discuss nuclear power plants	You want to know whether your data contains a human-interest frame, an economic-consequences frame, and/or a conflict frame.

Bottom-up approaches

Counting word frequencies

- Simply counting the most frequently occurring words
- Often visualized as word clouds
- When calculated for different corpora, most characteristic words can be easily determined by calculating the log-likelihood

Example of an application:
Boukes, M., & Trilling, D. (2017). Political relevance in the eye of the beholder: Determining the substantiveness of TV shows and political debates with Twitter data. *First Monday, 22*(4).

kissinger
hrc
#imwithher
#debatewithbernie
henry
#pbsdebate
money
street
women
government
pbs
obama
shes
criminal
campaign
americans
white
raise
reform
donors
racism
system
secretary
woman
havent
billionaires
healthcare
influence
responsible
foreign
wall
#berniesandwiches
clintonsanders
ran
limit
spanish
scalia
reagan
#cbsdebate
delay
court
cbs
supreme
moderator
#scalia
constitution
booed
website
george
#cruzcrew
scotus
justices
booing
nominate
crowd
john
confirmed
liar
ronald
deflated
amnesty
brother
rally
conservatives
boos
facts
silence
scalias
stuttering
hes
1000
0
1000
2000
3000
Loglikelihood

Word counts

pro

easy to do and easy to understand

con

doesn't tell much; all context is lost and it's hard to guess what a single word "means"[1]

⇒ Specific use cases aside, mostly only a useful first explorative step, but probably not your final analysis.

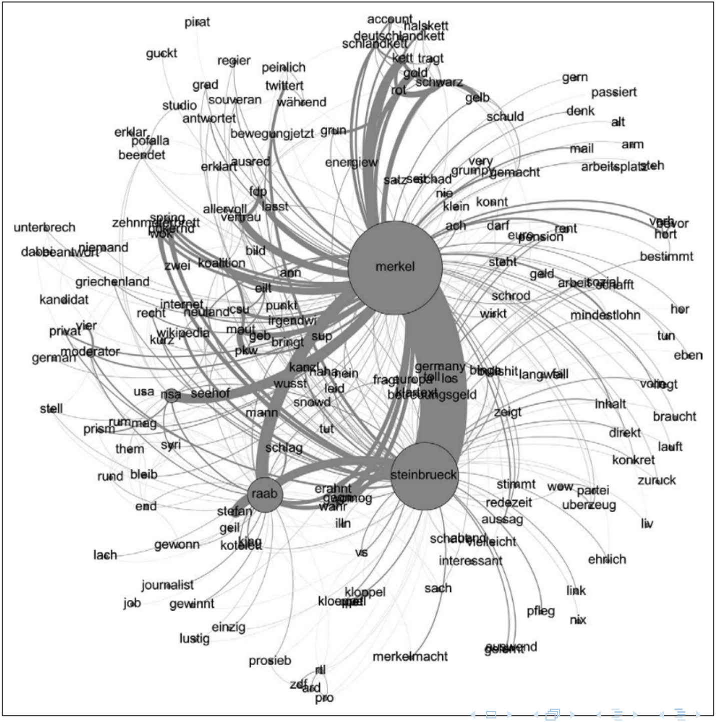
[1] Can be partly improved by including bigrams (ngrams), i.e. adjacent words

Counting word co-occurrences

- Count which words co-occur in the same sentence/paragraph/text
- Often visualized as networks ("semantic map")
  Node size ~ word frequency
  Edge weight ~ number of co-occurrences

Examples of applications:

Hellsten, I., Dawson, J., & Leydesdorff, L. (2010). Implicit media frames: Automated analysis of public debate on artificial sweeteners. *Public Understanding of Science, 19*(5), 590–608.
Trilling, D. (2015). Two Different Debates? Investigating the Relationship Between a Political Debate on TV and Simultaneous Comments on Twitter. *Social Science Computer Review, 33*(3), 259–276.



pirat
guckt
regier
peinlich
grad
twittert
studio
souveran
während
antwortet
erklar
pofalla
beendet
bewegungjetzt
grun
erklart
ausred
energiew
fdp
lasst
vertrau
unterbrech
niemand
bild
zwei
koalition
ann
griechenland
eilt
kandidat
internet
recht
neuland
csu
punkt
irgendwi
wikipedia
maut
geb
bringt
sup
kurz
pkw
privat
vier
moderator
german
kanzl
haha
nein
wusst
leid
usa
nsa
seehof
snowd
stell
mann
prism
tut
them
syri
schlag
rund
bleib
raab
end
stefan
geil
gewonn
lach
king
journalist
job
gewinnt
einzig
lustig
prosieb
rtl
zdf
ard
pro
merkel
steinbrueck
account
halskett
deutschlandkett
schlandkett
kett
tragt
gold
schwarz
rot
gelb
gern
passiert
schuld
denk
alt
mail
arm
arbeitsplatz
very
grumpy
gemacht
nie
klein
konnt
ach
darf
rent
pension
bevor
hort
bestimmt
steht
geld
schafft
schrod
wirkt
mindestlohn
hor
tun
eben
germany
europe
los
frag
betreuungsgeld
langweil
inhalt
zeigt
braucht
direkt
lauft
konkret
zuruck
stimmt
wow
partei
redezeit
uberzeug
aussag
liv
erahnt
wahr
illn
vielleicht
interessant
ehrlich
vs
link
sach
pfleg
kloppel
nix
merkelmacht

Counting word co-occurrences

pro

Visualizations to show “clusters” (topics, frames)

con

sensitive to preprocessing choices, cutoff values, etc.; unclear what’s the “right” visualization; hard to “substantiate” findings

$\Rightarrow$ Can be a good way to summarize texts visually, but also a bit out-dated and superseded by, e.g., topic models (more later)

Even though some still prefer it over topic models, see
Leydesdorff, L., & Nerghes, A. (2017). Co-word maps and topic modeling: A comparison using small and medium-sized corpora (N $<$ 1,000). *Journal of the Association for Information Science and Technology, 68*(4), 1024–1035.

Things to rember

If you choose for a bottom-up approach, you can start by counting word frequencies and word co-occurrences. These allow for easy-to-understand visualizations (word clouds, co-occurrence networks).

Things to rember

If you choose for a bottom-up approach, you can start by counting word frequencies and word co-occurrences. These allow for easy-to-understand visualizations (word clouds, co-occurrence networks).

But let's move on to more advanced approaches.

inductive and bottom-up:
unsupervised machine learning

inductive and bottom-up:
unsupervised machine learning

(something you aready did in your Bachelor – no kidding.)

Some terminology

Supervised machine learning

You have a dataset with both predictor and outcome (independent and dependent variables; features and labels) — a *labeled* dataset.

Some terminology

Supervised machine learning

You have a dataset with both predictor and outcome (independent and dependent variables; features and labels) — a *labeled* dataset. Think of regression: You measured `x1`, `x2`, `x3` and you want to predict `y`, which you also measured

Some terminology

Supervised machine learning

You have a dataset with both predictor and outcome (independent and dependent variables; features and labels) — a *labeled* dataset.

Unsupervised machine learning

You have no labels.

Some terminology

Supervised machine learning

You have a dataset with both predictor and outcome (independent and dependent variables; features and labels) — a *labeled* dataset.

Unsupervised machine learning

You have no labels. (You did not measure y)

Some terminology

Unsupervised machine learning

You have no labels.

Again, you already know some techniques to find out how `x1, x2,...x_i` co-occur from other courses:

- Principal Component Analysis (PCA)
- Cluster analysis
- ...

Principal Component Analysis? How does *that* fit in here?

Principal Component Analysis? How does *that* fit in here?

In fact, PCA is used everywhere, even in image compression

Principal Component Analysis? How does *that* fit in here?

PCA in ACA

- Find out what word cooccur (inductive frame analysis)
- Basically, transform each document in a vector of word frequencies and do a PCA

A so-called term-document-matrix

```
      w1,w2,w3,w4,w5,w6 ...
text1, 2, 0, 0, 1, 2, 3 ...
text2, 0, 0, 1, 2, 3, 4 ...
text3, 9, 0, 1, 1, 0, 0 ...
...
```

A so-called term-document-matrix

```
       w1,w2,w3,w4,w5,w6 ...
text1, 2, 0, 0, 1, 2, 3 ...
text2, 0, 0, 1, 2, 3, 4 ...
text3, 9, 0, 1, 1, 0, 0 ...
...
```

These can be simple counts, but also more advanced metrics, like tf-idf scores (where you weigh the frequency by the number of documents in which it occurs), cosine distances, etc.

Interpreting the loadings

Usually, the components (factors) are then interpreted as topics or frames, based on the highest-loading words.

Examples of applications:

Hellsten, I., Dawson, J., & Leydesdorff, L. (2010). Implicit media frames: Automated analysis of public debate on artificial sweeteners. *Public Understanding of Science, 19*(5), 590–608.

Van der Meer, G. L. A., Verhoeven, P., Beentjes, H., & Vliegenthart, R. (2014). When frames align: The interplay between PR, news media, and the public in times of crisis. *Public Relations Review, 40*(5), 751–761.

. . . but there are problems

pro

given a term-document matrix, easy to do with any tool; familiar to social scientists

con

probably extremely skewed distributions; some problematic assumptions: does the goal of PCA, to find a solution in which one word loads on *one* component match real life, where a word can belong to several topics or frames?

Things to rember

If we represent each document by the frequency the words in it, we can run familiar data reduction techniques like PCA to find topics/frames[1].

[1] or cluster analysis to group similar documents

Things to rember

If we represent each document by the frequency the words in it, we can run familiar data reduction techniques like PCA to find topics/frames[1].

[1] or cluster analysis to group similar documents
(I won't discuss a different use of PCA here: using it as a first step to reduce the number of *features* as input for future analyses)

Enter **topic modeling with Latent Dirichlet Allocation (LDA)**

LDA, what's that?

No mathematical details here, but the general idea

- There are k topics, $T_1 \ldots T_k$
- Each document D_i consists of a mixture of these topics, e.g.$80\% T_1, 15\% T_2, 0\% T_3, \ldots 5\% T_k$
- On the next level, each topic consists of a specific probability distribution of words
- Thus, based on the frequencies of words in D_i, one can infer its distribution of topics
- Note that LDA (like PCA) is a Bag-of-Words (BOW) approach

Doing a LDA in Python

You can use gensim (Řehůřek & Sojka, 2010) for this.
Let us assume you have a list of lists of words (!) called `texts`:

```
articles=['The tax deficit is higher than expected. This said xxx ...',
    'Germany won the World Cup. After a']
texts=[art.split() for art in articles]
```

which looks like this:

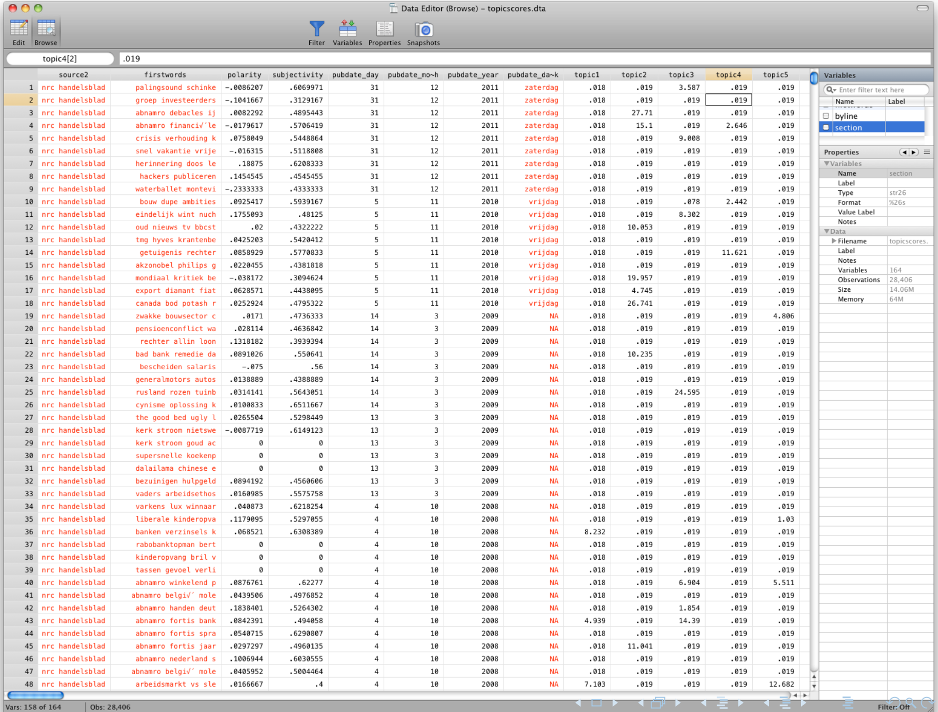
```
[['The', 'tax', 'deficit', 'is', 'higher', 'than', 'expected.', 'This',
    'said', 'xxx', '...'], ['Germany', 'won', 'the', 'World', 'Cup.', '
    After', 'a']]
```

Řehůřek, R., & Sojka, P. (2010). Software framework for topic modelling with large corpora. *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pp. 45–50. Valletta, Malta: ELRA.

```
from gensim import corpora, models

NTOPICS = 100
LDAOUTPUTFILE="topicscores.tsv"

# Create a BOW represenation of the texts
id2word = corpora.Dictionary(texts)
mm =[id2word.doc2bow(text) for text in texts]

# Train the LDA models.
mylda = models.ldamodel.LdaModel(corpus=mm, id2word=id2word, num_topics=
    NTOPICS, alpha="auto")

# Print the topics.
for top in mylda.print_topics(num_topics=NTOPICS, num_words=5):
  print ("\n",top)

print ("\nFor further analysis, a dataset with the topic score for each
    document is saved to",LDAOUTPUTFILE)

scoresperdoc=mylda.inference(mm)

with open(LDAOUTPUTFILE,"w",encoding="utf-8") as fo:
  for row in scoresperdoc[0]:
    fo.write("\t".join(["{:0.3f}".format(score) for score in row]))
fo.write("\n")
```

Output: Topics (below) & topic scores (next slide)

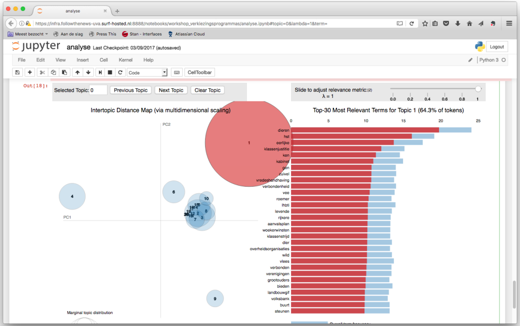
```
0.069*fusie + 0.058*brussel + 0.045*europesecommissie + 0.036*europese + 0.023*overname
0.109*bank + 0.066*britse + 0.041*regering + 0.035*financien + 0.033*minister
0.114*nederlandse + 0.106*nederland + 0.070*bedrijven + 0.042*rusland + 0.038*russische
0.093*nederlandsespoorwegen + 0.074*den + 0.036*jaar + 0.029*onderzoek + 0.027*raad
0.099*banen + 0.045*jaar + 0.045*productie + 0.036*ton + 0.029*aantal
0.041*grote + 0.038*bedrijven + 0.027*ondernemers + 0.023*goed + 0.015*jaar
0.108*werknemers + 0.037*jongeren + 0.035*werkgevers + 0.029*jaar + 0.025*werk
0.171*bank + 0.122* + 0.041*klanten + 0.035*verzekeraar + 0.028*euro
0.162*banken + 0.055*bank + 0.039*centrale + 0.027*leningen + 0.024*financiele
0.052*post + 0.042*media + 0.038*nieuwe + 0.034*netwerk + 0.025*personeel
...
```



	source2	firstwords	polarity	subjectivity	pubdate_day	pubdate_mo~h	pubdate_year	pubdate_da~k	topic1	topic2	topic3	topic4	topic5
1	nrc handelsblad	palingsound schinke	-.0086207	.6069971	31	12	2011	zaterdag	.018	.019	3.587	.019	.019
2	nrc handelsblad	groep investeerders	-.1041667	.3129167	31	12	2011	zaterdag	.018	.019	.019	.019	.019
3	nrc handelsblad	abnamro debacles ij	.0082292	.4895443	31	12	2011	zaterdag	.018	27.71	.019	.019	.019
4	nrc handelsblad	abnamro financi√´le	-.0179617	.5706419	31	12	2011	zaterdag	.018	15.1	.019	2.646	.019
5	nrc handelsblad	crisis verhouding k	.0758049	.5448864	31	12	2011	zaterdag	.018	.019	9.008	.019	.019
6	nrc handelsblad	snel vakantie vrije	-.016315	.5118808	31	12	2011	zaterdag	.018	.019	.019	.019	.019
7	nrc handelsblad	herinnering doos le	.18875	.6208333	31	12	2011	zaterdag	.018	.019	.019	.019	.019
8	nrc handelsblad	hackers publiceren	.1454545	.4545455	31	12	2011	zaterdag	.018	.019	.019	.019	.019
9	nrc handelsblad	waterballet montevi	-.2333333	.4333333	31	12	2011	zaterdag	.018	.019	.019	.019	.019
10	nrc handelsblad	bouw dupe ambities	.0925417	.5939167	5	11	2010	vrijdag	.018	.019	.078	2.442	.019
11	nrc handelsblad	eindelijk wint nuch	.1755093	.48125	5	11	2010	vrijdag	.018	.019	8.302	.019	.019
12	nrc handelsblad	oud nieuws tv bbcst	.02	.4322222	5	11	2010	vrijdag	.018	10.053	.019	.019	.019
13	nrc handelsblad	tmg hyves krantenbe	.0425203	.5420412	5	11	2010	vrijdag	.018	.019	.019	.019	.019
14	nrc handelsblad	getuigenis rechter	.0858929	.5770833	5	11	2010	vrijdag	.018	.019	.019	11.621	.019
15	nrc handelsblad	akzonobel philips g	.0220455	.4381818	5	11	2010	vrijdag	.018	.019	.019	.019	.019
16	nrc handelsblad	mondiaal kritiek be	-.038172	.3094624	5	11	2010	vrijdag	.018	19.957	.019	.019	.019
17	nrc handelsblad	export diamant fiat	.0628571	.4438095	5	11	2010	vrijdag	.018	4.745	.019	.019	.019
18	nrc handelsblad	canada bod potash r	.0252924	.4795322	5	11	2010	vrijdag	.018	26.741	.019	.019	.019
19	nrc handelsblad	zwakke bouwsector c	.0171	.4736333	14	3	2009	NA	.018	.019	.019	.019	4.806
20	nrc handelsblad	pensioenconflict wa	.028114	.4636842	14	3	2009	NA	.018	.019	.019	.019	.019
21	nrc handelsblad	rechter allin loon	.1318182	.3939394	14	3	2009	NA	.018	.019	.019	.019	.019
22	nrc handelsblad	bad bank remedie da	.0891026	.550641	14	3	2009	NA	.018	10.235	.019	.019	.019
23	nrc handelsblad	bescheiden salaris	-.075	.56	14	3	2009	NA	.018	.019	.019	.019	.019
24	nrc handelsblad	generalmotors autos	.0138889	.4388889	14	3	2009	NA	.018	.019	.019	.019	.019
25	nrc handelsblad	rusland rozen tuinb	.0314141	.5643051	14	3	2009	NA	.018	.019	24.595	.019	.019
26	nrc handelsblad	cynisme oplossing k	.0100833	.6511667	14	3	2009	NA	.018	.019	.019	.019	.019
27	nrc handelsblad	the good bed ugly l	.0265504	.5298449	13	3	2009	NA	.018	.019	.019	.019	.019
28	nrc handelsblad	kerk stroom nietswe	-.0087719	.6149123	13	3	2009	NA	.018	.019	.019	.019	.019
29	nrc handelsblad	kerk stroom goud ac	0	0	13	3	2009	NA	.018	.019	.019	.019	.019
30	nrc handelsblad	supersnelle koekenp	0	0	13	3	2009	NA	.018	.019	.019	.019	.019
31	nrc handelsblad	dalailama chinese e	0	0	13	3	2009	NA	.018	.019	.019	.019	.019
32	nrc handelsblad	bezuinigen hulpgeld	.0894192	.4560606	13	3	2009	NA	.018	.019	.019	.019	.019
33	nrc handelsblad	vaders arbeidsethos	.0160985	.5575758	13	3	2009	NA	.018	.019	.019	.019	.019
34	nrc handelsblad	varkens lux winnaar	.040873	.6218254	4	10	2008	NA	.018	.019	.019	.019	.019
35	nrc handelsblad	liberale kinderopva	.1179095	.5297055	4	10	2008	NA	.018	.019	.019	.019	1.03
36	nrc handelsblad	banken verzinsels k	.068521	.6308389	4	10	2008	NA	8.232	.019	.019	.019	.019
37	nrc handelsblad	rabobanktopman bert	0	0	4	10	2008	NA	.018	.019	.019	.019	.019
38	nrc handelsblad	kinderopvang bril v	0	0	4	10	2008	NA	.018	.019	.019	.019	.019
39	nrc handelsblad	tassen gevoel verli	0	0	4	10	2008	NA	.018	.019	.019	.019	.019
40	nrc handelsblad	abnamro winkelend p	.0876761	.62277	4	10	2008	NA	.018	.019	6.904	.019	5.511
41	nrc handelsblad	abnamro belgi√´ mole	.0439506	.4976852	4	10	2008	NA	.018	.019	.019	.019	.019
42	nrc handelsblad	abnamro handen deut	.1838401	.5264302	4	10	2008	NA	.018	.019	1.854	.019	.019
43	nrc handelsblad	abnamro fortis bank	.0842391	.494058	4	10	2008	NA	4.939	.019	14.39	.019	.019
44	nrc handelsblad	abnamro fortis spra	.0540715	.6290807	4	10	2008	NA	.018	.019	.019	.019	.019
45	nrc handelsblad	abnamro fortis jaar	.0297297	.4960135	4	10	2008	NA	.018	11.041	.019	.019	.019
46	nrc handelsblad	abnamro nederland s	.1006944	.6030555	4	10	2008	NA	.018	.019	.019	.019	.019
47	nrc handelsblad	abnamro belgi√´ mole	.0405952	.5004464	4	10	2008	NA	.018	.019	.019	.019	.019
48	nrc handelsblad	arbeidsmarkt vs sle	.0166667	.4	4	10	2008	NA	7.103	.019	.019	.019	12.682

Visualization with pyldavis

```
import pyLDAvis
import pyLDAvis.gensim
# first estiate gensim model, then:
vis_data = pyLDAvis.gensim.prepare(mylda,mm,id2word)
pyLDAvis.display(vis_data)
```



Things to rember

Topic models can be a good way to analyze your data if you cannot specify in advance *what* the topics are; however, even mathematically well-fitting models may be hard to interpret

Other techniques

Cluster analysis

- k-means
- hierarchical clustering (e.g., Ward's method)

Improved topic models that take into accounts co-variates and nested data structures

- Author-topic models
- Structured topic models (STM)

Also have a look at the websites of gensim and scikit-learn!

Top-down approaches

Dictionaries and regular expressions

- Use a (manually compiled) list of words (dictionary) and search for them
- For example, look for “stock exchange”, “exchange rate”, “closing price” . . . to find economic news
- Better: *regular expressions* to look for patterns, e.g. `[Ee]conom.?\w`
- There are pre-defined lists for many applications

Dictionaries and regular expressions

pro

easy to do and easy to understand

con

The more latent the construct, the lower the accuracy
Many false positives when using comprehensive lists, many false negatives when using short lists

⇒ The technique of choice if you try to measure something that is unambigous (how often is party X mentioned?). Largely outdated when it comes to more subtle concepts like topics or frames.

Things to rember

Regular expressions are a very powerful way of describing patterns in strings, and the technique of choice when looking for manifest and well-defined concepts (party names, company names) or things that are always formatted in a specific way (numbers, dates, words with specific characters in it)

$\Rightarrow$ You can formulate a rule? Use regular expressions! You cannot? Then enters...

predefined categories, but no predefined rules:
supervised machine learning

predefined categories, but no predefined rules:
supervised machine learning

(something you aready did in your Bachelor – no kidding.)

Recap: supervised vs. unsupervised

Unsupervised

- No manually coded data
- We want to identify patterns or to make groups of most similar cases

Recap: supervised vs. unsupervised

Unsupervised

- No manually coded data
- We want to identify patterns or to make groups of most similar cases

Example: We have a dataset of Facebook-massages on an organizations' page. We use clustering to group them and later interpret these clusters (e.g., as complaints, questions, praise, . . .)

Recap: supervised vs. unsupervised

Unsupervised

- No manually coded data
- We want to identify patterns or to make groups of most similar cases

Example: We have a dataset of Facebook-massages on an organizations' page. We use clustering to group them and later interpret these clusters (e.g., as complaints, questions, praise, . . .)

Supervised

- We code a small dataset by hand and use it to "train" a machine
- The machine codes the rest

Recap: supervised vs. unsupervised

Unsupervised

- No manually coded data
- We want to identify patterns or to make groups of most similar cases

Example: We have a dataset of Facebook-massages on an organizations' page. We use clustering to group them and later interpret these clusters (e.g., as complaints, questions, praise, . . .)

Supervised

- We code a small dataset by hand and use it to "train" a machine
- The machine codes the rest

Example: We have 2,000 of these messages grouped into such categories by human coders. We then use this data to group all remaining messages as well.

You have done it before!

You have done it before!

Regression

You have done it before!

Regression

1. Based on your data, you estimate some regression equation $y_i = \alpha + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i$

You have done it before!

Regression

1. Based on your data, you estimate some regression equation $y_i = \alpha + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i$
2. Even if you have some *new unseen data*, you can estimate your expected outcome $\hat{y}$!

You have done it before!

Regression

1. Based on your data, you estimate some regression equation $y_i = \alpha + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i$
2. Even if you have some *new unseen data*, you can estimate your expected outcome $\hat{y}$!
3. Example: You estimated a regression equation where y is newspaper reading in days/week:
   $y = -.8 + .4 \times man + .08 \times age$

You have done it before!

Regression

1. Based on your data, you estimate some regression equation $y_i = \alpha + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i$
2. Even if you have some *new unseen data*, you can estimate your expected outcome $\hat{y}$!
3. Example: You estimated a regression equation where y is newspaper reading in days/week: $y = -.8 + .4 \times man + .08 \times age$
4. You could now calculate $\hat{y}$ for a man of 20 years and a woman of 40 years – *even if no such person exists in your dataset*: $\hat{y}_{man20} = -.8 + .4 \times 1 + .08 \times 20 = 1.2$ $\hat{y}_{woman40} = -.8 + .4 \times 0 + .08 \times 40 = 2.4$

If we use such a model for prediction, then **this is Supervised Machine Learning!**

. . . but. . .

- We will only use *half* (or another fraction) of our data to estimate the model, so that we can use the other half to check if our predictions match the manual coding ("labeled data","annotated data" in SML-lingo)

. . . but. . .

- We will only use *half* (or another fraction) of our data to estimate the model, so that we can use the other half to check if our predictions match the manual coding ("labeled data","annotated data" in SML-lingo)
  - e.g., 2000 labeled cases, 1000 for training, 1000 for testing — if successful, run on 100,000 unlabeled cases

. . . but. . .

- We will only use *half* (or another fraction) of our data to estimate the model, so that we can use the other half to check if our predictions match the manual coding ("labeled data","annotated data" in SML-lingo)
  - e.g., 2000 labeled cases, 1000 for training, 1000 for testing — if successful, run on 100,000 unlabeled cases
- We use many more independent variables ("features")

. . . but. . .

- We will only use *half* (or another fraction) of our data to estimate the model, so that we can use the other half to check if our predictions match the manual coding ("labeled data","annotated data" in SML-lingo)
  - e.g., 2000 labeled cases, 1000 for training, 1000 for testing — if successful, run on 100,000 unlabeled cases
- We use many more independent variables ("features")
- Typically, IVs are word frequencies (often weighted, e.g. tf×idf) (⇒BOW-representation)

Applications

Applications

In other fields

A lot of different applications

- from recognizing hand-written characters to recommendation systems

Applications

In other fields

A lot of different applications

- from recognizing hand-written characters to recommendation systems

In our field

It starts to get popular to measure latent variables

- frames
- topics

SML to code frames and topics

Some work by Burscher and colleagues

- Humans can code generic frames (human-interest, economic, . . .)
- Humans can code topics from a pre-defined list

SML to code frames and topics

Some work by Burscher and colleagues

- Humans can code generic frames (human-interest, economic, . . .)
- Humans can code topics from a pre-defined list
- **But it is very hard to formulate an explicit rule**
(as in: code as 'Human Interest' if regular expression R is matched)

SML to code frames and topics

Some work by Burscher and colleagues

- Humans can code generic frames (human-interest, economic, . . .)
- Humans can code topics from a pre-defined list
- **But it is very hard to formulate an explicit rule** (as in: code as 'Human Interest' if regular expression R is matched)

⇒ This is where you need supervised machine learning!

Burscher, B., Odijk, D., Vliegenthart, R., De Rijke, M., & De Vreese, C. H. (2014). Teaching the computer to code frames in news: Comparing two supervised machine learning approaches to frame analysis. *Communication Methods and Measures, 8*(3), 190–206. doi:10.1080/19312458.2014.937527
Burscher, B., Vliegenthart, R., & De Vreese, C. H. (2015). Using supervised machine learning to code policy issues: Can classifiers generalize across contexts? *Annals of the American Academy of Political and Social Science, 659*(1), 122–131.

TABLE 4
Classification Accuracy of Frames in Sources Outside the Training Set

	VK/NRC →*Tel*	*VK/TEL* →*NRC*	*NRC/TEL* →*VK*
Conflict	.69	.74	.75
Economic Cons.	.88	.86	.86
Human Interest	.69	.71	.67
Morality	.97	.90	.89

Note. VK = Volkskrant, NRC = NRC/Handelsblad, TEL = Telegraaf

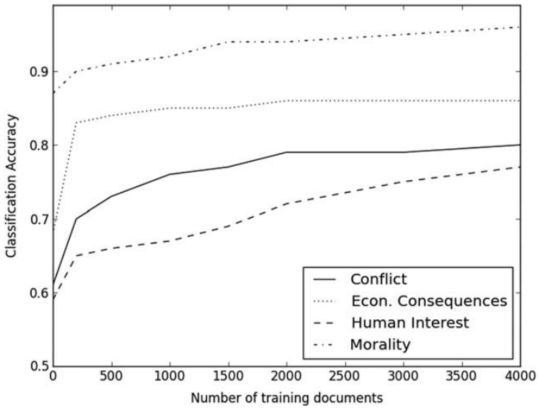


FIGURE 1 Relationship between classification accuracy and number of training documents.

FIGURE 1

Learning Curves for the Classification of News Articles and PQs

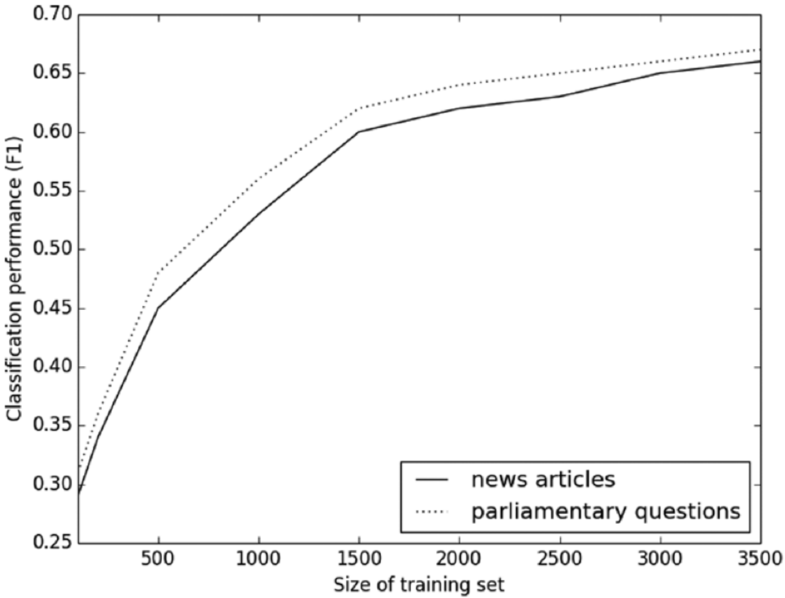
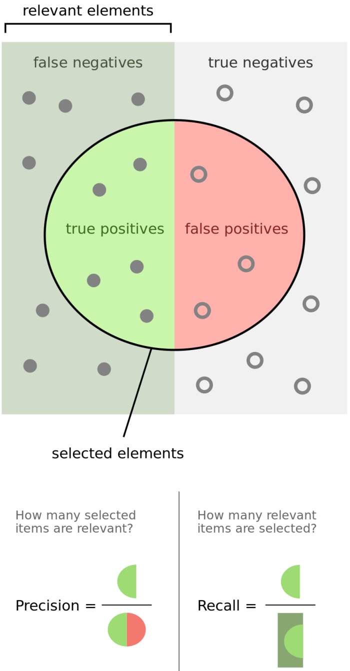


TABLE 1
F1 Scores for SML-Based Issue Coding in News Articles and PQs

Issue		News Articles		PQs	
		All Words	Lead Only		All Words
Features	*N*	F1	F1	*N*	F1
Macroeconomics	413	.54	.63	172	.46
Civil rights and minority issues	327	.34	.28	192	.53
Health	444	.70	.71	520	.81
Agriculture	114	.72	.76	159	.66
Labor and employment	217	.43	.49	174	.58
Education	188	.79	.71	229	.78
Environment	152	.34	.44	237	.59
Energy	81	.35	.59	67	.66
Immigration and integration	150	.50	.57	239	.78
Transportation	416	.58	.67	306	.81
Law and crime	1198	.70	.69	685	.77
Social welfare	115	.33	.34	214	.54
Community development and housing	113	.45	.44	136	.72
Banking, finance, and commerce	622	.62	.67	188	.58
Defense	393	.59	.55	196	.71
Science, technology, and communication	426	.64	.59	57	.53
International affairs and foreign aid	1,106	.70	.64	352	..65
Government operations	1,301	.71	.72	276	.48
Other issue	3,322	.84	.80	360	.51
Total	11,089	.71	.68	4,759	.69

NOTE: The F1 score is equal to the harmonic mean of recall and precision. Recall is the fraction of relevant documents that are retrieved, and precision is the fraction of retrieved documents that are relevant.

http://commons.wikimedia.org/wiki/File:Precisionrecall.svg



Some measures of accuracy

- Recall
- Precision
- $F1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$
- AUC (Area under curve) $[0, 1]$, $0.5 =$ random guessing

What does this mean for our research?

What does this mean for our research?

It we have 2,000 documents with manually coded frames and topics. . .

- we can use them to train a SML classifier
- which can code an unlimited number of new documents
- with an acceptable accuracy

Some easier tasks even need only 500 training documents, see Hopkins, D. J., & King, G. (2010). A method of automated nonparametric content analysis for social science. *American Journal of Political Science, 54*(1), 229–247.

An implementation

Let's say we have a list of tuples with movie reviews and their rating:

```
reviews=[("This is a great movie",1),("Bad movie",-1), ... ...]
```

And a second list with an identical structure:

```
test=[("Not that good",-1),("Nice film",1), ... ...]
```

Both are drawn from the same population, it is pure chance whether a specific review is on the one list or the other.

Based on an example from http://blog.dataquest.io/blog/naive-bayes-movies/

Training a A Naïve Bayes Classifier

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import CountVectorizer
from sklearn import metrics

# This is just an efficient way of computing word counts
vectorizer = CountVectorizer(stop_words='english')
train_features = vectorizer.fit_transform([r[0] for r in reviews])
test_features = vectorizer.transform([r[0] for r in test])

# Fit a naive bayes model to the training data.
nb = MultinomialNB()
nb.fit(train_features, [r[1] for r in reviews])

# Now we can use the model to predict classifications for our test
    features.
predictions = nb.predict(test_features)
actual=[r[1] for r in test]

# Compute the error.
fpr, tpr, thresholds = metrics.roc_curve(actual, predictions, pos_label
    =1)
print("Multinomal naive bayes AUC: {0}".format(metrics.auc(fpr, tpr)))
```

And it works!

Using 50,000 IMDB movies that are classified as either negative or positive,

- I created a list with 25,000 training tuples and another one with 25,000 test tuples and
- trained a classifier
- that achieved an AUC of .82.

Dataset obtained from `http://ai.stanford.edu/~amaas/data/sentiment`, Maas, A.L., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y., & Potts, C. (2011). Learning word vectors for sentiment analysis. *49th Annual Meeting of the Association for Computational Linguistics (ACL 2011)*

Playing around with new data

```
newdata=vectorizer.transform(["What a crappy movie! It sucks!", "This is
      awsome. I liked this movie a lot, fantastic actors","I would not
     recomment it to anyone.", "Enjoyed it a lot"])
predictions = nb.predict(newdata)
print(predictions)
```

This returns, as you would expect and hope:

```
[-1  1 -1  1]
```

But we can do even better

We can use different vectorizers and different classifiers.

Different vectorizers

- CountVectorizer (=simple word counts)
- TfidfVectorizer (word counts ("term frequency") weighted by number of documents in which the word occurs at all ("inverse document frequency"))
- additional options: stopwords, thresholds for minimum frequencies etc.

Different classifiers

- Naïve Bayes
- Logistic Regression
- Support Vector Machine (SVM)
- ...

Typical approach: Find out which setup performs best (see example source code in the book).
$\Rightarrow$ cross-validation; grid search

Things to rember

If you cannot formulate an explicit rule ($\Rightarrow$ regular expressions), but you can get hand-code ("annotate") a subset of your data of sufficient size, then you should consider using supervised machine learning.

From words to meaning

From words to meanings

In everything we discussed so far, our methods were agnostic to word meaning. But can't we incorporate information about the difference between words into our models?

Let's think about our features

Until now:

- basically BOW
- sometimes weighted ($tf \cdot idf$)
- sometimes ngrams

Let's think about our features

Until now:

- basically BOW
- sometimes weighted ($tf \cdot idf$)
- sometimes ngrams

How can we make this more meaningful?

NB: Sometimes it's just predictive performance that counts, sometimes it's meaning

More meaningful features

Strategy 1: Standardize

- stemming
- substitute known synonyms (e.g., with regular expressions)
- entity linking (named entity disambiguation)

More meaningful features

Strategy 2: Reduce number of features

- stopword removal
- keep only specific POS-tags (e.g., nouns and adjectives)
- if, e.g., NEs don't matter, replace all NER-Ps with `PERSON`

More meaningful features

Strategy 3: From ‘identical (yes/no)?’ towards ‘how similar?’

- Word embeddings, e.g. to calculate Word Mover’s Distance (WMD)

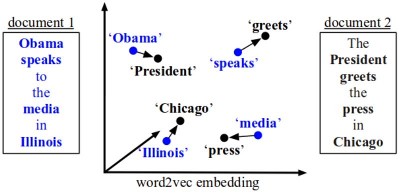


Figure 1. An illustration of the *word mover’s distance*. All non-stop words (***bold***) of both documents are embedded into a *word2vec* space. The distance between the two documents is the minimum cumulative distance that all words in document 1 need to travel to exactly match document 2. (Best viewed in color.)

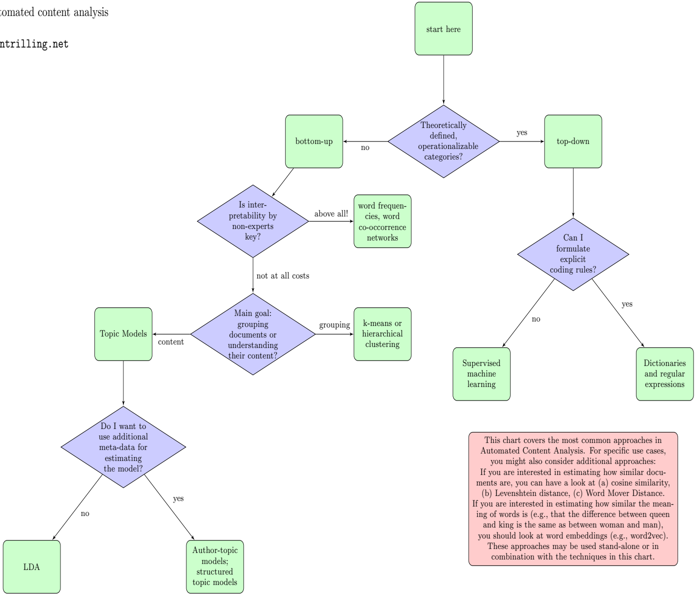
Kusner, M., Sun, Y., Kolkin, N., & Weinberger, K. (2015). From word embeddings to document distances. *International Conference on Machine Learning*

Take-home messages

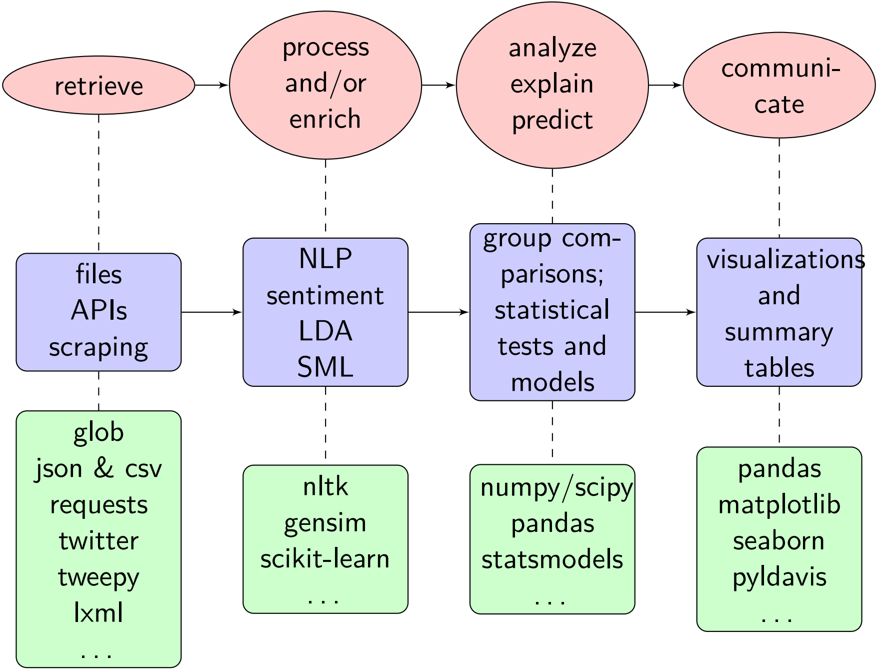
Take-home messages

- ACA can be bottom-up or top-down
- Which technique is to be used depends on
  - interpretability for non-experts vs. powerful models
  - whether explicit rules can be formulated
  - available data
- Outcomes and performance vary widely based on model specifications, preprocessing, ...

Decision aid for automated content analysis
by Damian Trilling
`http://www.damiantrilling.net`



This chart covers the most common approaches in Automated Content Analysis. For specific use cases, you might also consider additional approaches:
If you are interested in estimating how similar documents are, you can have a look at (a) cosine similarity, (b) Levenshtein distance, (c) Word Mover Distance.
If you are interested in estimating how similar the meaning of words is (e.g., that the difference between queen and king is the same as between woman and man), you should look at word embeddings (e.g., word2vec).
These approaches may be used stand-alone or in combination with the techniques in this chart.


retrieve
process and/or enrich
analyze explain predict
communi- cate
files APIs scraping
NLP sentiment LDA SML
group com- parisons; statistical tests and models
visualizations and summary tables
glob json & csv requests twitter tweepy lxml ...
nltk gensim scikit-learn ...
numpy/scipy pandas statsmodels ...
pandas matplotlib seaborn pyldavis ...

Questions?

Damian Trilling
d.c.trilling@uva.nl
@damian0604
www.damiantrilling.net