Big Data and Automated Content Analysis

Week 7 – Thursday

»Looking back and forward«

Anne Kroon

a.c.kroon@uva.nl
@annekroon

Afdeling Communicatiewetenschap
Universiteit van Amsterdam

May 16, 2019

Today

1. Looking back
   - Putting the pieces together
   - A good workflow
2. Looking forward
   - Techniqes we did not cover
3. Advanced Types of Automated Content Analysis
4. Unsupervised ML
   - PCA
   - LDA
5. Supervised Machine Learning
   - You have done it before!
   - Applications
   - An implementation
6. Final steps

Looking back

Putting the pieces together

First: Our epistomological underpinnings

Computational Social Science

Computational Social Science

"It is an approach to social inquiry defined by (1) the use of large, complex datasets, often—though not always— measured in terabytes or petabytes; (2) the frequent involvement of "naturally occurring" social and digital media sources and other electronic databases; (3) the use of computational or algorithmic solutions to generate patterns and inferences from these data; and (4) the applicability to social theory in a variety of domains from the study of mass opinion to public health, from examinations of political events to social movements"

Shah, D. V., Cappella, J. N., & Neuman, W. R. (2015). Big Data, digital media, and computational social science: Possibilities and perils. T*he ANNALS of the American Academy of Political and Social Science, 659*(1), 6–13. doi:10.1177/0002716215572084

Computational Social Science

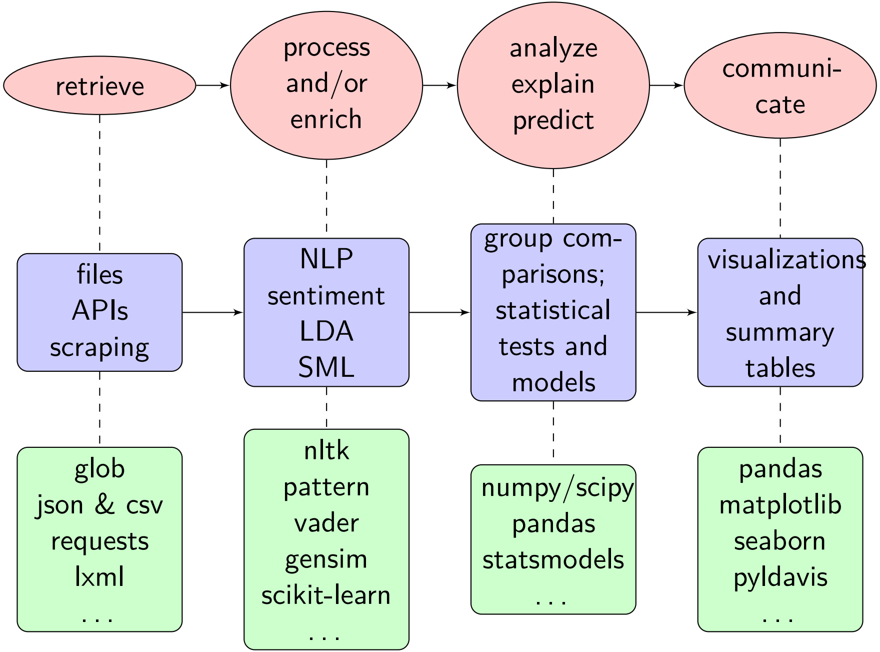
"[. . .] the computational social sciences employ the scientific method, complementing descriptive statistics with inferential statistics that seek to identify associations and causality. In other words, they are underpinned by an epistemology wherein the aim is to produce sophisticated statistical models that explain, simulate and predict human life."

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

Steps of a CSS project

We learned techniques for:

- retrieving data
- processing data
- analyzing data
- visualising data


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 lxml ...
nltk pattern vader gensim scikit-learn ...
numpy/scipy pandas statsmodels ...
pandas matplotlib seaborn pyldavis ...

A good workflow

The big picture

Start with pen and paper

1. Draw the Big Picture

The big picture

Start with pen and paper

1. Draw the Big Picture
2. Then work out what components you need

Develop components separately

One script for downloading the data, one script for analyzing

- Avoids waste of resources (e.g., unnecessary downloading multiple times)

Develop components separately

One script for downloading the data, one script for analyzing

- Avoids waste of resources (e.g., unnecessary downloading multiple times)
- Makes it easier to re-use your code or apply it to other data

Develop components separately

One script for downloading the data, one script for analyzing

- Avoids waste of resources (e.g., unnecessary downloading multiple times)
- Makes it easier to re-use your code or apply it to other data

Develop components separately

One script for downloading the data, one script for analyzing

- Avoids waste of resources (e.g., unnecessary downloading multiple times)
- Makes it easier to re-use your code or apply it to other data

Start small, then scale up

- Take your plan (see above) and solve *one* problem at a time (e.g., parsing a review page; or getting the URLs of all review pages)

Develop components separately

One script for downloading the data, one script for analyzing

- Avoids waste of resources (e.g., unnecessary downloading multiple times)
- Makes it easier to re-use your code or apply it to other data

Start small, then scale up

- Take your plan (see above) and solve *one* problem at a time (e.g., parsing a review page; or getting the URLs of all review pages)
- (for instance, by using functions [next slides])

Develop components separately

If you copy-paste code, you are doing something wrong

- Write loops!

Develop components separately

If you copy-paste code, you are doing something wrong

- Write loops!
- If something takes more than a couple of lines, write a function!

Copy-paste approach
(ugly, error-prone, hard to scale up)

```
allreviews = []

response = requests.get('http://xxxxx')
tree = fromstring(response.text)
reviewelements = tree.xpath('//div[@class="review"]')
reviews = [e.text for e in reviewelements]
allreviews.extend(reviews)

response = requests.get('http://yyyyy')
tree = fromstring(response.text)
reviewelements = tree.xpath('//div[@class="review"]')
reviews = [e.text for e in reviewelements]
allreviews.extend(reviews)
```

Better: for-loop
(easier to read, less error-prone, easier to scale up (e.g., more URLs, read URLs from a file or existing list)))

```
allreviews = []

urls = ['http://xxxxx', 'http://yyyyy']

for url in urls:
    response = requests.get(url)
    tree = fromstring(response.text)
    reviewelements = tree.xpath('//div[@class="review"]')
    reviews = [e.text for e in reviewelements]
    allreviews.extend(reviews)
```

Even better: for-loop with functions
(main loop is easier to read, function can be re-used in multiple contexts)

```
def getreviews(url):
    response = requests.get(url)
    tree = fromstring(response.text)
    reviewelements = tree.xpath('//div[@class="review"]')
    return [e.text for e in reviewelements]


urls = ['http://xxxxx', 'http://yyyyy']

allreviews = []

for url in urls:
    allreviews.extend(getreviews(url))
```

Scaling up

Until now, we did not look too much into aspects like code style, re-usability, scalability

- Use functions and classes (Appendix D.3) to make code more readable and re-usable
- Avoid re-calculating values
- Think about how to minimize memory usage (e.g., Generators, Appendix D.2)
- Do not hard-code values, file names, etc., but take them as arguments

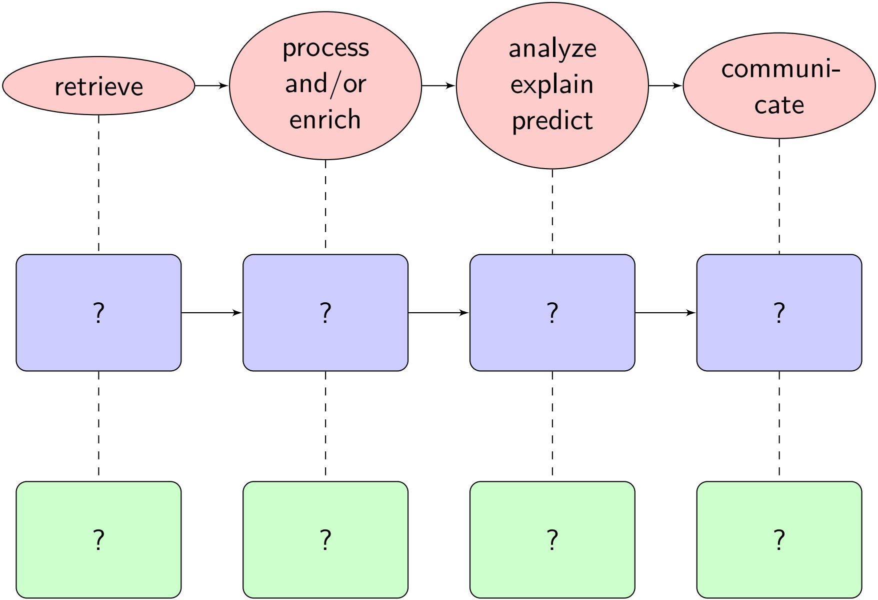
Make it robust

You cannot foresee every possible problem.
Most important: Make sure your program does not fail and loose all data just because something goes wrong at case 997/1000.

- Use `try/except` to explicitly tell the program how to handle errors
- Write data to files (or database) in between
- Use `assert len(x) == len(y)` for sanity checks

Looking forward

What other possibilities do exist for each step?


retrieve
process and/or enrich
analyze explain predict
communi- cate
?
?
?
?
?
?
?
?

Retrieve

Webscraping with Selenium

- If content is dynamically loaded (e.g., with JavaScript), our approach doesn't work (because we don't have a browser).
- Solution: Have Python literally open a browser and literally click on things
- $\Rightarrow$ Appendix E

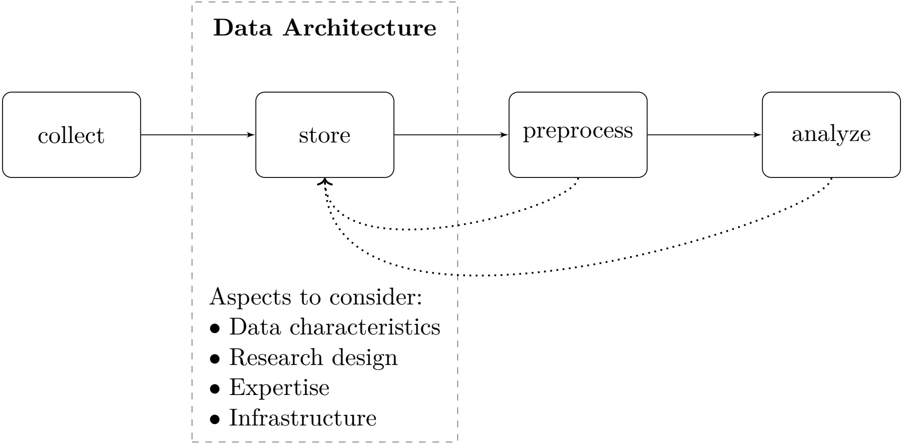
Retrieve

Use of databases

We did not discuss how to actually store the data

- We basically stored our data in files (often, one CSV or JSON file)
- But that's not very efficient if we have large datasets; especially if we want to select subsets later on
- SQL-databases to store tables (e.g., MySQL)
- NoSQL-databases to store less structured data (e.g., JSON with unknown keys) (e.g., MongoDB, ElasticSearch)
- $\Longrightarrow$ Günther, E., Trilling, D., & Van de Velde, R.N. (2018). But how do we store it? (Big) data architecture in the social-scientific research process. In: *Stuetzer, C.M., Welker, M., & Egger, M. (eds.): Computational Social Science in the Age of Big Data. Concepts, Methodologies, Tools, and Applications*. Cologne, Germany: Herbert von Halem.

Storing data



From retrieved data to enriched data

1. Retrieved →

```
<div>
<span id='author'
class='big'>
Author: John
Doe </span>
<span id='viewed'>
seen 42 times
</span>
</div>
```

2. Structured →

```
{
"author":
"Author: John Doe",
"viewed":
"seen 42 times"
}
```

3. Cleaned →

```
{
"author":
"john doe",
"viewed":
42,
}
```

4. Enriched

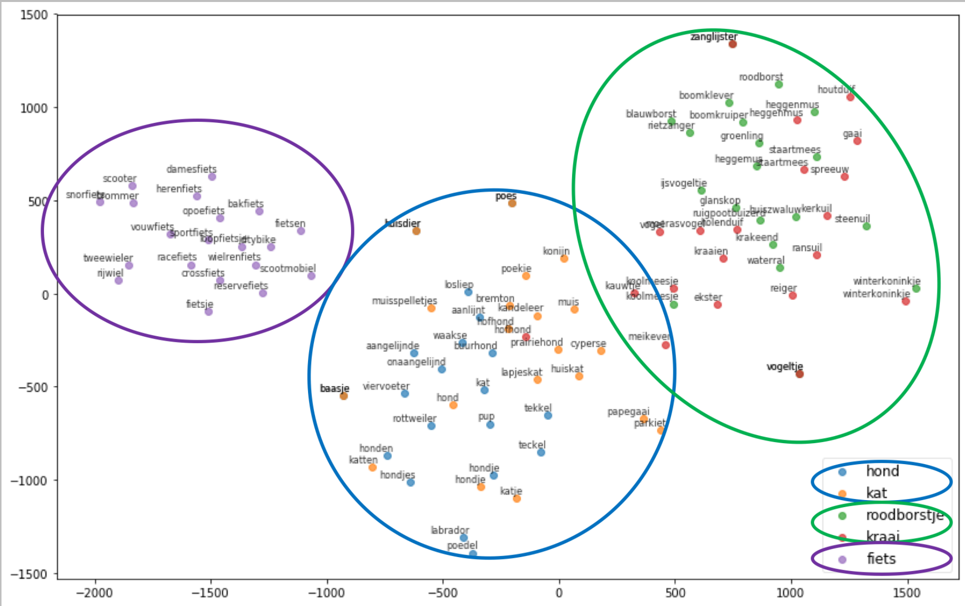
```
{
"author":
"john doe",
"author_gender":
"M",
"viewed":
42,
}
```

Process and/or enrich

Word embeddings

We did not really consider the *meaning* of words

- Word embeddings can be trained on large corpora (e.g., whole wikipedia or a couple of years of newspaper coverage)
- The trained model allows you to calculate with words (hence, word vectors): $king - man + woman = ?$
- You can find out whether documents are similar *even if they do not use the same words* (Word Mover Distance)
- $\Rightarrow$ word2vec (in gensim!), glove

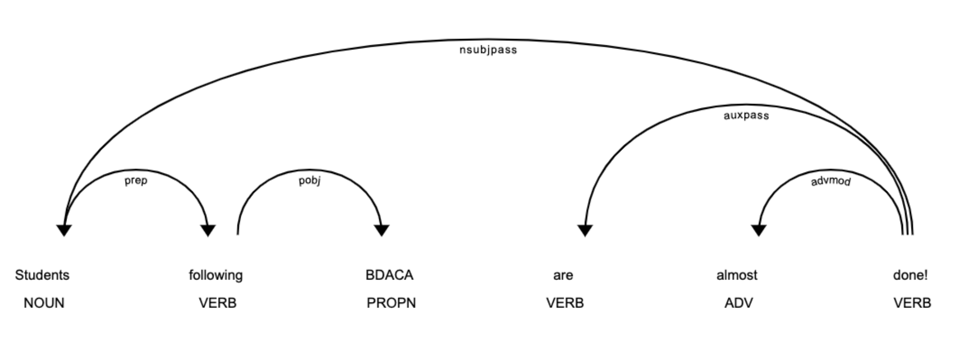

zanglijster
roodborst
boomklever
houtduif
heggenmus
blauwborst
boomkruiper
heggenmus
rietzanger
groenling
gaai
staartmees
heggemus
staartmees
spreeuw
ijsvogeltje
glanskop
huiszwaluw
kerkuil
ruigpootbuizerd
steenuil
holenduif
krakeend
ransuil
kraaien
waterral
kauwtje
koolmeesje
koolmeesje
ekster
reiger
winterkoninkje
winterkoninkje
meikever
vogeltje
damesfiets
scooter
herenfiets
snorfiets
brommer
bakfiets
opoefiets
fietsen
vouwfiets
sportfiets
loopfietsje
citybike
tweewieler
racefiets
wielrenfiets
scootmobiel
rijwiel
crossfiets
reservefiets
fietsje
poes
huisdier
konijn
poekie
loslopend
muisspelletjes
bremton
muis
aanlijnt
kandeleer
hofhond
waakse
prairiehond
cyperse
aangelijnde
buurhond
onaangelijnd
lapjeskat
huiskat
baasje
viervoeter
kat
hond
tekkel
papegaai
parkiet
rottweiler
pup
teckel
honden
katten
hondje
hondjes
hondje
katje
labrador
poedel
hond
kat
roodborstje
kraai
fiets
1500
1000
500
0
−500
−1000
−1500
−2000
−1500
−1000
−500
0
500
1000
1500

Process and/or enrich

Advanced NLP

We did a lot of BOW (and some POS-tagging), but we can get more

- Named Entity Recognition (NER) to get names of people, organizations, ...
- Dependency Parsing to find out exact relationships ⇒ nltk, Stanford, FROG, Spacy



PREP: Prepositional modifier
NSUBJPASS: Nominal subject (passive)
AUXPASS: Auxiliary (passive)

Analyze/explain/predict

More advanced modelling

We only did some basic statistical tests

- Especially with social media data, we often have time series (VAR models etc.)
- $\Rightarrow$ scikit-learn, statsmodels

Recap: Types of 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

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
- ...

inductive and bottom-up:
unsupervised machine learning

inductive and bottom-up:
unsupervised machine learning

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

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.

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.
$W_{i,j} = \mathrm{tf}_{i,j} \cdot \log(\frac{N}{\mathrm{df}_i})$

PCA: implications and problems

- given a term-document matrix, easy to do with any tool
- 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?

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
...
```

Data Editor (Browse) – topicscores.dta

Edit | Browse | Filter | Variables | Properties | Snapshots

topic4[2] .019

	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

Variables

Enter filter text here

Name	Label
byline	
section	

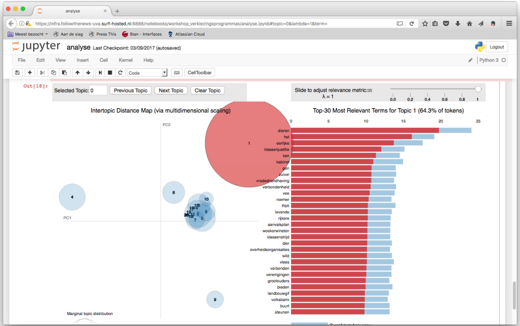
Properties

▼Variables	
Name	section
Label	
Type	str26
Format	%26s
Value Label	
Notes	
▼Data	
Filename	topicscores.
Label	
Notes	
Variables	164
Observations	28,406
Size	14.06M
Memory	64M

Vars: 158 of 164 | Obs: 28,406 | Filter: Off

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)
```



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

Supervised

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

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

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*: $\hat{y}_{man20} = -.8 + .4 \times 1 + .08 \times 20 = 1.2$ $\hat{y}_{woman40} = -.8 + .4 \times 0 + .08 \times 40 = 2.4$

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$\times$idf) ($\Rightarrow$BOW-representation)

Applications

Applications

In other fields

A lot of different applications

- This principle can be applied to many fields and disciplines – for example, it is possible to teach a computer to indicate if a tumor is present or not on X-rays of people's brains
- from recognizing hand-written characters to recommendation systems
- Supervised Machine learning does not care about what the features mean, so instead of texts we can also classify pictures

Applications

In other fields

A lot of different applications

- This principle can be applied to many fields and disciplines – for example, it is possible to teach a computer to indicate if a tumor is present or not on X-rays of people's brains
- from recognizing hand-written characters to recommendation systems
- Supervised Machine learning does not care about what the features mean, so instead of texts we can also classify pictures

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

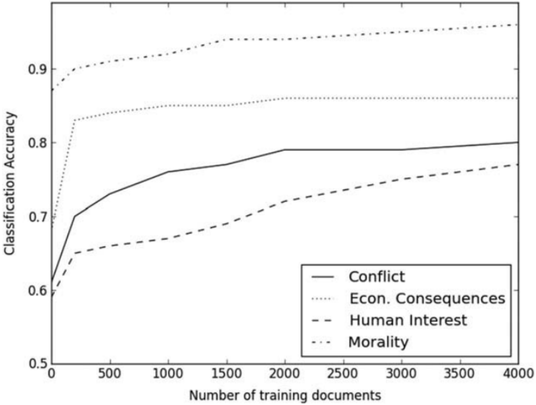


FIGURE 1 Relationship between classification accuracy and number of training documents.

FIGURE 1
Learning Curves for the Classification of News Articles and PQs

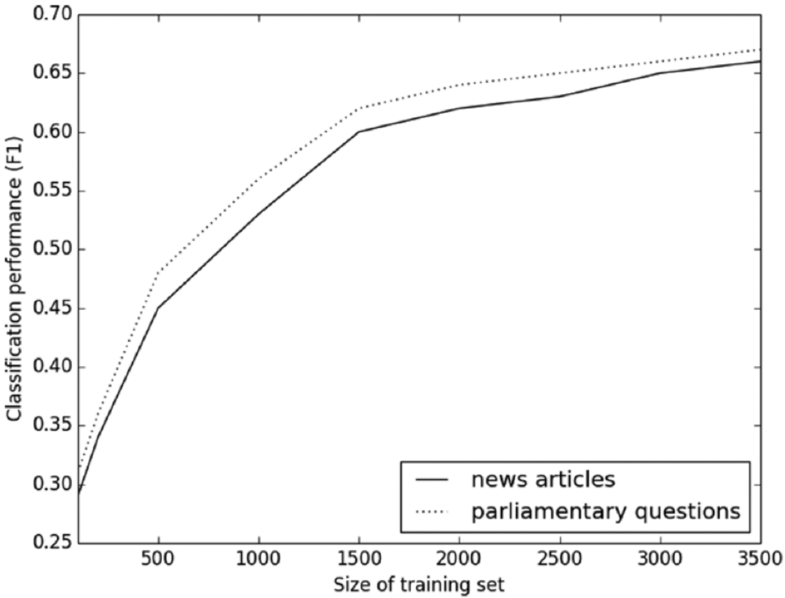


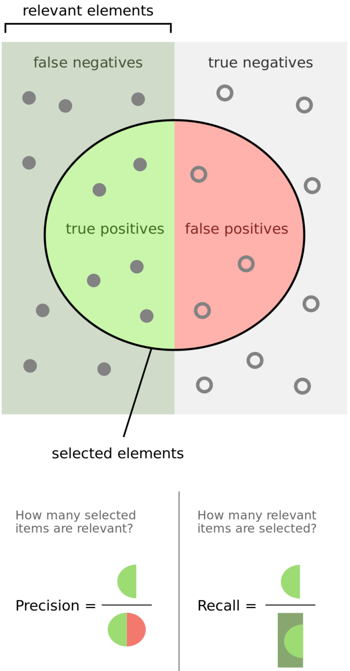
TABLE 1

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

- Precision
- Recall
- $\mathrm{F1} = 2 \cdot \dfrac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$
- 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 : {0}".format(metrics.auc(fpr, tpr)))
```

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).

Next meeting

Monday 20 – 5

Final chance for questions regarding final project (if you don't have any, you don't need to come.)

Deadline final exam

Hand in via filesender until Wednesday, 29–5, 23.59

One .zip or .tar.gz file with

- .py and/or .ipynb for code
- .pdf for text and figures
- .csv, .json, or .txt for data
- any additional file we need to understand or reproduce your work

Send to: A.C.kroon@uva.nl