Big Data & Automated Content Analysis
Week 7 – Wednesday: »Text as data«

Anne Kroon

a.c.kroon@uva.nl
@annekroon

10 May 2021

Afdeling Communicatiewetenschap
Universiteit van Amsterdam

Today

- Bag-of-words
  - General idea
- clean BOW
  - Better tokenization
  - Stopword removal
  - Stemming and lemmatization
  - How further?
- Unsupervised machine learning
- Finding similar variables
  - Principal Component Analysis
- LDA Topic models

How did the exam go?

Everything clear from last week?

Bag-of-words

Bag-of-words

General idea

A text as a collections of word

Let us represent a string

```
t = "This this is is is a test test test"
```

like this:

```
from collections import Counter
print(Counter(t.split()))
```

```
Counter({'is': 3, 'test': 3, 'This': 1, 'this': 1, 'a': 1})
```

Compared to the original string, this representation

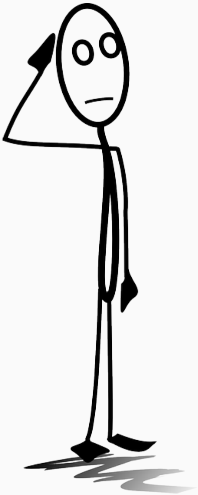
- is less repetitive
- preserves word frequencies
- but does *not* preserve word order
- can be interpreted as a vector to calculate with (!!!)

From vector to matrix

If we do this for multiple texts, we can arrange the vectors in a table.

t1 = "This this is is is a test test test"
t2 = "This is an example"

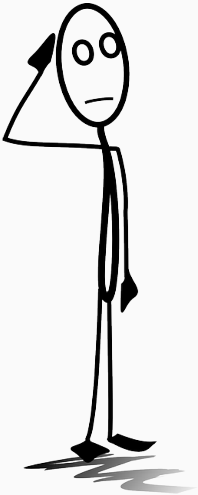
	a	an	example	is	this	This	test
t1	1	0	0	3	1	1	3
t2	0	1	1	1	0	1	0



What can you do with such a matrix?
Why would you want to represent a collection of texts in such a way?

The cell entries: raw counts versus tf·idf scores

- In the example, we entered simple counts (the "term frequency")



But are all terms equally important?

The cell entries: raw counts versus tf·idf scores

- In the example, we entered simple counts (the "term frequency")
- But does a word that occurs in almost all documents contain much information?
- And isn't the presence of a word that occurs in very few documents a pretty strong hint?
- **Solution: Weigh by *the number of documents in which the term occurs at least once) (the "document frequency")***

⇒ we multiply the "term frequency" (tf) by the inverse document frequency (idf)

(usually with some additional logarithmic transformation and normalization applied, see
https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text

tf·idf

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

$$\begin{aligned} tf_{i,j} &= \text{ number of occurrences of } i \text{ in } j \\ df_i &= \text{ number of documents containing } i \\ N &= \text{total number of documents} \end{aligned}$$

Is tf·idf always better?

It depends.

- Ultimately, it's an empirical question which works better ($\rightarrow$ weeks on machine learning)
- In many scenarios, "discounting" too frequent words and "boosting" rare words makes a lot of sense (most frequent words in a text can be highly un-informative)
- Beauty of raw tf counts, though: interpretability + describes document in itself, not in relation to other documents

clean BOW

Room for improvement

tokenization How do we (best) split a sentence into tokens (terms, words)?

pruning How can we remove unneccessary words?

lemmatization How can we make sure that slight variations of the same word are not counted differently?

clean BOW

Better tokenization

OK, good enough, perfect?

.split()

- space → new word
- no further processing whatsoever
- thus, only works well if we do a preprocessing outselves (e.g., remove punctuation)

```python
docs = ["This is a text", "I haven't seen John's derring-do. Second
     sentence!"]
tokens = [d.split() for d in docs]
```

```
[['This', 'is', 'a', 'text'], ['I', "haven't", 'seen', "John's", 'derring-do.', 'Second', '
     sentence!']]
```

OK, good enough, perfect?

Tokenizers from the NLTK pacakge

- multiple improved tokenizers that can be used instead of .split()
- e.g., Treebank tokenizer:
    - split standard contractions ("don't")
    - deals with punctuation

```
from nltk.tokenize import TreebankWordTokenizer
tokens = [TreebankWordTokenizer().tokenize(d) for d in docs]
```

```
[['This', 'is', 'a', 'text'], ['I', 'have', "n't", 'seen', 'John', "'s", 'derring-do.', 'Second
      ', 'sentence', '!']]
```

Notice the failure to split the . at the end of the first sentence in the second doc. That's because TreebankWordTokenizer expects *sentences* as input. See book for a solution.

OK, so we can tokenize with a list comprehension (and that's often a good idea!). But what if we want to *directly* get a DTM instead of lists of tokens?

OK, good enough, perfect?

scikit-learn's CountVectorizer (default settings)

- applies lowercasing
- deals with punctuation etc. itself
- minimum word length > 1
- more technically, tokenizes using this regular expression: `r"(?u)\b\w\w+\b"` [1]

```
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer()
dtm_sparse = cv.fit_transform(docs)
```

[1] ?u = support unicode, \b = word boundary

OK, good enough, perfect?

CountVectorizer supports more

- stopword removal
- custom regular expression
- or even using an external tokenizer
- ngrams instead of unigrams

see https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

Best of both worlds

Use the Count vectorizer with a NLTK-based external tokenizer! (see book)

clean BOW

Stopword removal

Stopword removal

What are stopwords?

- Very frequent words with little inherent meaning
- `the, a, he, she, ...`
- context-dependent: if you are interested in gender, `he` and `she` are no stopwords.
- Many existing lists as basis

When using the CountVectorizer, we can simply provide a stopword list.

But we can also remove stopwords “by hand” (next slide):

Stopword removal

```
1  from nltk.corpus import stopwords
2  mystopwords = stopwords.words("english")
3  mystopwords.extend(["test", "this"])
4
5  def tokenize_clean(s, stoplist):
6      cleantokens = []
7      for w in TreebankWordTokenizer().tokenize(s):
8          if w.lower() not in stoplist:
9              cleantokens.append(w)
10     return cleantokens
11
12 tokens = [tokenize_clean(d, mystopwords) for d in docs]
```

```
[['text'], ["n't", 'seen', 'John', 'derring-do.', 'Second', 'sentence', '!']]
```

You can do more!

For instance, in line 8, you could add an or statement to also exclude punctuation.

CountVectorizer, only stopword removal

```
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
myvectorizer = CountVectorizer(stop_words=mystopwords)
```

CountVectorizer, better tokenization, stopword removal (pay attention that stopword list uses same tokenization!):

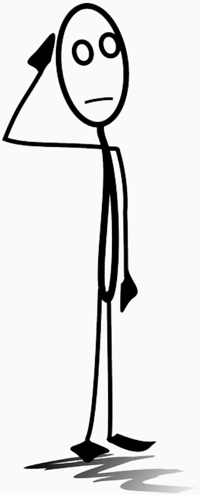
```
myvectorizer = CountVectorizer(tokenizer = TreebankWordTokenizer().tokenize, stop_words=mystopwords)
```

Additionally remove words that occur in more than 75% or less than $n = 2$ documents:

```
myvectorizer = CountVectorizer(tokenizer = TreebankWordTokenizer().tokenize, stop_words=mystopwords, max_df=.75, min_df=2)
```

All togehter: tf·idf, explicit stopword removal, pruning

```
myvectorizer = TfidfVectorizer(tokenizer = TreebankWordTokenizer().tokenize, stop_words=mystopwords, max_df=.75, min_df=2)
```



What is "best"? Which (combination of) techniques to use, and how to decide?

clean BOW

Stemming and lemmatization

Stemming and lemmatization

- Stemming: reduce words to its stem by removing last part (drinking → drink)
- Lemmatization: find word that you would need to look up in a dictionary (drinking → drink, but also went → go)
- stemming is simpler than lemmatization
- lemmatization often better

Example below: tokenization and lemmatization with spacy in one go:

```
import spacy
nlp = spacy.load('en') # potentially you need to install the language
    model first
lemmatized_tokens = [[token.lemma_ for token in nlp(doc)] for doc in
    docs]
```

clean BOW

How further?

Main takeaway

- It matters how you transform your text into numbers ("vectorization").
- Preprocessing matters, be able to make informed choices.
- Keep this in mind when we will discuss Machine Learning.

- Once you vectorized your texts, you can do all kinds of calculations (random example: get the cosine similarity between two texts)

More NLP

***n*-grams** Consider using *n*-grams instead of unigrams

POS-tagging grammatical function ("part-of-speach") of tokens

NER named entity recognition (persons, organizations, locations)

More NLP

I **really** recommend looking into spacy (https://spacy.io) for advanced natural language processing, such as part-of-speech-tagging and named entity recogntion.

Unsupervised machine learning

Unsupervised machine learning

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

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

- Principal Component Analysis (PCA) and Singular Value Decomposition (SVD)
- Cluster analysis
- Topic modelling (Non-negative matrix factorization and Latent Dirichlet Allocation)
- ...

Unsupervised techniques. . .

1. Finding similar variables (dimensionality reduction) – unsupervised
2. Finding similar cases (clustering) – unsupervised

	x1	x2	x3	x4	x5	y
case1	110	110	110	110	110	110
case2	110	110	110	110	110	110
case3	110	110	110	110	110	110
case4	110	110	110	110	110	110

	x1	x2	x3	x4	x5	(y)
case1	110	110	110	110	110	(110)
case2	110	110	110	110	110	(110)
case3	110	110	110	110	110	(110)
case4	110	110	110	110	110	(110)

Dimensionality reduction: finding similar variables (features)

	x1	x2	x3	x4	x5	(y)
case1	110	110	110	110	110	(110)
case2	110	110	110	110	110	(110)
case3	110	110	110	110	110	(110)
case4	110	110	110	110	110	(110)

Clustering: finding similar cases

Finding similar variables

Finding similar variables

Principal Component Analysis

PCA

Document-term 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)

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

PCA

- related to and often confused with Factor Analysis (same menu item in SPSS – many people who believe they run FA actually run PCA!)
- Components are ordered (first explains most variance)
- Components do *not* necessarily carry a meaningful interpretation

Running PCA

Example of a PCA on a BOW representation of some texts:

```
myvec = CountVectorizer(texts, max_df=.5, min_df=5)
mypca = PCA(n_components=2)

mypipe = make_pipeline(myvec, FunctionTransformer(lambda x: x.todense(),
      accept_sparse=True), mypca)

r = mypipe.fit_transform(texts)
```

PCA does not accept a *sparse matrix* as input (but the CountVectorizer gives one as output), so we need to transform it into a *dense matrix*.

We need other models to

1. model *simultaneously* (a) which topics we find in the whole corpus, and (b) which of these topics are present in which document; while at the same time
2. allowing (a) words to be part of multiple topics, and (b) multiple topics to be present in one document

Maier, D., Waldherr, A., Miltner, P., Wiedemann, G., Niekler, A., Keinert, A., . . . Adam, S. (2018). Applying LDA Topic Modeling in Communication Research: Toward a Valid and Reliable Methodology. *Communication Methods and Measures, 12*(2–3), 93–118. doi:10.1080/19312458.2018.1430754

LDA Topic models

LDA Topic models

An introduction to LDA

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

LDA, what's that?

No mathematical details here, but the general idea

- There are k topics, $T_1 \dots T_k$
- Each document D_i consists of a mixture of these topics, e.g.$80\% T_1, 15\% T_2, 0\% T_3, \dots 5\% T_k$
- 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=[[token for token in re.split(r"\W", art) if len(token)>0] 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]))
```

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	

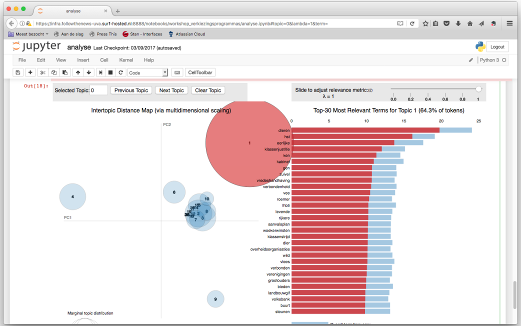
Properties

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

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



Visualization with pyldavis

Short note about the λ setting:

It influences the ordering of the words in pyldavis.

> *"For $\lambda = 1$, the ordering of the top words is equal to the ordering of the standard conditional word probabilities. For λ close to zero, the most specific words of the topic will lead the list of top words. In their case study, Sievert and Shirley (2014, p. 67) found the best interpretability of topics using a λ-value close to .6, which we adopted for our own case" (Maier et al., 2018, p. 107)*

Maier, D., Waldherr, A., Miltner, P., Wiedemann, G., Niekler, A., Keinert, A., . . . Adam, S. (2018). Applying LDA Topic Modeling in Communication Research: Toward a Valid and Reliable Methodology. *Communication Methods and Measures, 12*(2–3), 93–118. doi:10.1080/19312458.2018.1430754

Code examples

https://github.com/annekroon/bdaca-6ec/blob/master/6ec/week07/exercises/lda.ipynb

LDA Topic models

Choosing the best (or a good) topic model

Choosing the best (or a good) topic model

- There is no single best solution (e.g., do you want more coarse of fine-grained topics?)
- Non-deterministic
- Very sensitive to preprocessing choices
- Interplay of both metrics and (qualitative) interpretability

See for more elaborate guidance:

Maier, D., Waldherr, A., Miltner, P., Wiedemann, G., Niekler, A., Keinert, A., . . . Adam, S. (2018). Applying LDA Topic Modeling in Communication Research: Toward a Valid and Reliable Methodology. *Communication Methods and Measures, 12*(2–3), 93–118. doi:10.1080/19312458.2018.1430754

Evaluation metrics (closer to zero is better)

perplexity

A goodness-of-fit measure, answering the question: If we do a train-test split, how well does the trained model fit the test data?

coherence

- mean coherence of the whole model: attempts to quantify the interpretability
- coherence per topic: allows to get topics that are most likely to be coherently interpreted (`.top_topics()`)

Choosing k: How many topics do we want?

- Typical values: $10 < k < 200$
- Too low: losing nuance, so broad it becomes meaningless
- Too high: picks up tiny pecularities instead of finding general patterns
- There is no inherent ordering of topics (unlike PCA!)
- We can throw away or merge topics later, so if out of $k = 50$ topics 5 are not interpretable and a couple of others overlap, it still may be a good model

Choosing α: how sparse should the document-topic distribution θ be?

- The higher α, the more topics per document
- Default: $1/k$
- But: We can explicitly change it, or – really cool – even learn α from the data (`alpha = "auto"`)

Takeaway: It takes longer, but you probably want to learn alpha from the data, using multiple passes:

```
mylda LdaModel(corpus=tfidfcorpus[ldacorpus], id2word=id2word,
    num_topics=50, alpha='auto', passes=10)
```

LDA Topic models

Using topic models

Using topic models

You got your model – what now?

1. Assign topic scores to documents
2. Label topics
3. Merge/ throw away topics
4. Compare topics between, e.g., outlets
5. or do some time-series analysis.

Example: Tsur, O., Calacci, D., & Lazer, D. (2015). A Frame of Mind: Using Statistical Models for Detection of Framing and Agenda Setting Campaigns. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing* (pp. 1629–1638).

Exercise

Exercise for this week

- Work through the example notebook on LDA: https://github.com/annekroon/bdaca-6ec/blob/master/6ec/week07/exercises/lda.ipynb
- But most importantly: **Use a dataset of your choice** and find a suitable topic model.