Defining the problem

Remember our earlier distinction:

1. Finding similar variables (dimension reduction)
2. Finding similar cases (clustering)

Are we more interested in which features "belong together" or which cases "belong together"?

Conceptually, we want to know *both* which features (words) belong to each other (=form a topic), *and* which cases (documents) contain the same topics.

Defining the problem

Remember our earlier distinction:

1. Finding similar variables (dimension reduction)
2. Finding similar cases (clustering)

Are we more interested in which features "belong together" or which cases "belong together"?

Conceptually, we want to know *both* which features (words) belong to each other (=form a topic), *and* which cases (documents) contain the same topics.

Defining the problem

Remember our earlier distinction:

1. Finding similar variables (dimension reduction)
2. Finding similar cases (clustering)

Are we more interested in which features “belong together” or which cases “belong together”?

Conceptually, we want to know *both* which features (words) belong to each other (=form a topic), *and* which cases (documents) contain the same topics.

Defining the problem

We assume a BOW approach like this (as produced by scikit-learn vectorizer):

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 ...
...
```

raw counts or tf·idf scores

Defining the problem

We could then go via two routes:

1. We run a PCA/SVD to see which features (words) load on the same component; and *then* look at the component scores per document
2. We run a k-means cluster analysis to see which texts are similar; and *then* look at the most common words per cluster

Defining the problem

We could then go via two routes:

1. We run a PCA/SVD to see which features (words) load on the same component; and *then* look at the component scores per document
2. We run a k-means cluster analysis to see which texts are similar; and *then* look at the most common words per cluster

Some considerations

If we do PCA/SVD. . .

- Components are ordered (first explains most variance) ⇒ We assume that some topics are more important than others
- Components do *not* necessarily carry a meaningful interpretation ⇒ But maybe OK in practice?
- We assume that a word belongs to one (not multiple) topics
- We assume that a document has a score for each topic

Some considerations

If we do cluster analysis. . .

- We assume that (in the case of *k*-means) that topics (are roughly) simiarly sized
- We assume that a *document* belongs to one (not multiple) topics
- We assume that a word *can* belong to multiple topics.

Both approaches have been used

- Both have different assumptions, implications, and constraints
- Both easy to do with scikit-learn
- Both used in research (for instance, PCA by Leydesdorff and Nerghes (2017) or Greussing and Boomgaarden (2017); or *k*-means cluster analysis by Burscher et al. (2016))
- Typically, PCA groups features, cluster analysis groups texts – (but you can then use the component scrores to describe the texts, and the cluster centroids to describe the features)
- Still ocasionally used, but in general considered outdated

You find some slides with code examples in the appendix.

Beyond PCA and k-means

PCA was invented in 1901 (!), and k-means is around since the 1950s/1960s.

There surely must be something newer!

There is: Latent Dirichlet Allocation (LDA) (D. Blei et al., 2003).

Beyond PCA and k-means

PCA was invented in 1901 (!), and k-means is around since the 1950s/1960s.

There surely must be something newer!

There is: Latent Dirichlet Allocation (LDA) (D. Blei et al., 2003).

LDA solves some problems

Actually, we have *two* things we want to model:

1. Which topics can we extract from the corpus?
2. How present is each of these topics in each text in the corpus?

$\Rightarrow$ LDA does both simultaneously!

It also does not suffer from a few problems:

- 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?
- does the goal of cluster analysis, assigning each document to *one* cluster, match real life?

LDA solves some problems

LDA is a model that

1. estimates *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; and
3. being able to make connections between words “even if they never actually occured in a document together” (Maier et al., 2018, p. 96)

Let that last point sink in for a second!

LDA solves some problems

LDA is a model that

1. estimates *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; and
3. being able to make connections between words "even if they never actually occured in a document together" (Maier et al., 2018, p. 96)

Let that last point sink in for a second!

LDA solves some problems

LDA is a model that

1. estimates *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; and
3. being able to make connections between words "even if they never actually occured in a document together" (Maier et al., 2018, p. 96)

Let that last point sink in for a second!

LDA solves some problems

LDA is a model that

1. estimates *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; and
3. being able to make connections between words "even if they never actually occured in a document together" (Maier et al., 2018, p. 96)

Let that last point sink in for a second!

LDA solves some problems

LDA is a model that

1. estimates *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; and
3. being able to make connections between words "even if they never actually occured in a document together" (Maier et al., 2018, p. 96)

Let that last point sink in for a second!

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 and 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:

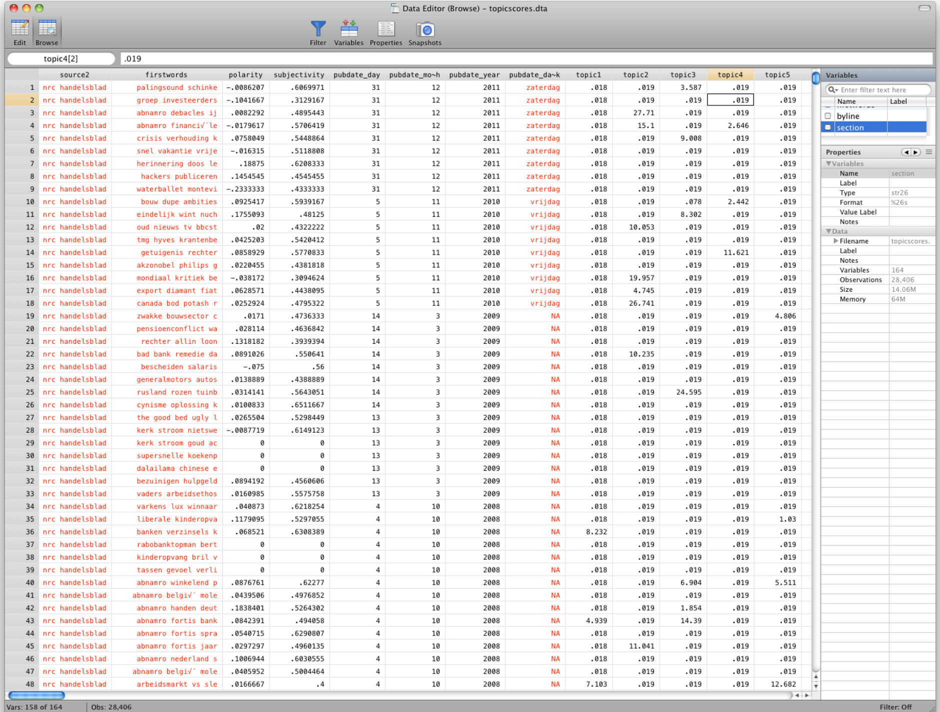
```
[['The', 'tax', 'deficit', 'is', 'higher', 'than', 'expected', 'This', 'said', 'xxx'], ['Germany', 'won', 'the', 'World', 'Cup', 'After', 'a']]
```

(note that we of course could use a better tokenizer!)

```
from gensim import corpora, models
import pandas as pd

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)

# the topic scores per document
topics = pd.DataFrame([dict(mylda.get_document_topics(doc, minimum_probability=0.0)) for doc in mm])
```

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

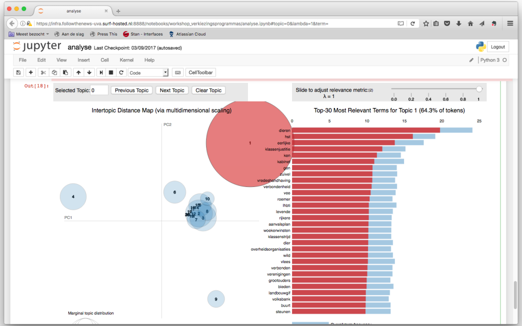
```
0.069*fusie + 0.058*brussel + 0.045*europesecommissie + 0.036*europese + 0.023*overname
0.109*bank + 0.066*britse + 0.041*regering + 0.035*financien + 0.033*minister
0.114*nederlandse + 0.106*nederland + 0.070*bedrijven + 0.042*rusland + 0.038*russische
0.093*nederlandsespoorwegen + 0.074*den + 0.036*jaar + 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_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)*

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()`)

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()`)

So, how do we do this?

- Basically, similar to the idea behind our grid search from two weeks ago: estimate multiple models, store the metrics for each model, and then compare them (numerically, or by plotting)
- Idea: We select some candidate models, and then look whether they can be interpreted.
- But what can we tune?

So, how do we do this?

- Basically, similar to the idea behind our grid search from two weeks ago: estimate multiple models, store the metrics for each model, and then compare them (numerically, or by plotting)
- Idea: We select some candidate models, and then look whether they can be interpreted.
- But what can we tune?

So, how do we do this?

- Basically, similar to the idea behind our grid search from two weeks ago: estimate multiple models, store the metrics for each model, and then compare them (numerically, or by plotting)
- Idea: We select some candidate models, and then look whether they can be interpreted.
- But what can we tune?

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)
```

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)
```

Choosing η: how sparse should the topic-word distribution λ be?

- Can be used to boost specific words
- Can also be learned from the data

Takeaway: Even though you can do `eta="auto"`, this usually does not help you much.

Choosing η: how sparse should the topic-word distribution λ be?

- Can be used to boost specific words
- Can also be learned from the data

Takeaway: Even though you can do `eta="auto"`, this usually does not help you much.

Using topic models

You got your model – what now?

1. Assign topic scores to documents
2. Label topics
3. Merge topics, throw away boilerplate topics and similar (manually, or aided by cluster analysis)
4. Compare topics between, e.g., outlets
5. or do some time-series analysis.

Example: Tsur et al., 2015

Unsupervised ML

Should one still use LDA?

The popularity of LDA

In the last decade, LDA has become *extremely* popular in the social sciences due to

- easy-to-use R and Python packages
- its promise to not require (a) manual (qual or quant) analysis; (b) annotations for SML; (c) creation of dictionaries etc.
- a bit of a "cool new technique" image

The popularity of LDA

But there is no silver bullet!

Unfortunately,

- validating topic models is hard – and many (most) studies don't do it (well);
- there are so many choices and parameters, in combination with no simple and definite evaluation metric, that it is very hard to justify why a particular model is chosen;
- experience shows that it often "doesn't work" $\Rightarrow$ it's quite normal to have many uninterpretable or ambigous topics;
- The smaller the dataset, the less likely it is to work
- LDA tends to also pick up pecularities that don't matter and outliers

Solutions?

There are some extensions on classical LDA, in particular:

- Author-topic models
- Structural topic models (STM) (Roberts et al., 2014)
- Dynamic topic models (D. M. Blei & Lafferty, 2006)

These allow covariates (e.g., add info on who wrote a text) to improve the model, or allow to account for the changing use of words and topics over time.

Also, there are techniques for validation available (e.g., topic intrusion and/or word intrusion tasks).

Solutions?

But some we can't solve everything.

- It's still BOW.
- We cannot incorporate any language knowledge from larger, pre-trained datasets (e.g., via embeddings)

$\Rightarrow$ If we think of the performance leap that we observe with Transformers in other areas, we have all reason to assume that we can do better.

Unsupervised ML

State-of-the-art approaches to topic modelling

Let's bring in embeddings and Transformers!

Using embeddings and transformers for topic modelling

For example:

- top2vec (Angelov, 2020), which embeds *topic vectors* in the same space as document vectors and word vectors
- Contextualized Topic models (Bianchi, Terragni, & Hovy, 2021; Bianchi, Terragni, Hovy, et al., 2021), with a lot of code examples at https://contextualized-topic-models.readthedocs.io/en/latest/introduction.html
- ...

Using embeddings and transformers for topic modelling

For example:

- top2vec (Angelov, 2020), which embeds *topic vectors* in the same space as document vectors and word vectors
- Contextualized Topic models (Bianchi, Terragni, & Hovy, 2021; Bianchi, Terragni, Hovy, et al., 2021), with a lot of code examples at https://contextualized-topic-models.readthedocs.io/en/latest/introduction.html
- ...

Using embeddings and transformers for topic modelling

For example:

- top2vec (Angelov, 2020), which embeds *topic vectors* in the same space as document vectors and word vectors
- Contextualized Topic models (Bianchi, Terragni, & Hovy, 2021; Bianchi, Terragni, Hovy, et al., 2021), with a lot of code examples at https://contextualized-topic-models.readthedocs.io/en/latest/introduction.html
- ...

BERTopic (Grootendorst, 2022)

"In this paper, we introduce BERTopic, a topic model that leverages clustering techniques and a class-based variation of TF-IDF to generate coherent topic representations. More specifically, we first create document embeddings using a pretrained language model to obtain document-level information. Second, we first reduce the dimensionality of document embeddings before creating semantically similar clusters of documents that each represent a distinct topic. Third, to overcome the centroid-based perspective, we develop a classbased version of TF-IDF to extract the topic representation from each topic. These three independent steps allow for a flexible topic model that can be used in a variety of use-cases, such as dynamic topic modeling."

(for details, read the paper)

BERTopic (Grootendorst, 2022)

Let's look at specific examples, for instance:
https://maartengr.github.io/BERTopic/getting_started/quickstart/quickstart.html

but also the visualization capabilites:
https://maartengr.github.io/BERTopic/getting_started/visualization/visualization.html#visualize-topics-per-class

Much more coherent topics than LDA!

	20 NewsGroups		BBC News		Trump	
	TC	TD	TC	TD	TC	TD
LDA	.058	.749	.014	.577	-.011	.502
NMF	.089	.663	.012	.549	.009	.379
T2V-*MPNET*	.068	.718	-.027	.540	-.213	.698
T2V-*Doc2Vec*	.192	.823	.171	.792	-.169	.658
CTM	.096	.886	.094	.819	.009	.855
BERTopic-*MPNET*	.166	.851	.167	.794	.066	.663

Table 1: Ranging from 10 to 50 topics with steps of 10, topic coherence (TC) and topic diversity (TD) were calculated at each step for each topic model. All results were averaged across 3 runs for each step. Thus, each score is the average of 15 separate runs.

(And no need to set k! And there is a dedicated “outlier topic” called -1!)

Are there downsides?

Of course!

- By definiton, much more “black-box”-y than BOW approaches
- Risk of biases introduced by LLM
- Much more resource-hungry (you probably want to do this with a GPU (e.g., on CoLab)

To conclude: PCA, k-means, LDA are interesting starting points – but if I were to start an unsupervised topic analysis model now, I'd go for BERTopic.