A Practical Introduction to Machine Learning in Python
Day 3 - Wednesday Afternoon
»Unsupervised Machine Learning«

Damian Trilling
Anne Kroon

d.c.trilling@uva.nl, @damian0604
a.c.kroon@uva.nl, @annekroon

September 29, 2021

Today

Recap: PCA and Clustering

LDA Topic models

- An introduction to LDA
- Choosing the best (or a good) topic model
- Using topic models
- Other forms of topic models

Next steps

Recap

Recap

PCA and Clustering

Let's assume we want to find out the topics in a large corpus of documents

We could either

- use PCA to find out related features (and interpret those as topics)
- or use clustering to find similar documents (and then look at the words they share to interpret as topics)

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?

Let's assume we want to find out the topics in a large corpus of documents

We could either

- use PCA to find out related features (and interpret those as topics)
- or use clustering to find similar documents (and then look at the words they share to interpret as topics)

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?

Recap: PCA

Document-term matrix

```
1       w1,w2,w3,w4,w5,w6 ...
2   text1, 2, 0, 0, 1, 2, 3 ...
3   text2, 0, 0, 1, 2, 3, 4 ...
4   text3, 9, 0, 1, 1, 0, 0 ...
5   ...
```

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.

- 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?

Recap: 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), cosine distances, etc.

- 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?

Recap: clustering

- given a term-document matrix, we can easily find clusters of documents that resemble each other
- but also here does the goal of cluster analysis, assigning each document to *one* cluster, match real life?

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; 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)

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

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; 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)

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

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; 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)

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, $T_1 \ldots T_k$
- 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`:

```
1 articles=['The tax deficit is higher than expected. This said xxx ...',
      'Germany won the World Cup. After a']
2 texts=[[token for token in re.split(r"\W", art) if len(token)>0] for art
       in articles]
```

which looks like this:

```
1 [['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)

```
1  0.069*fusie + 0.058*brussel + 0.045*europesecommissie + 0.036*europese +
       0.023*overname
2  0.109*bank + 0.066*britse + 0.041*regering + 0.035*financien + 0.033*
       minister
3  0.114*nederlandse + 0.106*nederland + 0.070*bedrijven + 0.042*rusland +
       0.038*russische
4  0.093*nederlandsespoorwegen + 0.074*den + 0.036*jaar + 0.029*onderzoek +
       0.027*raad
5  0.099*banen + 0.045*jaar + 0.045*productie + 0.036*ton + 0.029*aantal
6  0.041*grote + 0.038*bedrijven + 0.027*ondernemers + 0.023*goed + 0.015*
       jaar
7  0.108*werknemers + 0.037*jongeren + 0.035*werkgevers + 0.029*jaar +
       0.025*werk
8  0.171*bank + 0.122* + 0.041*klanten + 0.035*verzekeraar + 0.028*euro
9  0.162*banken + 0.055*bank + 0.039*centrale + 0.027*leningen + 0.024*
       financiele
10 0.052*post + 0.042*media + 0.038*nieuwe + 0.034*netwerk + 0.025*
       personeel
11 ...
```

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)*

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/damian0604/bdaca/blob/master/rm-course-2/week12/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()`)

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.

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 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, 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).

LDA Topic models

Other forms of topic models

Other forms of topic models

- Author-topic models
- Structural topic models
- Non-negative matrix factorization
- ...

Next steps

Exercise for this week

- Work through the example notebook on LDA: https://github.com/damian0604/bdaca/blob/master/12ec/week11/lda.ipynb
- But most importantly: **Use a dataset of your choice** and find a suitable topic model. You can also try to compare multiple approaches (e.g., clustering vs LDA). Possible inspiration: https://github.com/annekroon/gesis-ml-learning/blob/master/03wednesday/livecoding.ipynb

Next week: Word embeddings

Models that learn that 'teacher' and 'instructor' are more similar than 'teacher' and 'plumber'; or that 'king' - 'man' + 'woman' = 'queen'.

Final project

If you haven't done so yet, talk with me about your topic

Good luck with the take-home exam!