Tutorial: Topic Modeling

Niels Scholten | JM2050-M-6

Today

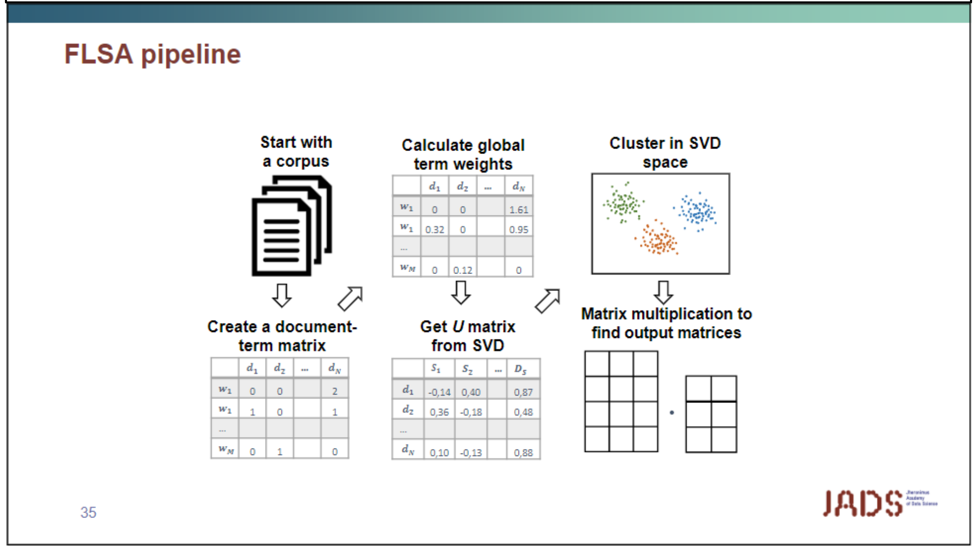
Learning Goals:

- Familiarize yourself with the training processs of different topic models
- Familiarize yourself with topic evaluations

FuzzyTM

FuzzyTM
A python library for training fuzzy topic models and creating topic embeddings for downstream tasks.

Last week



FLSA pipeline

Start with a corpus
Calculate global term weights
Cluster in SVD space
Create a document-term matrix
Get *U* matrix from SVD
Matrix multiplication to find output matrices

Create a document-term matrix

	d_1	d_2	...	d_N
w_1	0	0		2
w_1	1	0		1
...				
w_M	0	1		0

Calculate global term weights

	d_1	d_2	...	d_N
w_1	0	0		1.61
w_1	0.32	0		0.95
...				
w_M	0	0.12		0

Get *U* matrix from SVD

	S_1	S_2	...	D_S
d_1	-0,14	0,40		0,87
d_2	0,36	-0,18		0,48
...				
d_N	0,10	-0,13		0,88

35

FuzzyTM

- Algorithms
  - FLSA
  - **FLSA-W (For the assignment)**
  - FLSA-E
  - FLSA-V

The algorithms differ in the space in which they cluster

- Global-term weighting
  - Entropy
  - IDF
  - **Normal (Default)**
  - PorbIDF

- Clustering Algorithm
  - **FCM (Default)**
  - Gustafson-Kessel
  - FST-PSO

Jheronimus Academy of Data Science

Other useful methods

- Evaluation scores:
    - `get_coherence_score()`
    - `get_diversity_score()`
    - `get_interpretability_score()`
- Descriptive information:
    - `get_vocabulary()`
    - `get_vocabulary_size()`
- Embeddings:
    - `get_topic_embedding()`

Jheronimus Academy of Data Science

Calculate metrics for external topics!

```
flsaW.get_coherence_score(
                     data,
                     [['fax','modem','card’,],
                     [fax','disk','write',],],
                     )

flsaW.get_diversity_score(
                     data,
                     [['fax','modem','card’,],
                     [fax','disk','write',],],
                     )
```

Jheronimus Academy of Data Science

Example

Load data

```
from nltk.tokenize import word_tokenize
import re
from unidecode import unidecode
import pandas as pd
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
from collections import Counter

FILE_PATH = <your_path>
data = pd.read_csv(
              FILE_PATH,
              usecols=['Text’],
              )['Text'].tolist()
```

JADS Jheronimus Academy of Data Science

Preprocess data

```
def preprocess_texts(texts, n=0):
    processed_texts = []
        for text in texts:
            # lowercasing, keep text only, remove accents, tokenization
            tokens = [word for word in word_tokenize(re.sub(r'[^a-zA-Z\s]', ‘’, unidecode(text.lower())))]
            # stopword removal
            tokens = [token for token in tokens if token not in stopwords.words('english’)]
            processed_texts.append(tokens)

    # remove top-n% and bottom-n% words
    if n > 0:
        word_freq = Counter([word for sentence in processed_texts for word in sentence])
        top_n = set([word for word, _ in word_freq.most_common(int(n/100*len(word_freq)))])
        bottom_n = set([word for word, _ in word_freq.most_common()[:-int(n/100*len(word_freq))-1:-1]])
        processed_texts = [[word for word in sentence if word not in top_n and word not in bottom_n] for sentence in
processed_texts]

return processed_texts
```

Filter data and tokenize

Filter:

```
def filter_word_from_corpus(data, words):
    # Ensure words is a list, even if a single string is passed
    if isinstance(words, str):
        words = [words]
    # Filter words from data
        filtered_data = [[token for token in row if token not in words] for row in data]
    return filtered_data
```

Tokenize:

```
s = "This is a sample string."
tokens = s.split() # Splits on whitespace by default
print(tokens) # ['This', 'is', 'a', 'sample', 'string.’]
```

Jheronimus Academy of Data Science

Training FLSA-W with FuzzyTM

```
from FuzzyTM import FLSA_W
flsaW = FLSA_W(
    input_file = data,
    num_topics=10,
    num_words=10,
    )

flsaW.get_vocabulary_size()

pwgt, ptgd = flsaW.get_matrices() # THIS TRAINS THE MODEL

print(flsaW.show_topics())
for topic in flsaW.show_topics(representation='words’):
    print(topic)

print(flsaW.get_coherence_score())
print(flsaW.get_diversity_score())
print(flsaW.get_interpretability_score())
```

Jheronimus Academy of Data Science

Training an LDA Model with Gensim

```
import gensim
from gensim import corpora
from gensim.models.ldamodel import LdaModel
from gensim.models import CoherenceModel
dictionary = corpora.Dictionary(data)

# Convert the list of documents (corpus) into Document Term Matrix using the dictionary prepared above
doc_term_matrix = [dictionary.doc2bow(doc) for doc in data]

# Train the LDA model
lda_model = LdaModel(
    doc_term_matrix,
    num_topics=3,
    id2word=dictionary,
    )

topics = lda_model.print_topics()
# Print the topics
for topic in lda_model.print_topics():
    print(topic)

#Get coherence score
print(CoherenceModel(model=lda_model, texts=data, dictionary=dictionary, coherence='c_v').get_coherence())
```

Jheronimus Academy of Data Science

Training BERTopic

- If you do not use a GPU, this could take a while. Google Colab could speed it up

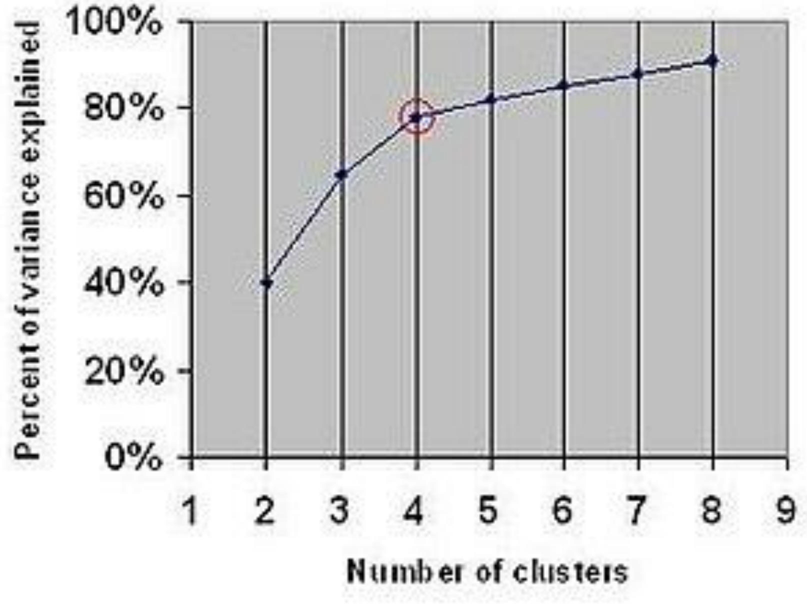
```
from bertopic import BERTopic
docs = [' '.join(doc) for doc in data]

# Create the model (uses DistilBERT by default)
bert_topic = BERTopic()

# Train the model and transform your data into topics
topic_assigned_to_doc, _ = bert_topic.fit_transform(docs)

topic_matrix = bert_topic.get_topic_info()
```

Elbow Plot

- Heuristic often used in clustering
- Can be used in FLSA-W to determine the number of clusters
- Find the number where the curve "breaks"

- However, you might not a nice curve like in the image

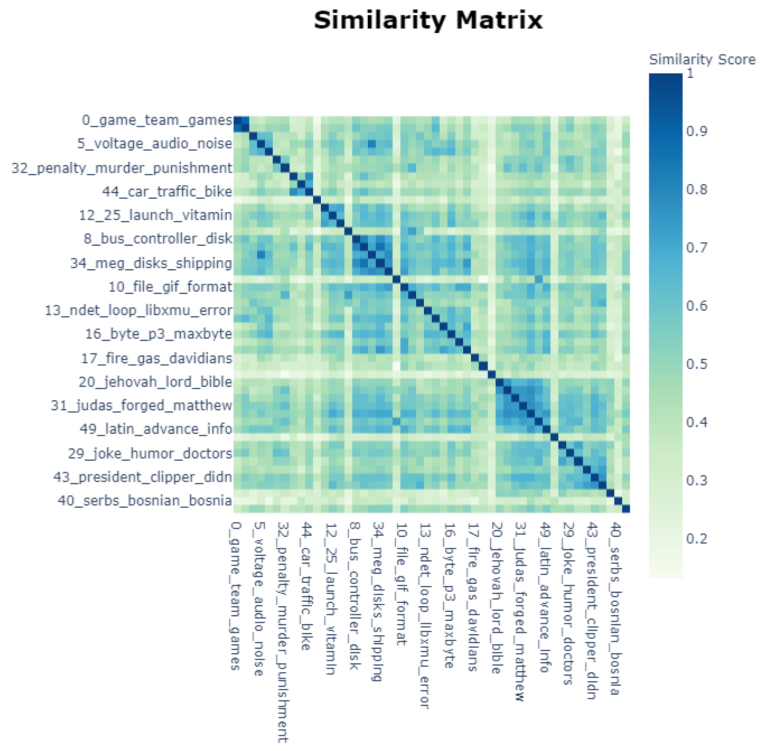
- For FLSA-W, use topic coherence instead for *Percent of variance explained.*



Topic Similarity Matrix

- Nice feature in Bertopic
- Calculates the cosine similarity between topic embeddings

```
Topic_model.visualize_heatmap()
```



Exercise

Niels Scholten | JM2050-M-6

JADS is powered by

For FLSA-W and LDA

Train topic models with both algorithms.

- Use an iterative process in which you:
    - train a topic model,
    - remove useless words from the corpus ,
    - retrain your model.

- How many iterations did it take to find your topics?
- What is the effect on the coherence score?

A few considerations:

- Setting the number of words per topics: Which number seems most intuitive?
- Setting the number of topics: How did you select this number?

Jheronimus Academy of Data Science

BERTopic

For the first ten documents in `topic_assigned_to_doc`, compare the produced topic to the document. Do you agree this document reflects the topic well?

What do you think -1 means?

How many words per topic do you have? What is the effect on the topic quality if you work with more topics?

JADS Jheronimus Academy of Data Science

All models

- Train all models with the same number of words per topic

- Calculate the coherence scores for all three models
  - How do the models compare?
  - Does your personal judgement align with the scores?

Interpreting Topics

- Ask ChatGPT to give names to the topics
- Are these descriptions good?
- How do you prompt ChatGPT to give good topic names and descriptions?