Open in Colab
(https://colab.research.google.com/github/hanhluukim/replication-topic-modelling-in-embedding-space/blob/main/notebook_replication.ipynb)

Das Projekt aus dem Github klonen und in den Projektsordner

In [1]:

```
#wenn die Ordner noch nicht geklont ist, soll dieser Fehler zuerst durchgeführt
 werden.
!git clone https://github.com/hanhluukim/replication-topic-modelling-in-embeddin
g-space.git
```

```
Cloning into 'replication-topic-modelling-in-embedding-space'...
remote: Enumerating objects: 2352, done.
remote: Counting objects: 100% (328/328), done.
remote: Compressing objects: 100% (247/247), done.
remote: Total 2352 (delta 170), reused 218 (delta 78), pack-reused
2024
Receiving objects: 100% (2352/2352), 531.47 MiB | 26.39 MiB/s, don
e.
Resolving deltas: 100% (1225/1225), done.
```

In [2]:

```
cd /content/replication-topic-modelling-in-embedding-space
```

```
/content/replication-topic-modelling-in-embedding-space
```

Das Trainieren über GPU: in dem Colab-runtime, wählen GPU

1. runtime/Laufzeit
2. change runtime type/Laufzeittyppen ändern
3. choose GPU and save/GPU auswählen und speichern

Die benötige Paketen für das Projekt mittels requirements.txt installieren

In [3]:

```
# Falls die Packages noch nicht installiert wurden,
!pip install -r "/content/replication-topic-modelling-in-embedding-space/require
ments.txt"
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.
dev/colab-wheels/public/simple/
Collecting gensim==3.8.3
  Downloading gensim-3.8.3-cp37-cp37m-manylinux1_x86_64.whl (24.2 M
B)
     |████████████████████████████████| 24.2 MB 1.5 MB/s
Requirement already satisfied: nltk in /usr/local/lib/python3.7/dis
t-packages (from -r /content/replication-topic-modelling-in-embeddi
ng-space/requirements.txt (line 2)) (3.2.5)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/di
st-packages (from -r /content/replication-topic-modelling-in-embedd
ing-space/requirements.txt (line 3)) (1.21.6)
Requirement already satisfied: scikit-learn in /usr/local/lib/pytho
n3.7/dist-packages (from -r /content/replication-topic-modelling-in
-embedding-space/requirements.txt (line 4)) (1.0.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/di
st-packages (from -r /content/replication-topic-modelling-in-embedd
ing-space/requirements.txt (line 5)) (1.4.1)
Requirement already satisfied: torch in /usr/local/lib/python3.7/di
st-packages (from -r /content/replication-topic-modelling-in-embedd
ing-space/requirements.txt (line 6)) (1.11.0+cu113)
Collecting transformers
  Downloading transformers-4.19.2-py3-none-any.whl (4.2 MB)
     |████████████████████████████████| 4.2 MB 59.1 MB/s
Collecting umap-learn
  Downloading umap-learn-0.5.3.tar.gz (88 kB)
     |████████████████████████████████| 88 kB 8.6 MB/s
Collecting plotly==5.7.0
  Downloading plotly-5.7.0-py2.py3-none-any.whl (28.8 MB)
     |████████████████████████████████| 28.8 MB 101.6 MB/s
Requirement already satisfied: pathlib in /usr/local/lib/python3.7/
dist-packages (from -r /content/replication-topic-modelling-in-embe
dding-space/requirements.txt (line 10)) (1.0.1)
Collecting pyyaml==5.4.1
  Downloading PyYAML-5.4.1-cp37-cp37m-manylinux1_x86_64.whl (636 k
B)
     |████████████████████████████████| 636 kB 52.7 MB/s
Collecting kaleido
  Downloading kaleido-0.2.1-py2.py3-none-manylinux1_x86_64.whl (79.
9 MB)
     |████████████████████████████████| 79.9 MB 115 kB/s
Requirement already satisfied: torchvision in /usr/local/lib/python
3.7/dist-packages (from -r /content/replication-topic-modelling-in-
embedding-space/requirements.txt (line 13)) (0.12.0+cu113)
Requirement already satisfied: pandas in /usr/local/lib/python3.7/d
ist-packages (from -r /content/replication-topic-modelling-in-embed
ding-space/requirements.txt (line 14)) (1.3.5)
Requirement already satisfied: six>=1.5.0 in /usr/local/lib/python
3.7/dist-packages (from gensim==3.8.3->-r /content/replication-topi
c-modelling-in-embedding-space/requirements.txt (line 1)) (1.15.0)
Requirement already satisfied: smart-open>=1.8.1 in /usr/local/lib/
python3.7/dist-packages (from gensim==3.8.3->-r /content/replicatio
n-topic-modelling-in-embedding-space/requirements.txt (line 1)) (6.
0.0)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/py
thon3.7/dist-packages (from plotly==5.7.0->-r /content/replication-
topic-modelling-in-embedding-space/requirements.txt (line 9)) (8.0.
1)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/pytho
n3.7/dist-packages (from scikit-learn->-r /content/replication-topi
c-modelling-in-embedding-space/requirements.txt (line 4)) (1.1.0)
```

```
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/l
ib/python3.7/dist-packages (from scikit-learn->-r /content/replicat
ion-topic-modelling-in-embedding-space/requirements.txt (line 4))
(3.1.0)
Requirement already satisfied: typing-extensions in /usr/local/lib/
python3.7/dist-packages (from torch->-r /content/replication-topic-
modelling-in-embedding-space/requirements.txt (line 6)) (4.2.0)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/py
thon3.7/dist-packages (from transformers->-r /content/replication-t
opic-modelling-in-embedding-space/requirements.txt (line 7)) (21.3)
Requirement already satisfied: importlib-metadata in /usr/local/li
b/python3.7/dist-packages (from transformers->-r /content/replicati
on-topic-modelling-in-embedding-space/requirements.txt (line 7))
(4.11.3)
Requirement already satisfied: filelock in /usr/local/lib/python3.
7/dist-packages (from transformers->-r /content/replication-topic-m
odelling-in-embedding-space/requirements.txt (line 7)) (3.7.0)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python
3.7/dist-packages (from transformers->-r /content/replication-topic
-modelling-in-embedding-space/requirements.txt (line 7)) (4.64.0)
Requirement already satisfied: requests in /usr/local/lib/python3.
7/dist-packages (from transformers->-r /content/replication-topic-m
odelling-in-embedding-space/requirements.txt (line 7)) (2.23.0)
Collecting tokenizers!=0.11.3,<0.13,>=0.11.1
  Downloading tokenizers-0.12.1-cp37-cp37m-manylinux_2_12_x86_64.ma
nylinux2010_x86_64.whl (6.6 MB)
     |████████████████████████████████| 6.6 MB 53.5 MB/s
Collecting huggingface-hub<1.0,>=0.1.0
  Downloading huggingface_hub-0.7.0-py3-none-any.whl (86 kB)
     |████████████████████████████████| 86 kB 7.4 MB/s
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/
python3.7/dist-packages (from transformers->-r /content/replication
-topic-modelling-in-embedding-space/requirements.txt (line 7)) (201
9.12.20)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/loc
al/lib/python3.7/dist-packages (from packaging>=20.0->transformers-
>-r /content/replication-topic-modelling-in-embedding-space/require
ments.txt (line 7)) (3.0.9)
Requirement already satisfied: numba>=0.49 in /usr/local/lib/python
3.7/dist-packages (from umap-learn->-r /content/replication-topic-m
odelling-in-embedding-space/requirements.txt (line 8)) (0.51.2)
Collecting pynndescent>=0.5
  Downloading pynndescent-0.5.7.tar.gz (1.1 MB)
     |████████████████████████████████| 1.1 MB 56.1 MB/s
Requirement already satisfied: llvmlite<0.35,>=0.34.0.dev0 in /usr/
local/lib/python3.7/dist-packages (from numba>=0.49->umap-learn->-r
/content/replication-topic-modelling-in-embedding-space/requirement
s.txt (line 8)) (0.34.0)
Requirement already satisfied: setuptools in /usr/local/lib/python
3.7/dist-packages (from numba>=0.49->umap-learn->-r /content/replic
ation-topic-modelling-in-embedding-space/requirements.txt (line 8))
(57.4.0)
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in /usr/local/
lib/python3.7/dist-packages (from torchvision->-r /content/replicat
ion-topic-modelling-in-embedding-space/requirements.txt (line 13))
(7.1.2)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/loca
l/lib/python3.7/dist-packages (from pandas->-r /content/replication
-topic-modelling-in-embedding-space/requirements.txt (line 14)) (2.
8.2)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/pytho
```

```
n3.7/dist-packages (from pandas->-r /content/replication-topic-mode
lling-in-embedding-space/requirements.txt (line 14)) (2022.1)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.
7/dist-packages (from importlib-metadata->transformers->-r /conten
t/replication-topic-modelling-in-embedding-space/requirements.txt
(line 7)) (3.8.0)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/pytho
n3.7/dist-packages (from requests->transformers->-r /content/replic
ation-topic-modelling-in-embedding-space/requirements.txt (line 7))
(2.10)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/li
b/python3.7/dist-packages (from requests->transformers->-r /conten
t/replication-topic-modelling-in-embedding-space/requirements.txt
(line 7)) (2022.5.18.1)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/
python3.7/dist-packages (from requests->transformers->-r /content/r
eplication-topic-modelling-in-embedding-space/requirements.txt (lin
e 7)) (3.0.4)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.2
1.1 in /usr/local/lib/python3.7/dist-packages (from requests->trans
formers->-r /content/replication-topic-modelling-in-embedding-spac
e/requirements.txt (line 7)) (1.24.3)
Building wheels for collected packages: umap-learn, pynndescent
  Building wheel for umap-learn (setup.py) ... done
  Created wheel for umap-learn: filename=umap_learn-0.5.3-py3-none-
any.whl size=82829 sha256=6ce95f2dd3c9a755a144ad0892d993e2d130586c7
71e420fe7b8e9001bf9d6bf
  Stored in directory: /root/.cache/pip/wheels/b3/52/a5/1fd9e3e76a7
ab34f134c07469cd6f16e27ef3a37aeff1fe821
  Building wheel for pynndescent (setup.py) ... done
  Created wheel for pynndescent: filename=pynndescent-0.5.7-py3-non
e-any.whl size=54286 sha256=58b51a31787d7621de51508a1a1e64357025881
11cb3271feaed6e844f520db9
  Stored in directory: /root/.cache/pip/wheels/7f/2a/f8/7bd5dcec71b
d5c669f6f574db3113513696b98f3f9b51f496c
Successfully built umap-learn pynndescent
Installing collected packages: pyyaml, tokenizers, pynndescent, hug
gingface-hub, umap-learn, transformers, plotly, kaleido, gensim
  Attempting uninstall: pyyaml
    Found existing installation: PyYAML 3.13
    Uninstalling PyYAML-3.13:
      Successfully uninstalled PyYAML-3.13
  Attempting uninstall: plotly
    Found existing installation: plotly 5.5.0
    Uninstalling plotly-5.5.0:
      Successfully uninstalled plotly-5.5.0
  Attempting uninstall: gensim
    Found existing installation: gensim 3.6.0
    Uninstalling gensim-3.6.0:
      Successfully uninstalled gensim-3.6.0
Successfully installed gensim-3.8.3 huggingface-hub-0.7.0 kaleido-
0.2.1 plotly-5.7.0 pynndescent-0.5.7 pyyaml-5.4.1 tokenizers-0.12.1
transformers-4.19.2 umap-learn-0.5.3
```

Struktur

In diesem Notebook kann man zwei Versionen durchführen:

1. durch command lines (LDA Command, ETM-Command, BERT-ETM Command)
2. Notebook für alle Schritten nach und nach durchführen
3. Wenn nur Notebook benutzen möchte, springen zum Teil Notebook

Für Notebooks:

Wenn jemand Notebooks benutzt, um alle Schritten anzuschauen, bitte überspringen den Teil, in den Command-Lines sich befinden

Command-Teil

LDA Command

1. batch-test-size: Testdataset wird zu kleineren Batches mit batch-size-test zerlegt und dann Perplexity berechnen.
2. die *Endperplexity* ist der Durchschnitt von allen Batch-Perplexites

In [ ]:

```
run main_lda.py --filter-stopwords "True" --min-df 30 --epochs 20 --use-tensor T
rue --batch-test-size 1000
```

```
filter stopwords: True
filter stopwords: True
loading texts: ...
From: lerxst@wam.umd.edu (where's my thing)
Subject: WHAT car is this!?
Nntp-Posting-Host: rac3.wam.umd.edu
Organization: University of Maryland, College Park
Lines: 15

 I was wondering if anyone out there could enlighten me on this car
I saw
the other day. It was a 2-door sports car, looked to be from the la
te 60s/
early 70s. It was called a Bricklin. The doors were really small. I
n addition,
the front bumper was separate from the rest of the body. This is
all I know. If anyone can tellme a model name, engine specs, years
of production, where this car is made, history, or whatever info yo
u
have on this funky looking car, please e-mail.

Thanks,
- IL
   ---- brought to you by your neighborhood Lerxst ----





train-size after loading: 11314
test-size after loading: 7532
finished load!
start: preprocessing: ...
preprocessing step: remove stopwords
finised: preprocessing!
vocab-size in df: 8496
preprocessing remove stopwords from vocabulary
start creating vocabulary ...
length of the vocabulary: 8496
length word2id list: 8496
length id2word list: 8496
finished: creating vocabulary
save docs in txt...
save docs finished
train-size-after-all: 11214
test-size-after-all: 7532
validation-size-after-all: 100
test-size-after-all: 11214
test-indices-length: 11214
test-size-after-all: 100
test-indices-length: 100
test-size-after-all: 7532
test-indices-length: 7532
length train-documents-indices : 1150368
length of the vocabulary: 8496


start: creating bow representation...
finised creating bow input!

start: creating bow representation...
```

```
finised creating bow input!

start: creating bow representation...
finised creating bow input!

start: creating bow representation...
finised creating bow input!

start: creating bow representation...
finised creating bow input!

compact representation for LDA
save docs in txt...
save docs finished
run LDA training...

100%|██████████| 20/20 [00:00<00:00, 49402.87it/s]

number of topics: 20
calculate perplexity:....


test-docs: from 0 to 1000
ppl of batch 1: 8.150738847141124
test-docs: from 1000 to 2000
ppl of batch 2: 8.148755636492822
test-docs: from 2000 to 3000
ppl of batch 3: 8.146552472151644
test-docs: from 3000 to 4000
ppl of batch 4: 8.133709139763923
test-docs: from 4000 to 5000
ppl of batch 5: 8.144285514676735
test-docs: from 5000 to 6000
ppl of batch 6: 8.1097638767229
test-docs: from 6000 to 7000
ppl of batch 7: 8.155350564571188
test-docs: from 7000 to 7532
ppl of batch 8: 8.170441779038889
end perplexity - show perplexity:
e-normalized-perplexity-lda: 0.40559086629001884
calculate coherence and diversity
topic coherence 0.17668937635583054
topic diversity 0.754
ending coherence and diversity
```

Skipgram-ETM

1. epochs
2. wordvec-model: skipgram oder cbow
3. min-df
4. filter-stopwords: "True"/"False" as String
5. activate-func: "ReLU" oder "tanh"
6. hidden-size:
7. optimizer-name: "adam" oder "sgd"
8. learing rate: lr
9. weight-decay wdecay

In [ ]:

```
run main.py --model "ETM" --epochs 150 --wordvec-model "skipgram" --loss-name "cross-entropy" --min-df 100 --num-topics 20 --filter-stopwords "True" --hidden-size 800 --activate-func "ReLU" --optimizer-name "adam" --lr 0.002 --wdecay 0.0000012
```

BERT-ETM Command

1. für Bert nur in dem Fall, dass Datensaz ohne Stopwörter ist das Durchführen möglich

In [ ]:

```
run main.py --model "ETM" --epochs 2 --wordvec-model "bert" --loss-name "cross-entropy" --min-df 10 --num-topics 20 --filter-stopwords "True" --hidden-size 800 --activate-func "ReLU" --optimizer-name "adam" --lr 0.002 --wdecay 0.0000012
```

```
using cuda: False
filter-stopwords: True
--------------------------------------------------------------------
---------------------------------


loading texts: ...
From: lerxst@wam.umd.edu (where's my thing)
Subject: WHAT car is this!?
Nntp-Posting-Host: rac3.wam.umd.edu
Organization: University of Maryland, College Park
Lines: 15

 I was wondering if anyone out there could enlighten me on this car
I saw
the other day. It was a 2-door sports car, looked to be from the la
te 60s/
early 70s. It was called a Bricklin. The doors were really small. I
n addition,
the front bumper was separate from the rest of the body. This is
all I know. If anyone can tellme a model name, engine specs, years
of production, where this car is made, history, or whatever info yo
u
have on this funky looking car, please e-mail.

Thanks,
- IL
   ---- brought to you by your neighborhood Lerxst ----





train-size after loading: 11314
test-size after loading: 7532
finished load!
start: preprocessing: ...
preprocessing step: remove stopwords
will use bert embedding, so delete words from not_in_bert_vocab.txt
finised: preprocessing!

total documents 18846
vocab-size in df: 18637
preprocessing remove stopwords from vocabulary
start creating vocabulary ...
length of the vocabulary: 18637
length word2id list: 18637
length id2word list: 18637
finished: creating vocabulary


save docs in txt...
save docs finished
train-size-after-all: 11214
test-size-after-all: 7532
validation-size-after-all: 100
test-size-after-all: 11214
test-indices-length: 11214
test-size-after-all: 100
test-indices-length: 100
test-size-after-all: 7532
```

```
test-indices-length: 7532
length train-documents-indices : 1295140
length of the vocabulary: 18637


start: creating bow representation...
finised creating bow input!

start: creating bow representation...
finised creating bow input!

start: creating bow representation...
finised creating bow input!

start: creating bow representation...
finised creating bow input!

start: creating bow representation...
finised creating bow input!

train-bow-representation for ETM:

example ids of dict-id2word for ETM: [0, 1, 2, 3, 4]
example words of dict-id2word for ETM: ['bobbs', 'opposing', 'gm',
'xterminals', 'trim']
Size of the vocabulary after prprocessing ist: 18637
Size of train set: 11214
Size of val set: 100
Size of test set: 7532
save docs in txt...
save docs finished
prepare data finished
-------------------------------------------------------------------
---------------------------------
word-embedding training begin
using prepared_data/bert_vocab_embedding.txt
bert-embeddings were already builded
word-embedding finised
-------------------------------------------------------------------
---------------------------------
training parameter setting...
using epochs: 2
using optimizer: adam
using learning rate: 0.002
using wdecay: 1.2e-06
total train docs: 11214
sum of vector: 0.9999996423721313
length of vector: 0.14386114478111267
reading bert prefitted-embedding...
prepared_data//bert_vocab_embedding.txt
loading bert from npy and pkl

iter over bert-vocab:   0%|          | 84/104008 [00:00<02:04, 837.
68it/s]

bert-reading finished
update bert by given vocab

iter over bert-vocab: 100%|██████████| 104008/104008 [01:27<00:00,
1185.24it/s]
```

```
example 5 element of word-vector: [-0.29880613 -2.86340356 -0.14214
839 -1.84090602  3.61773515]
ETM initilize...
--------------------------------------------------MODEL-SUMMARY----
----------------------------------------------
ETM(
  (theta_act): ReLU()
  (topic_embeddings_alphas): Linear(in_features=768, out_features=2
0, bias=False)
  (q_theta): Sequential(
    (0): Linear(in_features=18637, out_features=800, bias=True)
    (1): ReLU()
    (2): Linear(in_features=800, out_features=800, bias=True)
    (3): ReLU()
  )
  (mu_q_theta): Linear(in_features=800, out_features=20, bias=True)
  (logsigma_q_theta): Linear(in_features=800, out_features=20, bias
=True)
)
--------------------------------------------------TRAIN------------
--------------------------------------
number of batches: 12
Epoch: 1/2  -  Loss: 1097.84509          Rec: 1097.45032         K
L: 0.39488
Epoch: 2/2  -  Loss: 1036.34802          Rec: 1035.17322         K
L: 1.17482
Checkpoint saved at checkpoints/etm_epoch_2.pth.tar

<Figure size 640x480 with 1 Axes>

<Figure size 640x480 with 1 Axes>

100%|██████████| 20/20 [00:00<00:00, 138425.87it/s]

topic-coherrence: 0.09996242525380584
topic-diversity: 0.508
calculate perplexitiy of test dataset: ...
test-1-loader: 7
test-2-loader: 7
batch 0 finished
batch 1 finished
batch 2 finished
batch 3 finished
batch 4 finished
batch 5 finished
batch 6 finished
topic-normalized-perplexity: 0.5014701937007029
```

Notebooks für alle Schritten: LDA und ETM

1. Da der Umfang der Implementierung ziemlich große ist, wird die Implementierung für unterschiedliche Komponenten in dem Ordner `src` gespeichert [hier: (https://github.com/hanhluukim/replication-topic-modelling-in-embedding-space/tree/main/src)](https://github.com/hanhluukim/replication-topic-modelling-in-embedding-space/tree/main/src)
2. Die gebrachten Komponenten werden in diesem Notebook dann importieren und die Ausgaben werden angezeigt.

Gebrauchte Paketen importieren

In [4]:

```
# einige Paketten wurden für Visualisierung gebraucht
import pandas as pd
from pathlib import Path
import matplotlib.pyplot as plt
%matplotlib inline
import umap.umap_ as umap
import time
import plotly.express as px
from sklearn import cluster
from sklearn import metrics
```

```
/usr/local/lib/python3.7/dist-packages/distributed/config.py:20: YA
MLLoadWarning: calling yaml.load() without Loader=... is deprecate
d, as the default Loader is unsafe. Please read https://msg.pyyaml.
org/load for full details.
  defaults = yaml.load(f)
```

Vorverarbeitung und BOW-Repräsentationen für Textdaten durchführen

1. Vocabular erstellen
2. BOW-Repräsentationen für allen Teildatensätzen
3. Wichtige Parameters sind:

- stopwords_filter = True/False
- use_bert_embedding = True/False
- min_df für unterschiedliche Vocabulargröße
- stopwords_remove_from_vocab = True/False

In [5]:

```
use_bert_embedding = False #in this notebook we do not use BERT-Embeddings
stopwords_filter = True
```

In [6]:

```
# init TextDataLoader für die Datenquelle 20 News Groups
# Daten abrufen vom Sklearn, tokenisieren und besondere Charaktern entfernen
from src.prepare_dataset import TextDataLoader
textsloader = TextDataLoader(source="20newsgroups", train_size=None, test_size=N
one)
textsloader.load_tokenize_texts("20newsgroups")
textsloader.show_example_raw_texts(n_docs=2)
# Vorverarbeitung von Daten mit folgenden Schritten:
textsloader.preprocess_texts(length_one_remove=True,
                             punctuation_lower = True,
                             stopwords_filter = stopwords_filter,
                             use_bert_embedding = use_bert_embedding)
# Daten zerlegen für Train, Test und Validation. Erstellen Vocabular aus dem Tra
inset

min_df=100
textsloader.split_and_create_voca_from_trainset(max_df=0.7,
                                                min_df=min_df,
                                                stopwords_remove_from_voca=stopw
ords_filter)
```

```
loading texts: ...
From: lerxst@wam.umd.edu (where's my thing)
Subject: WHAT car is this!?
Nntp-Posting-Host: rac3.wam.umd.edu
Organization: University of Maryland, College Park
Lines: 15

 I was wondering if anyone out there could enlighten me on this car
I saw
the other day. It was a 2-door sports car, looked to be from the la
te 60s/
early 70s. It was called a Bricklin. The doors were really small. I
n addition,
the front bumper was separate from the rest of the body. This is
all I know. If anyone can tellme a model name, engine specs, years
of production, where this car is made, history, or whatever info yo
u
have on this funky looking car, please e-mail.

Thanks,
- IL
   ---- brought to you by your neighborhood Lerxst ----





train-size after loading: 11314
test-size after loading: 7532
finished load!
check some sample texts of the dataset after filter punctuation and
digits
['From', ':', 'lerxst', '@', 'wam', '.', 'umd', '.', 'edu', '(', "w
here's", 'my', 'thing', ')', 'Subject', ':', 'WHAT', 'car', 'is',
'this', '!', '?', 'Nntp', 'Posting', 'Host', ':', 'rac3', '.', 'wa
m', '.', 'umd', '.', 'edu', 'Organization', ':', 'University', 'o
f', 'Maryland', ',', 'College', 'Park', 'Lines', ':', '15', 'I', 'w
as', 'wondering', 'if', 'anyone', 'out', 'there', 'could', 'enlight
en', 'me', 'on', 'this', 'car', 'I', 'saw', 'the', 'other', 'day',
'.', 'It', 'was', 'a', '2', 'door', 'sports', 'car', ',', 'looked',
'to', 'be', 'from', 'the', 'late', '60s', '/', 'early', '70s', '.',
'It', 'was', 'called', 'a', 'Bricklin', '.', 'The', 'doors', 'wer
e', 'really', 'small', '.', 'In', 'addition', ',', 'the', 'front',
'bumper', 'was', 'separate', 'from', 'the', 'rest', 'of', 'the', 'b
ody', '.', 'This', 'is', 'all', 'I', 'know', '.', 'If', 'anyone',
'can', 'tellme', 'a', 'model', 'name', ',', 'engine', 'specs', ',',
'years', 'of', 'production', ',', 'where', 'this', 'car', 'is', 'ma
de', ',', 'history', ',', 'or', 'whatever', 'info', 'you', 'have',
'on', 'this', 'funky', 'looking', 'car', ',', 'please', 'e', 'mai
l', '.', 'Thanks', ',', 'IL', 'brought', 'to', 'you', 'by', 'your',
'neighborhood', 'Lerxst']
====================================================================
=================================
['From', ':', 'guykuo', '@', 'carson', '.', 'u', '.', 'washington',
'.', 'edu', '(', 'Guy', 'Kuo', ')', 'Subject', ':', 'SI', 'Clock',
'Poll', 'Final', 'Call', 'Summary', ':', 'Final', 'call', 'for', 'S
I', 'clock', 'reports', 'Keywords', ':', 'SI', ',', 'acceleration',
',', 'clock', ',', 'upgrade', 'Article', 'I', '.', 'D', '.', ':',
'shelley', '.', '1qvfo9INNc3s', 'Organization', ':', 'University',
'of', 'Washington', 'Lines', ':', '11', 'NNTP', 'Posting', 'Host',
':', 'carson', '.', 'u', '.', 'washington', '.', 'edu', 'A', 'fai
```

```
r', 'number', 'of', 'brave', 'souls', 'who', 'upgraded', 'their',
'SI', 'clock', 'oscillator', 'have', 'shared', 'their', 'experience
s', 'for', 'this', 'poll', '.', 'Please', 'send', 'a', 'brief', 'me
ssage', 'detailing', 'your', 'experiences', 'with', 'the', 'procedu
re', '.', 'Top', 'speed', 'attained', ',', 'CPU', 'rated', 'speed',
',', 'add', 'on', 'cards', 'and', 'adapters', ',', 'heat', 'sinks',
',', 'hour', 'of', 'usage', 'per', 'day', ',', 'floppy', 'disk', 'f
unctionality', 'with', '800', 'and', '1', '.', '4', 'm', 'floppie
s', 'are', 'especially', 'requested', '.', 'I', 'will', 'be', 'summ
arizing', 'in', 'the', 'next', 'two', 'days', ',', 'so', 'please',
'add', 'to', 'the', 'network', 'knowledge', 'base', 'if', 'you', 'h
ave', 'done', 'the', 'clock', 'upgrade', 'and', "haven't", 'answere
d', 'this', 'poll', '.', 'Thanks', '.', 'Guy', 'Kuo', '<', 'guyku
o', '@', 'u', '.', 'washington', '.', 'edu', '>']
====================================================================
=================================
start: preprocessing: ...
preprocessing step: remove stopwords
finised: preprocessing!
vocab-size in df: 3102
preprocessing remove stopwords from vocabulary
start creating vocabulary ...
length of the vocabulary: 3102
length word2id list: 3102
length id2word list: 3102
finished: creating vocabulary
```

LDA Model

1. Benutzen das fertige Paket von Gensim, um die Topics mit LDA zu finden: LDA Model GENSIM (https://radimrehurek.com/gensim/models/ldamodel.html)
2. Der Klasse textsloader hat bereits die geeignete Format für LDA vorbereitet:

- Setzen das Parameter: for_lda_model = True

In [7]:

```
from src.evaluierung import topicCoherence2, topicDiversity
from src.lda import lda
from gensim.models import LdaModel
from gensim.parsing.preprocessing import preprocess_string, strip_punctuation, s
trip_numeric


for_lda_model = True
num_topics = 20

# Erstellen BOW-Repräsentation für LDA Model
if for_lda_model == True:
    word2id, id2word, train_set, test_set, val_set, test_set_h1, test_set_h2 = t
extsloader.create_bow_and_savebow_for_each_set(for_lda_model=for_lda_model)
    gensim_corpus_train_set = train_set
else:
    print('for_lda_model is True but still here?')
    word2id, id2word, train_set, test_set, val_set = textsloader.create_bow_and_
savebow_for_each_set(for_lda_model=for_lda_model)

docs_tr, docs_t, docs_v = textsloader.get_docs_in_words_for_each_set()
#lda model
print(100*"-")
```

```
save docs in txt...
save docs finished
train-size-after-all: 11214
test-size-after-all: 7532
validation-size-after-all: 100
test-size-after-all: 11214
test-indices-length: 11214
test-size-after-all: 100
test-indices-length: 100
test-size-after-all: 7532
test-indices-length: 7532
length train-documents-indices : 896087
length of the vocabulary: 3102


start: creating bow representation...
finised creating bow input!

start: creating bow representation...
finised creating bow input!

start: creating bow representation...
finised creating bow input!

start: creating bow representation...
finised creating bow input!

start: creating bow representation...
finised creating bow input!

compact representation for LDA
save docs in txt...
save docs finished
--------------------------------------------------------------------
---------------------------------
```

In [8]:

```
#ldamodel = lda(train_set,10,id2word)
ldamodel = LdaModel(train_set,
                    num_topics= num_topics,
                    id2word = id2word,
                    passes = 5,
                    random_state = 42)
```

In [9]:

```
lda_topics = ldamodel.show_topics(num_topics = 20, num_words=10)
topics = []
filters = [lambda x: x.lower(), strip_punctuation, strip_numeric]
for topic in lda_topics:
    topics.append(preprocess_string(topic[1], filters))
for topic in topics:
  print(topic)
```

```
['windows', 'file', 'dos', 'graphics', 'software', 'pc', 'system',
'files', 'ftp', 'program']
['god', 'people', 'jesus', 'christian', 'bible', 'life', 'church',
'christ', 'christians', 'time']
['writes', 'article', 'cs', 'university', 'colorado', 'posting', 'c
c', 'nntp', 'host', 'science']
['ca', 'uk', 'ac', 'writes', 'article', 'org', 'posting', 'nntp',
'host', 'mit']
['drive', 'sale', 'scsi', 'disk', 'hp', 'hard', 'drives', 'system',
'ide', 'computer']
['window', 'information', 'data', 'application', 'source', 'time',
'widget', 'set', 'include', 'list']
['brian', 'indiana', 'gatech', 'apple', 'article', 'ucs', 'writes',
'georgia', 'sandvik', 'kent']
['university', 'posting', 'host', 'nntp', 'de', 'au', 'computer',
'writes', 'article', 'distribution']
['money', 'people', 'time', 'car', 'make', 'pay', 'work', 'year',
'list', 'good']
['power', 'state', 'ohio', 'back', 'acs', 'time', 'ground', 'home',
'work', 'left']
['andrew', 'cmu', 'washington', 'att', 'posting', 'ibm', 'host', 'n
ntp', 'san', 'la']
['israel', 'jews', 'armenians', 'armenian', 'people', 'war', 'turki
sh', 'jewish', 'israeli', 'world']
['writes', 'article', 'posting', 'nntp', 'host', 'university', 'bik
e', 'sun', 'world', 'distribution']
['people', 'mr', 'writes', 'fire', 'president', 'fbi', 'time', 'art
icle', 'police', 'koresh']
['space', 'nasa', 'gov', 'access', 'mil', 'digex', 'net', 'shuttl
e', 'launch', 'pat']
['max', 'car', 'cwru', 'writes', 'cleveland', 'caltech', 'article',
'posting', 'se', 'nntp']
['medical', 'information', 'research', 'health', 'disease', 'nation
al', 'april', 'school', 'number', 'cancer']
['people', 'article', 'gun', 'law', 'writes', 'government', 'right
s', 'guns', 'control', 'make']
['game', 'team', 'year', 'games', 'writes', 'uiuc', 'hockey', 'seas
on', 'university', 'article']
['key', 'netcom', 'chip', 'clipper', 'encryption', 'keys', 'publi
c', 'des', 'government', 'security']
```

In [10]:

```
tc = topicCoherence2(topics,len(topics),docs_tr,len(docs_tr))
td = topicDiversity(topics)
print(f'topic-coherrence: {tc}')
print(f'topic-diversity: {td}')
```

```
topic-coherrence: 0.18777587945253085
topic-diversity: 0.74
```

Perplexity for LDA

In [11]:

```
from src.evaluierung import topicPerplexityTeil1, topicPerplexityNew
from src.utils_perplexity import get_theta_from_lda, get_beta_from_lda
import gensim
vocab = list(id2word.values())
vocab_size=len(vocab)
# get beta and theta
beta_KV = get_beta_from_lda(ldamodel, num_topics, vocab, vocab_size)
theta_test_1_DK = get_theta_from_lda(ldamodel, num_topics, test_set_h1)
n_test_docs_2 = len(test_set_h2)
test_set_h2_in_bow_sparse_matrix = gensim.matutils.corpus2csc(test_set_h2).trans
pose()
```

In [14]:

```
#covert to tensor
import math
import torch
import numpy as np

ppl_over_batches = []
batch_test_size = 1000
beta_KV = torch.from_numpy(np.array(beta_KV))
i = 0
j = 0
while i <= n_test_docs_2:
    if (i+batch_test_size) <= n_test_docs_2:
        theta_test_1_batch = torch.tensor(theta_test_1_DK[i:i+batch_test_size])
        bows_test_2_batch = torch.from_numpy(test_set_h2_in_bow_sparse_matrix[i:
i+batch_test_size].toarray())
    else:

        theta_test_1_batch = torch.tensor(theta_test_1_DK[i:])
        bows_test_2_batch = torch.from_numpy(test_set_h2_in_bow_sparse_matrix[i
:].toarray())

    avg_ppl = topicPerplexityNew(theta_test_1_batch, bows_test_2_batch, vocab_si
ze, beta_KV)
    print(f'ppl of batch {j+1}: {avg_ppl}')
    ppl_over_batches.append(avg_ppl)
    i = i + batch_test_size
    j += 1
avg_over_batches = (sum(ppl_over_batches)/len(ppl_over_batches))
ppl_total = round(math.exp(avg_over_batches),1)
normalized_ppl = ppl_total/vocab_size

print(f'end perplexity - show perplexity: ')
print(f'e-normalized-perplexity-lda: {normalized_ppl}')
ldamodel.clear()
```

```
ppl of batch 1: 7.4626675565497775
ppl of batch 2: 7.459199193874847
ppl of batch 3: 7.4777010005196125
ppl of batch 4: 7.472666592536151
ppl of batch 5: 7.489781073904341
ppl of batch 6: 7.457684657228188
ppl of batch 7: 7.48213299047086
ppl of batch 8: 7.471542143482913
end perplexity - show perplexity:
e-normalized-perplexity-lda: 0.5665699548678272
```

In [15]:

```
del test_set_h2
del test_set_h1
del test_set_h2_in_bow_sparse_matrix
del theta_test_1_DK
del beta_KV
del ppl_over_batches
del batch_test_size
```

Alle Schritten im Experiment mit dem ETM-Modell

Daten für ETM

1. Input Daten für der ersten Teil ETM ist (normalisierte)Bag-Of-Words-Repräsentation
   - for_lda_model = False
2. textsloader.create_bow_and_savebow_for_each_set(for_lda_model=True) stellt die folgenden Daten für das Modell:
   - word2id
   - id2word
   - train_set, test_set, val_set in der Form von BoW

In [16]:

```
# Erstellen BOW-Repräsentation für ETM Modell
for_lda_model = False
word2id, id2word, train_set, test_set, val_set = textsloader.create_bow_and_save
bow_for_each_set(for_lda_model=for_lda_model)
```

```
save docs in txt...
save docs finished
train-size-after-all: 11214
test-size-after-all: 7532
validation-size-after-all: 100
test-size-after-all: 11214
test-indices-length: 11214
test-size-after-all: 100
test-indices-length: 100
test-size-after-all: 7532
test-indices-length: 7532
length train-documents-indices : 896087
length of the vocabulary: 3102


start: creating bow representation...
finised creating bow input!

start: creating bow representation...
finised creating bow input!

start: creating bow representation...
finised creating bow input!

start: creating bow representation...
finised creating bow input!

start: creating bow representation...
finised creating bow input!
```

Vocabular und IDs anzeigen als Beispiel

In [17]:

```
# show for samples: 100 word2id and id2 word
word2id_df_sample = pd.DataFrame()
word2id_df_sample['word'] = list(word2id.keys())[:20]
word2id_df_sample['id'] = list(word2id.values())[:20]
word2id_df_sample
```

Out[17]:

	word	id
0	totally	0
1	top	1
2	claim	2
3	helping	3
4	purdue	4
5	conduct	5
6	implies	6
7	solutions	7
8	affect	8
9	multi	9
10	authorities	10
11	france	11
12	joseph	12
13	chris	13
14	roads	14
15	newer	15
16	supports	16
17	mr	17
18	position	18
19	macintosh	19

Die Größe von Datensätzen kontrollieren

In [18]:

```
# Kontrollieren die Größen von verschiedenen Datensätzen
print(f'Size of the vocabulary after prprocessing ist: {len(textsloader.vocabulary)}')
print(f'Size of train set: {len(train_set["tokens"])}')
print(f'Size of val set: {len(val_set["tokens"])}')
print(f'Size of test set: {len(test_set["test"]["tokens"])}')
```

```
Size of the vocabulary after prprocessing ist: 3102
Size of train set: 11214
Size of val set: 100
Size of test set: 7532
```

Word-Embedding: Word2Vec mit Skipgramm/CBOW

Dokumenten wiederstellen für Word2Vec Embedding

1. Da wir Embeddings für jedes Wort des Vocabulares (das Vocab nur aus dem Trainset) trainieren möchten, brauchen die Train_set (Dokumenten in Wörtern)
2. Wir trainieren Wort-Embedding für jedes Wort mit Skipgram Methode (die Autoren benutzten Skipgram. Sie stellen nur über CBOW in dem Hintergrund vor, aber sie benutzen tatsächlich Skipgram)
3. Trainierensetting = Word2Vec (siehe [Word2Vec-Tomas Mikolov (https://arxiv.org/pdf/1310.4546.pdf)](https://arxiv.org/pdf/1310.4546.pdf))

In [19]:

```
# re-erstellen von Dokumenten nach der Vorverarbeitungen. Die Dokumenten sind in
Wörtern und werden für Word-Embedding Training benutzt
docs_tr, docs_t, docs_v = textsloader.get_docs_in_words_for_each_set()
train_docs_df = pd.DataFrame()
train_docs_df['text-after-preprocessing'] = [' '.join(doc) for doc in docs_tr[:1
00]]
train_docs_df
```

```
save docs in txt...
save docs finished
```

Out[19]:

	text-after-preprocessing
0	jackson defense nntp posting host university i...
1	apollo hp red police state usa nntp posting ho...
2	dartmouth brian hughes installing ram quadra r...
3	bu boston university physics department articl...
4	king eng umd doug computer design lab maryland...
...	...
95	physics ca pc windows os unix reply physics ca...
96	ncr jim parts information distribution world n...
97	sera zuma serdar argic nazi germany armenians ...
98	chips astro temple bible research temple unive...
99	loss cmu doug loss crazy electrical computer e...

100 rows × 1 columns

Word-Embedding trainieren mit dem Traindatensatz und gespeichert für ETM später

Wichtige Parameters sind:

1. model_name = "skipgram"/"cbow"/"bert"

In [20]:

```
save_path = Path.joinpath(Path.cwd(), f'prepared_data/min_df_{min_df}')
figures_path = Path.joinpath(Path.cwd(), f'figures/min_df_{min_df}')
Path(save_path).mkdir(parents=True, exist_ok=True)
Path(figures_path).mkdir(parents=True, exist_ok=True)
print(save_path)

vocab = list(word2id.keys())
model_name = "skipgram"
```

```
/content/replication-topic-modelling-in-embedding-space/prepared_da
ta/min_df_100
```

In [21]:

```python
from src.embedding import WordEmbeddingCreator
from pathlib import Path

if model_name != "bert" and use_bert_embedding == False:
  wb_creator = WordEmbeddingCreator(model_name=model_name, documents = docs_tr, save_path= save_path)
  wb_creator.train(min_count=0, embedding_size= 300)
  wb_creator.create_and_save_vocab_embedding(vocab, save_path)
else:
  print('festlegen welches Modell für word2vec soll genutzt werden!\n Wenn bert-Modell, bitte die Vocabular aktualisieren durch use_bert_embedding = True')
```

```
train word-embedding with skipgram
length of vocabulary from word-embedding with skipgram: 3102
length of vocabulary after creating BOW: 3102

100%|██████████| 3102/3102 [00:00<00:00, 5894.95it/s]
```

Visualierung von Wortembeddings mit UMAP (UMAP werden die Embeddings zu 2D reduziert. KMeans wurden benutzen, um zu sehen, wie die Clusters von Wörtern aussehen - nur zu sehen, nicht zu dem Paper gehören)

- Dieses Experiment gehört nicht zum Artikel, der repliziert wurde

In [ ]:

```
if model_name != "bert" and use_bert_embedding == False:
  wb_creator.cluster_words(save_path, figures_path, n_components=2, text = False
)
else:
  print('festlegen welches Modell für word2vec soll genutzt werden!\n Wenn bert-
Modell, bitte die Vocabular aktualisieren durch use_bert_embedding = True')
```

```
/usr/local/lib/python3.7/dist-packages/numba/np/ufunc/parallel.py:3
63: NumbaWarning: The TBB threading layer requires TBB version 201
9.5 or later i.e., TBB_INTERFACE_VERSION >= 11005. Found TBB_INTERF
ACE_VERSION = 9107. The TBB threading layer is disabled.
  warnings.warn(problem)
```

Testen ein paar Word-Embeddings and ähnliche semantische Wörter

In [22]:

```
if model_name != "bert" and use_bert_embedding == False:
  v = list(wb_creator.model.wv.vocab)[0]
  vec = list(wb_creator.model.wv.__getitem__(v))
  print(f'{model_name} word-embedding of the word: {v}')
  print(f'some elements of vector: {vec[:5]}')
  print(f'total dim of vector: {len(vec)}')
  print(f'show some semantic similar words \n')
  for i in range(5,10):
      print(f'neighbors of word: {vocab[i]}')
      print([r[0] for r in wb_creator.find_most_similar_words(n_neighbor=5, word
=vocab[i])])
      print([r[1] for r in wb_creator.find_most_similar_words(n_neighbor=5, word
=vocab[i])])
      print(100*"-")
else:
  print('festlegen welches Modell für word2vec soll genutzt werden!\n Wenn bert-
Modell, bitte die Vocabular aktualisieren durch use_bert_embedding = True')
```

```
skipgram word-embedding of the word: jackson
some elements of vector: [-0.0064425697, 0.05336388, -0.0791205, 0.
14298755, 0.01092479]
total dim of vector: 300
show some semantic similar words

neighbors of word: conduct
['establish', 'interests', 'grounds', 'measures', 'economic']
[0.8785027861595154, 0.8771042227745056, 0.8538452982902527, 0.8492
910265922546, 0.8467714786529541]
--------------------------------------------------------------------
--------------------------------
neighbors of word: implies
['deny', 'irrelevant', 'imply', 'justification', 'arguing']
[0.8767746090888977, 0.8689412474632263, 0.8676267266273499, 0.8623
065948486328, 0.861356258392334]
--------------------------------------------------------------------
--------------------------------
neighbors of word: solutions
['tool', 'tools', 'machines', 'platform', 'bug']
[0.580713152885437, 0.5781648755073547, 0.5764162540435791, 0.56970
91817855835, 0.5674580335617065]
--------------------------------------------------------------------
--------------------------------
neighbors of word: affect
['virtually', 'circumstances', 'rely', 'serves', 'causing']
[0.7895834445953369, 0.7865622639656067, 0.7850110530853271, 0.7843
527793884277, 0.7793745398521423]
--------------------------------------------------------------------
--------------------------------
neighbors of word: multi
['interface', 'displays', 'capabilities', 'platform', 'select']
[0.8433104753494263, 0.7820426225662231, 0.7805764675140381, 0.7790
523767471313, 0.7777268290519714]
--------------------------------------------------------------------
--------------------------------
```

Wenn Bert-Embedding benutzt wurde: BERT-Embedding

1. Bert-Embedding wurde in einem anderen Prozess durchgeführt
2. Um Bert-Durchführen zu können, bitte herunterladen diesen Daten: `bert_vocab_embedding.txt` ,und in den richtigen Order einpacken, konvertiern mittels `src/bert_covert_format.py` , wie im dem READme beschrieben wurde.
3. Bert-Embedding wurde in dem `prepared_data/bert_vocab_embedding.txt` gespeichert
4. Um Bert-Embedding verwendet, muss man im Vorfield folgende Punkten achten:

- Bei der Vorbereitung von Daten muss folgende Argument setzen: use_bert_embedding = True
- model_name = "bert" setzen

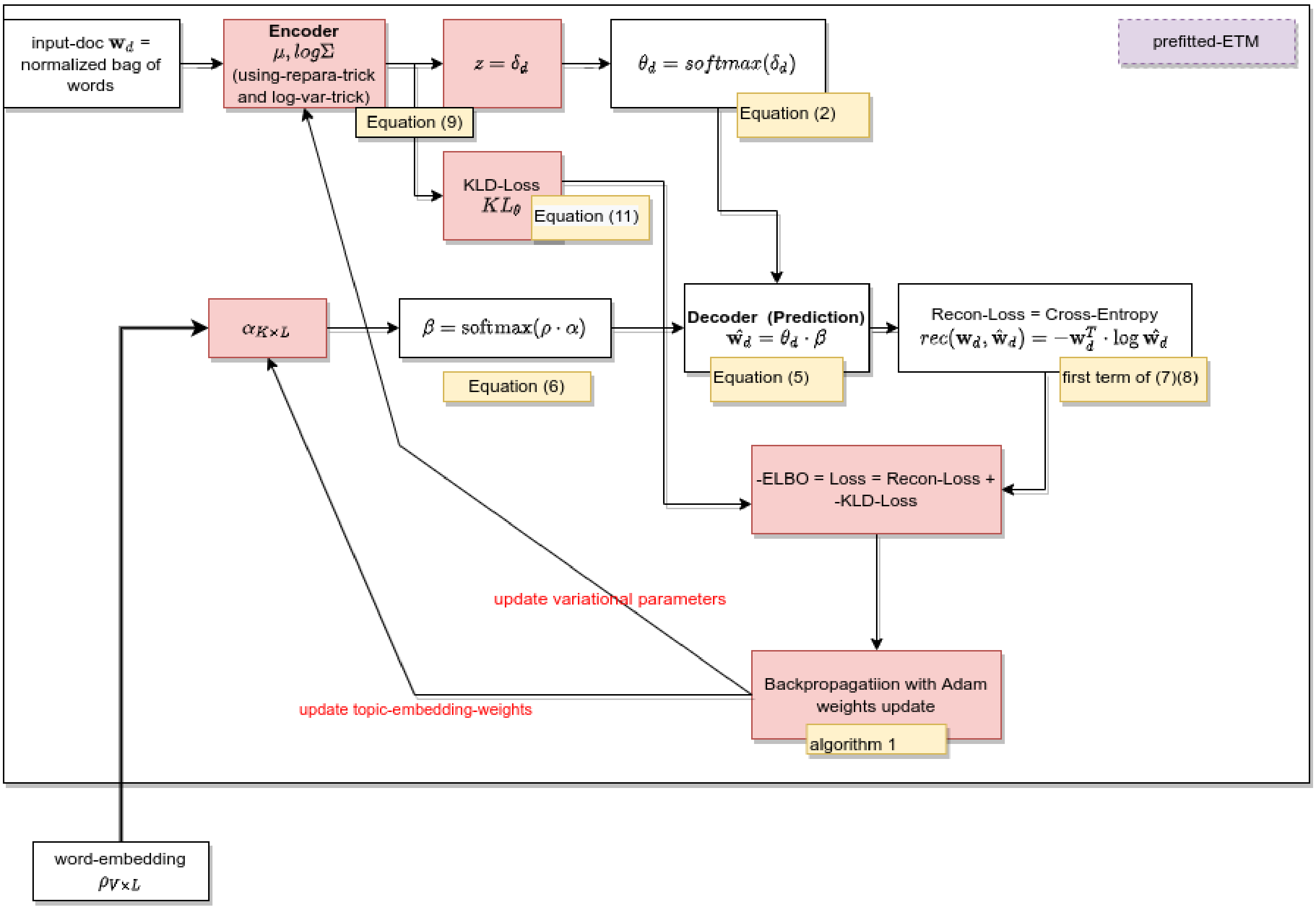
In [23]:

```
#run bert_main.py
#dieser Teil wegen der Laufzeit werden separat durchgeführt. Die Vocab-Embedding
s wurde gespeichert und kann geladen und benutzt.
#bert_vocab_embedding.txt in prepared_data
if model_name == "bert" and use_bert_embedding == True:
  from src.embedding import BertEmbedding
  bert_eb = BertEmbedding('prepared_data') #directory, where the txt.file of ber
t_vocab_embedding.txt ist
  try:
    bert_eb.get_bert_embeddings(vocab)
    print(bert_eb.bert_embeddings.shape)
  except:
    print("musst bert_main.py lokal durchgeführt werden, um die Bert-Embedding f
ür Vocabular zu erstellen. \n")
    print("die Embedding wird erst danach durch bert_main.py in dem Ordner prepa
red_data\bert_vocab_embedding.txt' gespeichert")
```

ETM Model und ETM Trainieren

ETM hat die ähnliche Architektur eines Variational Autoencoders (Encoder für Sampling latent Repräsentation von Dokument) und (Decoder: eigenlich nur das Produkt von doc-over-topics und topic-over-vocabulary)

ETM wird mit den pretrainierten Embedding kombiniert. Die Embeddings für Topics werden als Gewichten eines Teiles des Netzes aktualiert mittels der negative-ELBO (Reconstruction-Loss + KLD Loss)



kontrollieren die Inputdaten DocSet

1. Diese Klasse `DocSet` wurde implementiert, damit die Daten effizienter mit Pytorch geladen innerhalb des Batches werden können
2. `tr_set.__getitem__(0`) return die Repräsentation für das Dokument 0. Wegen der Normalisierung - die Summe = `0.9999997615814209`

In [24]:

```
# using DocSet to use easier the modul DataSet from torch
from src.train_etm import DocSet, TrainETM
from src.etm import ETM
import torch

vocab_size = len(list(word2id.keys()))
tr_set = DocSet("train", vocab_size, train_set, normalize_data=True)
print(f'total train docs: {len(tr_set)}')
print(f'sum of vector: {sum(tr_set.__getitem__(0)["normalized"])}')
print(f'length of vector: {torch.norm(tr_set.__getitem__(0)["normalized"])}')
```

```
total train docs: 11214
sum of vector: 1.0000003576278687
length of vector: 0.1649533212184906
```

Prefitted-Embeddings einlesen

In [25]:

```
# read embedding
from src.embedding import read_prefitted_embedding
_, embedding_data = read_prefitted_embedding(model_name, vocab, save_path)
```

```
start reading lines embeddings file:...

reading word-embedding...: 100%|██████████| 3102/3102 [00:00<00:00,
9954.79it/s]

end reading lines embeddings file!
```

Trainingsparametern vorbereiten

In [26]:

```
epochs = 150
batch_size = 1000
lr = 0.002
wdecay = 0.0000012
num_topics = 20
t_hidden_size = 8
theta_act = "tanh"
```

In [27]:

```
class TrainArguments:
      def __init__(self, epochs, batch_size, log_interval):
          self.epochs = epochs
          self.batch_size = batch_size
          self.log_interval = log_interval

class OptimizerArguments:
      def __init__(self, optimizer_name, lr, wdecay):
            self.optimizer = optimizer_name
            self.lr = lr
            self.wdecay = wdecay


train_args = TrainArguments(epochs=epochs, batch_size=batch_size, log_interval=None)
optimizer_args = OptimizerArguments(optimizer_name="adam", lr=lr, wdecay=wdecay)
print(train_args.epochs)
print(optimizer_args.optimizer)
rho_size = len(embedding_data[0])
emb_size = len(embedding_data[0])
```

```
150
adam
```

ETM mit Cross-Entropy

1. ETM-initialisieren
2. Trainieren das ETM-Modell mit den Training-Settings-Parameters

Aktivierte Funktion: Tanh

In [28]:

```
#-------------------------training--------------------------------------------
-------
#del etm_model
# define the ETM-model with setting-parameters
etm_model = ETM(
      num_topics,
      vocab_size,
      t_hidden_size, rho_size, emb_size, theta_act,
      embedding_data,
      enc_drop=0.5)

print(etm_model)

loss_name = "cross-entropy"

train_class = TrainETM().train(
    etm_model,
    loss_name,
    vocab_size,
    train_args, optimizer_args, train_set,
    normalize_data = True,
    figures_path = figures_path,
    visualization = True)

#-------------------show topics
topics = etm_model.show_topics(id2word, 10)
#for tp in topics:
#   print(tp)

topics = etm_model.show_topics(id2word, 10)
topics = [[e[0] for e in tp] for tp in topics] #get only top words
for tp in topics:
  print(tp)
```

```
ETM(
  (theta_act): Tanh()
  (topic_embeddings_alphas): Linear(in_features=300, out_features=2
0, bias=False)
  (q_theta): Sequential(
    (0): Linear(in_features=3102, out_features=8, bias=True)
    (1): Tanh()
    (2): Linear(in_features=8, out_features=8, bias=True)
    (3): Tanh()
  )
  (mu_q_theta): Linear(in_features=8, out_features=20, bias=True)
  (logsigma_q_theta): Linear(in_features=8, out_features=20, bias=T
rue)
)
number of batches: 12
Epoch: 1/150  -  Loss: 641.94141        Rec: 641.33563          K
L: 0.60583
Epoch: 2/150  -  Loss: 641.37244        Rec: 641.03174          K
L: 0.3407
Epoch: 3/150  -  Loss: 633.64502        Rec: 633.46173          K
L: 0.18331
Epoch: 4/150  -  Loss: 629.30756        Rec: 629.21301          K
L: 0.09456
Epoch: 5/150  -  Loss: 636.04968        Rec: 636.00208          K
L: 0.04756
Epoch: 6/150  -  Loss: 627.70471        Rec: 627.68103          K
L: 0.02362
Epoch: 7/150  -  Loss: 632.35791        Rec: 632.34375          K
L: 0.01418
Epoch: 8/150  -  Loss: 632.05511        Rec: 632.0448   KL: 0.0103
4
Epoch: 9/150  -  Loss: 630.68201        Rec: 630.66779          K
L: 0.01422
Epoch: 10/150  -  Loss: 633.72559       Rec: 633.70343          K
L: 0.02216
Epoch: 11/150  -  Loss: 631.77728       Rec: 631.73358          K
L: 0.0438
Epoch: 12/150  -  Loss: 631.47937       Rec: 631.38635          K
L: 0.09309
Epoch: 13/150  -  Loss: 630.909         Rec: 630.70239          K
L: 0.20657
Epoch: 14/150  -  Loss: 630.11023       Rec: 629.62891          K
L: 0.48133
Epoch: 15/150  -  Loss: 630.41028       Rec: 629.35614          K
L: 1.05419
Epoch: 16/150  -  Loss: 618.37512       Rec: 616.5495   KL: 1.8255
7
Epoch: 17/150  -  Loss: 620.46033       Rec: 618.2265   KL: 2.2337
1
Epoch: 18/150  -  Loss: 627.32092       Rec: 625.07288          K
L: 2.24803
Epoch: 19/150  -  Loss: 617.276         Rec: 614.93481          K
L: 2.34126
Epoch: 20/150  -  Loss: 619.20752       Rec: 616.68628          K
L: 2.52135
Epoch: 21/150  -  Loss: 622.58838       Rec: 619.96069          K
L: 2.62767
Epoch: 22/150  -  Loss: 616.5025        Rec: 613.76001          K
L: 2.74254
Epoch: 23/150  -  Loss: 613.87732       Rec: 611.02051          K
L: 2.85679
```

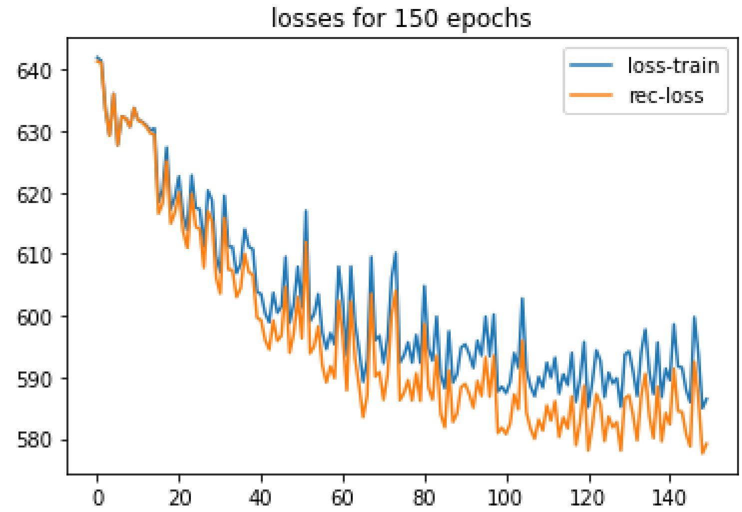
```
Epoch: 24/150  -  Loss: 622.7713        Rec: 619.81677          K
L: 2.95452
Epoch: 25/150  -  Loss: 617.50507       Rec: 614.41309          K
L: 3.09199
Epoch: 26/150  -  Loss: 617.24329       Rec: 614.02234          K
L: 3.22097
Epoch: 27/150  -  Loss: 611.14722       Rec: 607.84076          K
L: 3.30646
Epoch: 28/150  -  Loss: 620.29187       Rec: 616.91846          K
L: 3.37352
Epoch: 29/150  -  Loss: 618.659         Rec: 615.23132          K
L: 3.42766
Epoch: 30/150  -  Loss: 609.69763       Rec: 606.20892          K
L: 3.48861
Epoch: 31/150  -  Loss: 607.13043       Rec: 603.63428          K
L: 3.49611
Epoch: 32/150  -  Loss: 619.45953       Rec: 615.88037          K
L: 3.5792
Epoch: 33/150  -  Loss: 611.18762       Rec: 607.53412          K
L: 3.65352
Epoch: 34/150  -  Loss: 611.12872       Rec: 607.40393          K
L: 3.72477
Epoch: 35/150  -  Loss: 606.9585        Rec: 603.07495          K
L: 3.88348
Epoch: 36/150  -  Loss: 608.43793       Rec: 604.48724          K
L: 3.95077
Epoch: 37/150  -  Loss: 613.99915       Rec: 610.00208          K
L: 3.99706
Epoch: 38/150  -  Loss: 611.15588       Rec: 607.13788          K
L: 4.01811
Epoch: 39/150  -  Loss: 610.74951       Rec: 606.62781          K
L: 4.12165
Epoch: 40/150  -  Loss: 603.88501       Rec: 599.70532          K
L: 4.17971
Epoch: 41/150  -  Loss: 603.54315       Rec: 599.28442          K
L: 4.25868
Epoch: 42/150  -  Loss: 600.42847       Rec: 596.08752          K
L: 4.341
Epoch: 43/150  -  Loss: 598.93994       Rec: 594.53003          K
L: 4.40993
Epoch: 44/150  -  Loss: 603.73389       Rec: 599.24286          K
L: 4.49099
Epoch: 45/150  -  Loss: 600.50464       Rec: 595.92761          K
L: 4.57697
Epoch: 46/150  -  Loss: 601.52435       Rec: 596.81696          K
L: 4.70741
Epoch: 47/150  -  Loss: 609.49188       Rec: 604.72723          K
L: 4.76468
Epoch: 48/150  -  Loss: 598.85083       Rec: 594.00275          K
L: 4.84795
Epoch: 49/150  -  Loss: 601.98059       Rec: 597.08997          K
L: 4.8906
Epoch: 50/150  -  Loss: 607.97034       Rec: 603.0542   KL: 4.9160
9
Epoch: 51/150  -  Loss: 601.39307       Rec: 596.35767          K
L: 5.03544
Epoch: 52/150  -  Loss: 617.00031       Rec: 611.92847          K
L: 5.07185
Epoch: 53/150  -  Loss: 599.11151       Rec: 593.90887          K
L: 5.20271
Epoch: 54/150  -  Loss: 600.28235       Rec: 595.0282   KL: 5.2541
```

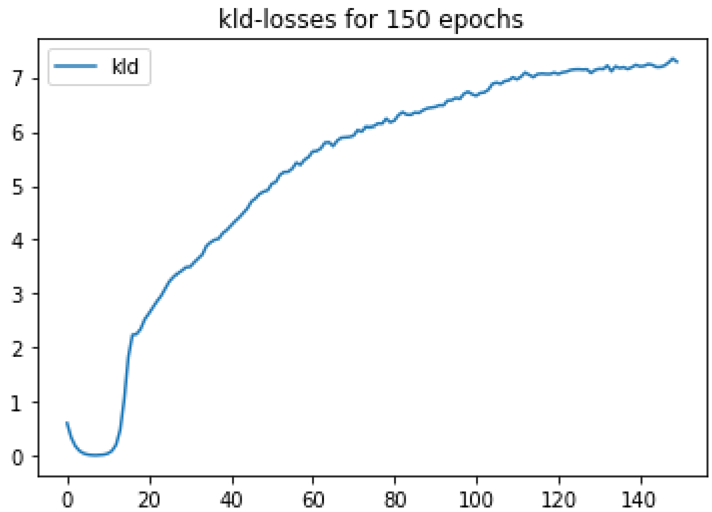
```
9
Epoch: 55/150  -  Loss: 603.53802       Rec: 598.28015          K
L: 5.25789
Epoch: 56/150  -  Loss: 597.25629       Rec: 591.93896          K
L: 5.31733
Epoch: 57/150  -  Loss: 594.59729       Rec: 589.1712   KL: 5.4260
4
Epoch: 58/150  -  Loss: 597.18481       Rec: 591.7981   KL: 5.3868
4
Epoch: 59/150  -  Loss: 595.36938       Rec: 589.88751          K
L: 5.48189
Epoch: 60/150  -  Loss: 608.01544       Rec: 602.48254          K
L: 5.53298
Epoch: 61/150  -  Loss: 602.31946       Rec: 596.68323          K
L: 5.63619
Epoch: 62/150  -  Loss: 593.51733       Rec: 587.87219          K
L: 5.64515
Epoch: 63/150  -  Loss: 608.03174       Rec: 602.33923          K
L: 5.69255
Epoch: 64/150  -  Loss: 598.89368       Rec: 593.09521          K
L: 5.79838
Epoch: 65/150  -  Loss: 594.49249       Rec: 588.68579          K
L: 5.80669
Epoch: 66/150  -  Loss: 589.26892       Rec: 583.53406          K
L: 5.73472
Epoch: 67/150  -  Loss: 592.87671       Rec: 587.04163          K
L: 5.83511
Epoch: 68/150  -  Loss: 609.49518       Rec: 603.60437          K
L: 5.89083
Epoch: 69/150  -  Loss: 596.02014       Rec: 590.12115          K
L: 5.899
Epoch: 70/150  -  Loss: 596.75189       Rec: 590.84705          K
L: 5.90498
Epoch: 71/150  -  Loss: 592.28888       Rec: 586.36035          K
L: 5.92849
Epoch: 72/150  -  Loss: 596.48511       Rec: 590.4541   KL: 6.0309
8
Epoch: 73/150  -  Loss: 606.28528       Rec: 600.28168          K
L: 6.0036
Epoch: 74/150  -  Loss: 610.17926       Rec: 604.0918   KL: 6.0873
8
Epoch: 75/150  -  Loss: 592.36169       Rec: 586.28308          K
L: 6.07857
Epoch: 76/150  -  Loss: 593.51343       Rec: 587.41479          K
L: 6.09866
Epoch: 77/150  -  Loss: 595.68408       Rec: 589.53754          K
L: 6.14665
Epoch: 78/150  -  Loss: 592.38898       Rec: 586.24731          K
L: 6.14164
Epoch: 79/150  -  Loss: 596.86597       Rec: 590.63049          K
L: 6.23545
Epoch: 80/150  -  Loss: 592.33905       Rec: 586.17194          K
L: 6.16706
Epoch: 81/150  -  Loss: 604.86389       Rec: 598.66418          K
L: 6.19966
Epoch: 82/150  -  Loss: 594.66132       Rec: 588.35956          K
L: 6.30172
Epoch: 83/150  -  Loss: 592.71863       Rec: 586.36353          K
L: 6.35513
Epoch: 84/150  -  Loss: 599.86902       Rec: 593.55872          K
L: 6.31026
```

```
Epoch: 85/150  -  Loss: 590.32379       Rec: 584.01862          K
L: 6.30524
Epoch: 86/150  -  Loss: 588.23157       Rec: 581.88269          K
L: 6.34877
Epoch: 87/150  -  Loss: 597.4859        Rec: 591.14594          K
L: 6.34001
Epoch: 88/150  -  Loss: 589.13507       Rec: 582.7439   KL: 6.3911
8
Epoch: 89/150  -  Loss: 590.5351        Rec: 584.1095   KL: 6.4256
8
Epoch: 90/150  -  Loss: 594.84167       Rec: 588.4032   KL: 6.4383
9
Epoch: 91/150  -  Loss: 595.36865       Rec: 588.91602          K
L: 6.45264
Epoch: 92/150  -  Loss: 593.75922       Rec: 587.276    KL: 6.4831
6
Epoch: 93/150  -  Loss: 591.55273       Rec: 585.07288          K
L: 6.47988
Epoch: 94/150  -  Loss: 595.96362       Rec: 589.39325          K
L: 6.57035
Epoch: 95/150  -  Loss: 593.55798       Rec: 586.98254          K
L: 6.57547
Epoch: 96/150  -  Loss: 599.87646       Rec: 593.25769          K
L: 6.6188
Epoch: 97/150  -  Loss: 593.45386       Rec: 586.85339          K
L: 6.60052
Epoch: 98/150  -  Loss: 600.21637       Rec: 593.52283          K
L: 6.69353
Epoch: 99/150  -  Loss: 587.68573       Rec: 580.95001          K
L: 6.73584
Epoch: 100/150  -  Loss: 588.47156      Rec: 581.78259          K
L: 6.68891
Epoch: 101/150  -  Loss: 587.42883      Rec: 580.77124          K
L: 6.65756
Epoch: 102/150  -  Loss: 589.19141      Rec: 582.48145          K
L: 6.7099
Epoch: 103/150  -  Loss: 593.93628      Rec: 587.21503          K
L: 6.72118
Epoch: 104/150  -  Loss: 591.57422      Rec: 584.8042   KL: 6.7700
2
Epoch: 105/150  -  Loss: 602.82935      Rec: 595.94904          K
L: 6.88032
Epoch: 106/150  -  Loss: 591.15979      Rec: 584.25543          K
L: 6.90436
Epoch: 107/150  -  Loss: 588.51746      Rec: 581.63806          K
L: 6.87939
Epoch: 108/150  -  Loss: 586.94556      Rec: 580.01465          K
L: 6.93093
Epoch: 109/150  -  Loss: 590.05664      Rec: 583.11102          K
L: 6.94572
Epoch: 110/150  -  Loss: 588.34485      Rec: 581.34387          K
L: 7.00102
Epoch: 111/150  -  Loss: 592.37219      Rec: 585.41052          K
L: 6.96175
Epoch: 112/150  -  Loss: 589.96204      Rec: 582.94946          K
L: 7.01252
Epoch: 113/150  -  Loss: 593.19177      Rec: 586.10229          K
L: 7.0894
Epoch: 114/150  -  Loss: 587.35052      Rec: 580.30505          K
L: 7.0455
Epoch: 115/150  -  Loss: 590.4707       Rec: 583.46625          K
```

```
L: 7.00442
Epoch: 116/150  -  Loss: 588.74866      Rec: 581.68945         K
L: 7.05924
Epoch: 117/150  -  Loss: 593.96942      Rec: 586.90015         K
L: 7.06925
Epoch: 118/150  -  Loss: 586.02454      Rec: 578.9624   KL: 7.0621
Epoch: 119/150  -  Loss: 589.70441      Rec: 582.64258         K
L: 7.06188
Epoch: 120/150  -  Loss: 595.73792      Rec: 588.64929         K
L: 7.08859
Epoch: 121/150  -  Loss: 585.15137      Rec: 578.09039         K
L: 7.06095
Epoch: 122/150  -  Loss: 589.20355      Rec: 582.11084         K
L: 7.0927
Epoch: 123/150  -  Loss: 594.39331      Rec: 587.28796         K
L: 7.10533
Epoch: 124/150  -  Loss: 592.68884      Rec: 585.5528   KL: 7.1360
1
Epoch: 125/150  -  Loss: 586.77545      Rec: 579.62769         K
L: 7.14776
Epoch: 126/150  -  Loss: 590.7536       Rec: 583.60315         K
L: 7.15045
Epoch: 127/150  -  Loss: 589.01215      Rec: 581.86871         K
L: 7.14351
Epoch: 128/150  -  Loss: 589.85974      Rec: 582.70972         K
L: 7.15004
Epoch: 129/150  -  Loss: 585.16913      Rec: 578.08765         K
L: 7.0815
Epoch: 130/150  -  Loss: 593.74414      Rec: 586.6062   KL: 7.1379
8
Epoch: 131/150  -  Loss: 594.22217      Rec: 587.06274         K
L: 7.15949
Epoch: 132/150  -  Loss: 590.96344      Rec: 583.81036         K
L: 7.15312
Epoch: 133/150  -  Loss: 586.97205      Rec: 579.75281         K
L: 7.21926
Epoch: 134/150  -  Loss: 594.14001      Rec: 587.02844         K
L: 7.11152
Epoch: 135/150  -  Loss: 597.77637      Rec: 590.57214         K
L: 7.20424
Epoch: 136/150  -  Loss: 590.8291       Rec: 583.65753         K
L: 7.17156
Epoch: 137/150  -  Loss: 587.29797      Rec: 580.10938         K
L: 7.18861
Epoch: 138/150  -  Loss: 595.66943      Rec: 588.51538         K
L: 7.15409
Epoch: 139/150  -  Loss: 586.77808      Rec: 579.58887         K
L: 7.18925
Epoch: 140/150  -  Loss: 591.42798      Rec: 584.1958   KL: 7.2321
1
Epoch: 141/150  -  Loss: 589.54346      Rec: 582.34045         K
L: 7.20293
Epoch: 142/150  -  Loss: 598.61938      Rec: 591.40741         K
L: 7.21197
Epoch: 143/150  -  Loss: 591.76001      Rec: 584.51379         K
L: 7.24626
Epoch: 144/150  -  Loss: 591.58795      Rec: 584.35718         K
L: 7.23078
Epoch: 145/150  -  Loss: 587.94019      Rec: 580.74823         K
L: 7.19196
Epoch: 146/150  -  Loss: 585.94458      Rec: 578.75098         K
```

```
L: 7.19359
Epoch: 147/150  -  Loss: 599.77527      Rec: 592.55676          K
L: 7.2185
Epoch: 148/150  -  Loss: 593.73077      Rec: 586.45483          K
L: 7.27584
Epoch: 149/150  -  Loss: 584.98438      Rec: 577.63501          K
L: 7.34939
Epoch: 150/150  -  Loss: 586.47314      Rec: 579.18323          K
L: 7.28994
Checkpoint saved at checkpoints/etm_epoch_150.pth.tar
```





```
['left', 'research', 'side', 'center', 'message', 'back', 'org', 'p
roblems', 'major', 'disclaimer']
['writes', 'host', 'ca', 'posting', 'university', 'article', 'nnt
p', 'mail', 'computer', 'cs']
['writes', 'article', 'posting', 'host', 'nntp', 'university', 'c
s', 'reply', 'apr', 'org']
['time', 'work', 'back', 'years', 'good', 'run', 'read', 'long', 'm
ake', 'times']
['windows', 'file', 'dos', 'drive', 'software', 'system', 'informat
ion', 'program', 'data', 'version']
['people', 'problem', 'things', 'find', 'point', 'made', 'lot', 'th
ing', 'big', 'good']
['space', 'key', 'nasa', 'encryption', 'clipper', 'technology', 'ch
ip', 'keys', 'public', 'security']
['phone', 'line', 'power', 'send', 'find', 'buy', 'work', 'good',
'info', 'stuff']
['article', 'writes', 'computer', 'posting', 'case', 'wrote', 'pos
t', 'steve', 'type', 'power']
['writes', 'article', 'state', 'world', 'john', 'man', 'day', 'lov
e', 'news', 'david']
['car', 'speed', 'engine', 'bike', 'hp', 'high', 'cover', 'low', 'w
ire', 'rider']
['time', 'year', 'long', 'state', 'good', 'order', 'university', 'd
istribution', 'single', 'place']
['max', 'ibm', 'bit', 'graphics', 'color', 'card', 'scsi', 'mac',
'lc', 'pc']
['game', 'team', 'games', 'play', 'players', 'win', 'league', 'play
er', 'playoffs', 'division']
['god', 'jesus', 'israel', 'christians', 'armenian', 'jews', 'men',
'jewish', 'christian', 'history']
['ca', 'university', 'host', 'posting', 'nntp', 'distribution', 'd
e', 'uk', 'net', 'cs']
['world', 'article', 'read', 'book', 'good', 'university', 'david',
'issue', 'reply', 'question']
['case', 'mit', 'set', 'call', 'questions', 'local', 'high', 'insti
tute', 'part', 'support']
['people', 'government', 'law', 'gun', 'life', 'rights', 'fact', 'q
uestion', 'reason', 'person']
['fbi', 'health', 'children', 'mr', 'day', 'information', 'medica
l', 'house', 'care', 'general']
```

In [29]:

```python
from src.evaluierung import topicCoherence2, topicDiversity
tc = topicCoherence2(topics,len(topics),docs_tr,len(docs_tr))
td = topicDiversity(topics)
print(f'etm-topic-coherrence: {tc}')
print(f'etm-topic-diversity: {td}')
```

```
etm-topic-coherrence: 0.1595585135160988
etm-topic-diversity: 0.775
```

In [30]:

```
from src.utils_perplexity import get_perplexity
test_batch_size = 1000
_, test_ppl = get_perplexity(etm_model, test_set, vocab_size, test_batch_size)
print(f'etm-normalized-perplexity: {test_ppl}')
```

```
calculate perplexitiy of test dataset: ...
test-1-loader: 7
test-2-loader: 7
batch 0 finished
batch 1 finished
batch 2 finished
batch 3 finished
batch 4 finished
batch 5 finished
batch 6 finished
etm-normalized-perplexity: 0.9576724693745969
```

Aktivierte Funktion ReLU

In [31]:

```
del etm_model
del train_class
del train_args
del optimizer_args
```

In [32]:

```
train_args = TrainArguments(epochs=150, batch_size=1000, log_interval=None)
optimizer_args = OptimizerArguments(optimizer_name="adam", lr=0.002, wdecay=0.00
00012)

from src.embedding import read_prefitted_embedding
_, embedding_data = read_prefitted_embedding(model_name, vocab, save_path)

num_topics = 20
t_hidden_size = 800
rho_size = len(embedding_data[0])
emb_size = len(embedding_data[0])
theta_act = "ReLU"


etm_model = ETM(
      num_topics,
      vocab_size,
      t_hidden_size, rho_size, emb_size, theta_act,
      embedding_data,
      enc_drop=0.5)

print(etm_model)

#--------------------------training------------------------------------------------
-------
loss_name = "cross-entropy"

train_class = TrainETM().train(
    etm_model,
    loss_name,
    vocab_size,
    train_args, optimizer_args, train_set,
    normalize_data = True,
    figures_path = figures_path,
    visualization = True)


#------------------show topics
topics = etm_model.show_topics(id2word, 10)
topics = [[e[0] for e in tp] for tp in topics] #get only top words
for tp in topics:
  print(tp)

tc = topicCoherence2(topics,len(topics),docs_tr,len(docs_tr))
td = topicDiversity(topics)
print(f'topic-coherrence: {tc}')
print(f'topic-diversity: {td}')
```

```
start reading lines embeddings file:...

reading word-embedding...: 100%|██████████| 3102/3102 [00:00<00:00,
8653.27it/s]
```

```
end reading lines embeddings file!
ETM(
  (theta_act): ReLU()
  (topic_embeddings_alphas): Linear(in_features=300, out_features=2
0, bias=False)
  (q_theta): Sequential(
    (0): Linear(in_features=3102, out_features=800, bias=True)
    (1): ReLU()
    (2): Linear(in_features=800, out_features=800, bias=True)
    (3): ReLU()
  )
  (mu_q_theta): Linear(in_features=800, out_features=20, bias=True)
  (logsigma_q_theta): Linear(in_features=800, out_features=20, bias
=True)
)
number of batches: 12
Epoch: 1/150  -  Loss: 649.70953        Rec: 649.70026          K
L: 0.00929
Epoch: 2/150  -  Loss: 643.68079        Rec: 643.61145          K
L: 0.06925
Epoch: 3/150  -  Loss: 643.56934        Rec: 643.14886          K
L: 0.42048
Epoch: 4/150  -  Loss: 630.76715        Rec: 629.49231          K
L: 1.27488
Epoch: 5/150  -  Loss: 623.78253        Rec: 621.58148          K
L: 2.2011
Epoch: 6/150  -  Loss: 628.23364        Rec: 625.47089          K
L: 2.7628
Epoch: 7/150  -  Loss: 620.57849        Rec: 617.32025          K
L: 3.25821
Epoch: 8/150  -  Loss: 617.84705        Rec: 614.32288          K
L: 3.5242
Epoch: 9/150  -  Loss: 614.91205        Rec: 610.87817          K
L: 4.03386
Epoch: 10/150  -  Loss: 612.82678       Rec: 608.42169          K
L: 4.40505
Epoch: 11/150  -  Loss: 617.08679       Rec: 612.60571          K
L: 4.48112
Epoch: 12/150  -  Loss: 619.34595       Rec: 614.72351          K
L: 4.62245
Epoch: 13/150  -  Loss: 607.37134       Rec: 602.61591          K
L: 4.75541
Epoch: 14/150  -  Loss: 603.24695       Rec: 598.37384          K
L: 4.87311
Epoch: 15/150  -  Loss: 609.01404       Rec: 603.92578          K
L: 5.08835
Epoch: 16/150  -  Loss: 608.3667        Rec: 603.05609          K
L: 5.3106
Epoch: 17/150  -  Loss: 603.90204       Rec: 598.46741          K
L: 5.43465
Epoch: 18/150  -  Loss: 601.24731       Rec: 595.82745          K
L: 5.41986
Epoch: 19/150  -  Loss: 598.20386       Rec: 592.6178   KL: 5.5860
4
Epoch: 20/150  -  Loss: 599.75989       Rec: 594.17981          K
L: 5.58006
Epoch: 21/150  -  Loss: 599.51416       Rec: 593.76776          K
L: 5.74641
Epoch: 22/150  -  Loss: 601.95874       Rec: 596.20911          K
L: 5.74966
Epoch: 23/150  -  Loss: 607.34955       Rec: 601.51532          K
```

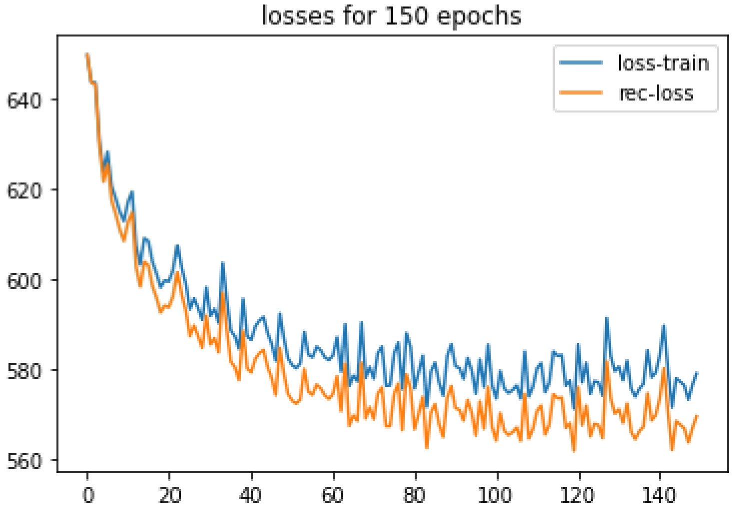
```
L: 5.83426
Epoch: 24/150  -  Loss: 602.46393       Rec: 596.57776          K
L: 5.88624
Epoch: 25/150  -  Loss: 598.89398       Rec: 593.00256          K
L: 5.89141
Epoch: 26/150  -  Loss: 593.34216       Rec: 587.38818          K
L: 5.95393
Epoch: 27/150  -  Loss: 595.76923       Rec: 589.7478   KL: 6.0214
5
Epoch: 28/150  -  Loss: 593.58099       Rec: 587.37988          K
L: 6.2011
Epoch: 29/150  -  Loss: 590.97974       Rec: 584.78387          K
L: 6.19588
Epoch: 30/150  -  Loss: 598.23376       Rec: 591.95673          K
L: 6.27708
Epoch: 31/150  -  Loss: 591.86389       Rec: 585.51416          K
L: 6.3497
Epoch: 32/150  -  Loss: 593.42725       Rec: 586.9198   KL: 6.5074
7
Epoch: 33/150  -  Loss: 590.31665       Rec: 583.79285          K
L: 6.52384
Epoch: 34/150  -  Loss: 603.64355       Rec: 597.00549          K
L: 6.63807
Epoch: 35/150  -  Loss: 595.60803       Rec: 588.93793          K
L: 6.66999
Epoch: 36/150  -  Loss: 588.60504       Rec: 581.74811          K
L: 6.85691
Epoch: 37/150  -  Loss: 587.33685       Rec: 580.47852          K
L: 6.85833
Epoch: 38/150  -  Loss: 584.55872       Rec: 577.617    KL: 6.9417
5
Epoch: 39/150  -  Loss: 595.64557       Rec: 588.56982          K
L: 7.07579
Epoch: 40/150  -  Loss: 587.37   Rec: 580.23553         KL: 7.1345
3
Epoch: 41/150  -  Loss: 586.52966       Rec: 579.3208   KL: 7.2087
6
Epoch: 42/150  -  Loss: 589.55798       Rec: 582.32275          K
L: 7.23526
Epoch: 43/150  -  Loss: 590.84833       Rec: 583.58508          K
L: 7.26326
Epoch: 44/150  -  Loss: 591.65442       Rec: 584.32367          K
L: 7.3307
Epoch: 45/150  -  Loss: 588.08789       Rec: 580.6156   KL: 7.4723
2
Epoch: 46/150  -  Loss: 585.65576       Rec: 578.07874          K
L: 7.57712
Epoch: 47/150  -  Loss: 581.89935       Rec: 574.30603          K
L: 7.5934
Epoch: 48/150  -  Loss: 592.39032       Rec: 584.74933          K
L: 7.641
Epoch: 49/150  -  Loss: 586.8938        Rec: 579.11584          K
L: 7.77801
Epoch: 50/150  -  Loss: 582.4361        Rec: 574.57471          K
L: 7.86134
Epoch: 51/150  -  Loss: 580.95898       Rec: 573.0708   KL: 7.8881
6
Epoch: 52/150  -  Loss: 580.27081       Rec: 572.40063          K
L: 7.87013
Epoch: 53/150  -  Loss: 581.30249       Rec: 573.26813          K
L: 8.03443
```

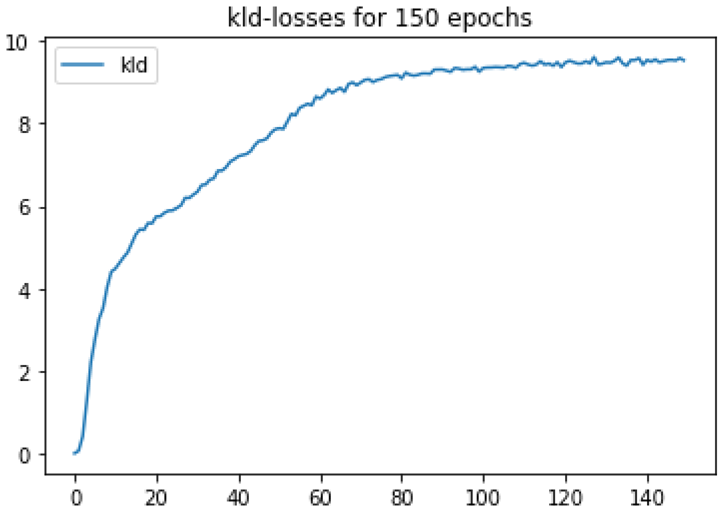
```
Epoch: 54/150  -  Loss: 588.30853       Rec: 580.08496          K
L: 8.22352
Epoch: 55/150  -  Loss: 583.23206       Rec: 575.04437          K
L: 8.18769
Epoch: 56/150  -  Loss: 582.6853        Rec: 574.31555          K
L: 8.36986
Epoch: 57/150  -  Loss: 585.03479       Rec: 576.61304          K
L: 8.42188
Epoch: 58/150  -  Loss: 584.24628       Rec: 575.77649          K
L: 8.46974
Epoch: 59/150  -  Loss: 582.69336       Rec: 574.26111          K
L: 8.43218
Epoch: 60/150  -  Loss: 582.06116       Rec: 573.41846          K
L: 8.64262
Epoch: 61/150  -  Loss: 583.0791        Rec: 574.48181          K
L: 8.59739
Epoch: 62/150  -  Loss: 587.14484       Rec: 578.46729          K
L: 8.67754
Epoch: 63/150  -  Loss: 579.534         Rec: 570.71442          K
L: 8.81965
Epoch: 64/150  -  Loss: 590.03656       Rec: 581.30298          K
L: 8.73353
Epoch: 65/150  -  Loss: 576.27319       Rec: 567.47144          K
L: 8.8017
Epoch: 66/150  -  Loss: 578.62646       Rec: 569.77234          K
L: 8.85409
Epoch: 67/150  -  Loss: 577.3291        Rec: 568.57031          K
L: 8.75873
Epoch: 68/150  -  Loss: 590.41821       Rec: 581.46851          K
L: 8.94968
Epoch: 69/150  -  Loss: 578.06909       Rec: 569.08044          K
L: 8.98871
Epoch: 70/150  -  Loss: 580.53448       Rec: 571.61475          K
L: 8.91973
Epoch: 71/150  -  Loss: 577.9209        Rec: 568.93292          K
L: 8.98801
Epoch: 72/150  -  Loss: 583.63129       Rec: 574.58191          K
L: 9.04943
Epoch: 73/150  -  Loss: 585.04871       Rec: 575.98242          K
L: 9.06636
Epoch: 74/150  -  Loss: 576.37909       Rec: 567.38306          K
L: 8.99601
Epoch: 75/150  -  Loss: 576.43524       Rec: 567.38647          K
L: 9.04868
Epoch: 76/150  -  Loss: 583.60596       Rec: 574.53845          K
L: 9.06751
Epoch: 77/150  -  Loss: 585.96283       Rec: 576.84473          K
L: 9.118
Epoch: 78/150  -  Loss: 575.633         Rec: 566.49164          K
L: 9.14132
Epoch: 79/150  -  Loss: 588.05463       Rec: 578.89636          K
L: 9.15828
Epoch: 80/150  -  Loss: 585.00647       Rec: 575.83575          K
L: 9.17069
Epoch: 81/150  -  Loss: 575.76355       Rec: 566.67804          K
L: 9.08546
Epoch: 82/150  -  Loss: 579.38452       Rec: 570.1593   KL: 9.2251
7
Epoch: 83/150  -  Loss: 583.06531       Rec: 573.89099          K
L: 9.1743
Epoch: 84/150  -  Loss: 571.67468       Rec: 562.52246          K
```

```
L: 9.15228
Epoch: 85/150  -  Loss: 579.49585       Rec: 570.32385          K
L: 9.17209
Epoch: 86/150  -  Loss: 581.49506       Rec: 572.28967          K
L: 9.20542
Epoch: 87/150  -  Loss: 577.08057       Rec: 567.87445          K
L: 9.20611
Epoch: 88/150  -  Loss: 574.16687       Rec: 564.96948          K
L: 9.19738
Epoch: 89/150  -  Loss: 582.93433       Rec: 573.63464          K
L: 9.29978
Epoch: 90/150  -  Loss: 585.60815       Rec: 576.30811          K
L: 9.30002
Epoch: 91/150  -  Loss: 580.74445       Rec: 571.44171          K
L: 9.3027
Epoch: 92/150  -  Loss: 580.28003       Rec: 571.00647          K
L: 9.27355
Epoch: 93/150  -  Loss: 577.86646       Rec: 568.62256          K
L: 9.24386
Epoch: 94/150  -  Loss: 582.53162       Rec: 573.19452          K
L: 9.33712
Epoch: 95/150  -  Loss: 579.73035       Rec: 570.40692          K
L: 9.32338
Epoch: 96/150  -  Loss: 574.60962       Rec: 565.31592          K
L: 9.29369
Epoch: 97/150  -  Loss: 582.21606       Rec: 572.90967          K
L: 9.30641
Epoch: 98/150  -  Loss: 576.09668       Rec: 566.79089          K
L: 9.3058
Epoch: 99/150  -  Loss: 585.47681       Rec: 576.10956          K
L: 9.36729
Epoch: 100/150  -  Loss: 576.43304      Rec: 567.18518          K
L: 9.24793
Epoch: 101/150  -  Loss: 573.51184      Rec: 564.17175          K
L: 9.34012
Epoch: 102/150  -  Loss: 579.64758      Rec: 570.29742          K
L: 9.35014
Epoch: 103/150  -  Loss: 575.73547      Rec: 566.38403          K
L: 9.35141
Epoch: 104/150  -  Loss: 574.75275      Rec: 565.39142          K
L: 9.36126
Epoch: 105/150  -  Loss: 575.40155      Rec: 566.04517          K
L: 9.35642
Epoch: 106/150  -  Loss: 576.44714      Rec: 567.10101          K
L: 9.34612
Epoch: 107/150  -  Loss: 573.50085      Rec: 564.11298          K
L: 9.38786
Epoch: 108/150  -  Loss: 583.97192      Rec: 574.59338          K
L: 9.37856
Epoch: 109/150  -  Loss: 573.97064      Rec: 564.62982          K
L: 9.34082
Epoch: 110/150  -  Loss: 576.10956      Rec: 566.6759   KL: 9.4337
2
Epoch: 111/150  -  Loss: 580.20343      Rec: 570.74005          K
L: 9.46342
Epoch: 112/150  -  Loss: 581.39673      Rec: 571.97424          K
L: 9.42253
Epoch: 113/150  -  Loss: 574.93127      Rec: 565.53711          K
L: 9.39416
Epoch: 114/150  -  Loss: 577.03156      Rec: 567.60999          K
L: 9.42163
```

```
Epoch: 115/150  -  Loss: 583.95667      Rec: 574.45319          K
L: 9.50351
Epoch: 116/150  -  Loss: 582.99036      Rec: 573.56909          K
L: 9.42127
Epoch: 117/150  -  Loss: 583.18701      Rec: 573.74048          K
L: 9.4466
Epoch: 118/150  -  Loss: 576.37805      Rec: 566.98785          K
L: 9.39022
Epoch: 119/150  -  Loss: 577.52185      Rec: 568.04346          K
L: 9.47841
Epoch: 120/150  -  Loss: 571.25177      Rec: 561.89075          K
L: 9.36099
Epoch: 121/150  -  Loss: 585.54175      Rec: 576.06787          K
L: 9.47383
Epoch: 122/150  -  Loss: 577.10803      Rec: 567.59442          K
L: 9.51355
Epoch: 123/150  -  Loss: 581.44855      Rec: 571.96985          K
L: 9.47867
Epoch: 124/150  -  Loss: 574.55334      Rec: 565.10724          K
L: 9.44614
Epoch: 125/150  -  Loss: 577.34613      Rec: 567.89789          K
L: 9.44827
Epoch: 126/150  -  Loss: 577.11316      Rec: 567.61133          K
L: 9.5018
Epoch: 127/150  -  Loss: 574.18317      Rec: 564.7334   KL: 9.4497
3
Epoch: 128/150  -  Loss: 591.43036      Rec: 581.82446          K
L: 9.60588
Epoch: 129/150  -  Loss: 582.91589      Rec: 573.49243          K
L: 9.42343
Epoch: 130/150  -  Loss: 579.60535      Rec: 570.16211          K
L: 9.44319
Epoch: 131/150  -  Loss: 580.62231      Rec: 571.146    KL: 9.4763
7
Epoch: 132/150  -  Loss: 577.56531      Rec: 568.09766          K
L: 9.46766
Epoch: 133/150  -  Loss: 581.91437      Rec: 572.40466          K
L: 9.50976
Epoch: 134/150  -  Loss: 575.71509      Rec: 566.12183          K
L: 9.59315
Epoch: 135/150  -  Loss: 573.93652      Rec: 564.48553          K
L: 9.45094
Epoch: 136/150  -  Loss: 575.66864      Rec: 566.27539          K
L: 9.39326
Epoch: 137/150  -  Loss: 576.77478      Rec: 567.24451          K
L: 9.53028
Epoch: 138/150  -  Loss: 584.24188      Rec: 574.71271          K
L: 9.52913
Epoch: 139/150  -  Loss: 578.24115      Rec: 568.66296          K
L: 9.57823
Epoch: 140/150  -  Loss: 579.26373      Rec: 569.8468   KL: 9.417
Epoch: 141/150  -  Loss: 583.10834      Rec: 573.57495          K
L: 9.53341
Epoch: 142/150  -  Loss: 589.71106      Rec: 580.22052          K
L: 9.49049
Epoch: 143/150  -  Loss: 580.0802       Rec: 570.53149          K
L: 9.54881
Epoch: 144/150  -  Loss: 571.57166      Rec: 562.09497          K
L: 9.47676
Epoch: 145/150  -  Loss: 578.0188       Rec: 568.51672          K
L: 9.50211
```

```
Epoch: 146/150  -  Loss: 577.26465       Rec: 567.73431          K
L: 9.53037
Epoch: 147/150  -  Loss: 576.4339        Rec: 566.89777          K
L: 9.53615
Epoch: 148/150  -  Loss: 573.29053       Rec: 563.76947          K
L: 9.52104
Epoch: 149/150  -  Loss: 576.54486       Rec: 566.9668   KL: 9.5780
9
Epoch: 150/150  -  Loss: 579.0882        Rec: 569.56366          K
L: 9.52451
Checkpoint saved at checkpoints/etm_epoch_150.pth.tar
```





```
['max', 'uk', 'ac', 'de', 'au', 'tu', 'sgi', 'rz', 'wa', 'keith']
['mail', 'information', 'list', 'group', 'email', 'internet', 'inf
o', 'send', 'ftp', 'faq']
['key', 'public', 'chip', 'clipper', 'encryption', 'government', 'l
aw', 'keys', 'phone', 'secure']
['posting', 'host', 'nntp', 'university', 'cc', 'news', 'reply', 'a
pr', 'article', 'message']
['years', 'read', 'back', 'times', 'heard', 'left', 'israel', 'ag
o', 'history', 'told']
['cs', 'university', 'writes', 'article', 'andrew', 'uiuc', 'cmu',
'cwru', 'engineering', 'ohio']
['state', 'writes', 'article', 'distribution', 'world', 'computer',
'university', 'usa', 'netcom', 'david']
['space', 'nasa', 'gov', 'access', 'health', 'national', 'shuttle',
'billion', 'research', 'medical']
['time', 'problem', 'work', 'find', 'long', 'system', 'line', 'goo
d', 'day', 'number']
['car', 'power', 'high', 'speed', 'front', 'bike', 'engine', 'wate
r', 'good', 'cars']
['god', 'jesus', 'people', 'life', 'christian', 'christians', 'chri
stianity', 'religion', 'church', 'christ']
['john', 'writes', 'article', 'bill', 'washington', 'man', 'art',
'ed', 'book', 'good']
['drive', 'card', 'mac', 'ibm', 'scsi', 'pc', 'disk', 'price', 'app
le', 'bus']
['article', 'writes', 'org', 'michael', 'hp', 'research', 'att', 'o
pinions', 'colorado', 'disclaimer']
['windows', 'file', 'dos', 'program', 'files', 'window', 'data', 'i
mage', 'software', 'system']
['game', 'team', 'games', 'players', 'league', 'year', 'play', 'wi
n', 'player', 'division']
['ca', 'sun', 'writes', 'wrote', 'version', 'posting', 'canada', 'm
ike', 'newsreader', 'post']
['question', 'point', 'true', 'fact', 'wrong', 'claim', 'case', 'pa
rt', 'evidence', 'simply']
['gun', 'government', 'people', 'israeli', 'armenians', 'armenian',
'world', 'villages', 'children', 'states']
['people', 'make', 'good', 'things', 'lot', 'made', 'problems', 'ba
d', 'thing', 'president']
topic-coherrence: 0.18821489049264942
topic-diversity: 0.9
```