```
## adapted from https://colab.research.google.com/drive/12zBvL0v0qZHbpSGXLLjHfQ0w6OZOVHaB#scrollTo=KhLt6VA3wv
# https://github.com/MilaNLProc/contextualized-topic-models#preprocessing


%%capture
!pip install contextualized-topic-models==1.8.1
!pip install torch==1.6.0+cu101 torchvision==0.7.0+cu101 -f https://download.pytorch.org/whl/torch_stable.htm


from contextualized_topic_models.models.ctm import CombinedTM
from contextualized_topic_models.utils.data_preparation import bert_embeddings_from_file, TopicModelDataPrepa
from contextualized_topic_models.utils.preprocessing import WhiteSpacePreprocessing
from contextualized_topic_models.datasets.dataset import CTMDataset
from contextualized_topic_models.evaluation.measures import CoherenceNPMI, InvertedRBO
from gensim.corpora.dictionary import Dictionary
from gensim.models import ldamodel
import os
import numpy as np
import pickle
import pandas as pd
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
True
```

Let's read our data files and store the documents as lists of strings

```
# MODEL DATA CATALOG ABSTRACT DATA:
df = pd.read_csv('abstracts.csv')
df["title_text"] = df["Title"] + ' ' + df["Text"]
wp = WhiteSpacePreprocessing(df.title_text)
text_training_preprocessed, text_training_not_preprocessed, vocab_list = wp.preprocess()
# TODO: is this the right way to combine the title see https://colab.research.google.com/drive/12hfBveGHRsxhP
```

NOTE: Make sure that the lenghts of the two lists of documents are the same and the index of a not preprocessed document corresponds to the index of the same preprocessed document.

```
print(len(text_training_preprocessed) == len(text_training_not_preprocessed))
```

```
True
```

Split into train/test

```
training_bow_documents = text_training_preprocessed[0:38]
training_contextual_document = text_training_not_preprocessed[0:38]

testing_bow_documents = text_training_preprocessed[38:]
testing_contextual_documents = text_training_not_preprocessed[38:]
```

Create the training set

```
#uncomment to use regular BERT
# tp = TopicModelDataPreparation("bert-base-nli-mean-tokens")

# Using SPECTER from ALLENAI
tp = TopicModelDataPreparation("allenai-specter")

# if using SPECTER, add titles to training_contextual_document:


# TODO implement sciBERT:
# notes...
# from transformers import AutoTokenizer, AutoModel
# tokenizer = AutoTokenizer.from_pretrained("gsarti/scibert-nli")
# model = AutoModel.from_pretrained("gsarti/scibert-nli")

# see https://www.kaggle.com/karlie777/covid-19-papers for an example


training_dataset = tp.create_training_set(training_contextual_document, training_bow_documents)
```

Let's check the vocabulary

```
tp.vocab[:10]
```

```
['ability',
 'absence',
 'acceleration',
 'accepted',
 'acceptor',
 'acceptors',
 'access',
 'accommodates',
 'account',
 'accounted']
```

```
#TODO check BERT input size
ctm = CombinedTM(input_size=len(tp.vocab), bert_input_size=768, num_epochs=200, n_components=9)
ctm.fit(training_dataset)
```

```
Epoch: [200/200]         Seen Samples: [7600/7600]      Train Loss: 854.8287417763158   Time: 0:00:00.238847: : 200it [00:50,  3.95it/s]
```

Get topics list

```
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
```

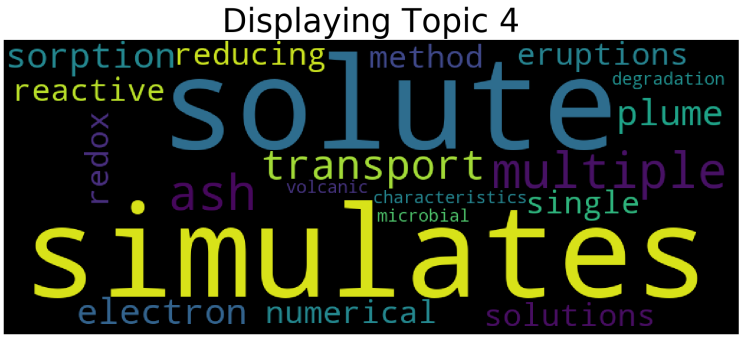
```
ctm.get_topics(5)
```

```
defaultdict(list,
            {0: ['count', 'circle', 'map', 'stereonet', 'point'],
             1: ['sediment', 'sea', 'sensitive', 'identify', 'coastal'],
             2: ['predicted', 'wet', 'runoff', 'may', 'distribution'],
             3: ['region', 'climate', 'conditions', 'projected', 'percent'],
             4: ['solute', 'simulates', 'multiple', 'ash', 'transport'],
             5: ['motion', 'function', 'magnitudes', 'update', 'trend'],
             6: ['groundwater',
              'package',
              'approach',
              'solution',
              'nonlinearities'],
             7: ['user', 'pl', 'compiled', 'form', 'made'],
             8: ['pumping',
              'interface',
              'modpath',
              'relation',
              'analyzehole']})
```

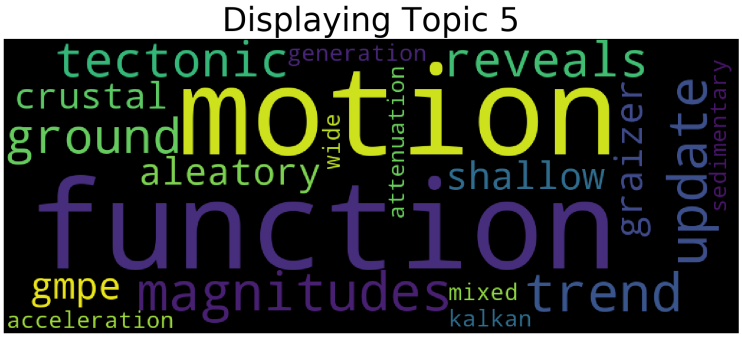
Look at a word cloud

```
%%capture
pip install matplotlib==3.1.3
```

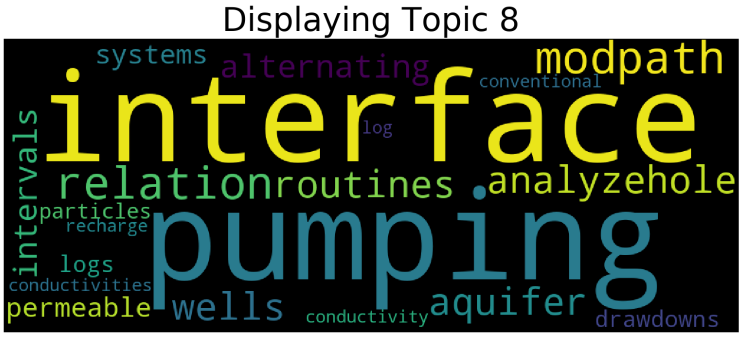
```
ctm.get_wordcloud(topic_id=4, n_words=20)
```




```
ctm.get_wordcloud(topic_id=5, n_words=20)
```




```
ctm.get_wordcloud(topic_id=8, n_words=20)
```



Use the test set

```
testing_dataset = tp.create_test_set(testing_contextual_documents, testing_bow_documents) # create dataset fc
# print(testing_dataset[0])
```

```
Batches: 100%                                                    1/1 [00:03<00:00, 3.78s/it]
```

```
predictions = ctm.get_doc_topic_distribution(testing_dataset, n_samples=1)
```

```
Sampling: [1/1]: : 1it [00:00,  5.93it/s]
```

```
#Select the test_document to view:
selected_doc = 0

print('probability for each topic', predictions[selected_doc])
print()
topic_index = np.argmax(predictions[selected_doc])
print('arg_max:', topic_index)
print()
print('original text:', testing_contextual_documents[selected_doc][0:350])
print()
print('topic_keywords:', ctm.get_topics(5)[topic_index])
# print(ctm.get_word_distribution_by_topic_id(topic_index)[0:20])
```

```
probability for each topic [0.03165091 0.01279292 0.06111116 0.02382202 0.03169251 0.01556775
 0.00698364 0.78069937 0.03567968]

arg_max: 7

original text: Scoops3D: software to analyze 3D slope stability throughout a digital landscape The computer program, Scoops3D, evaluates sl

topic_keywords: ['user', 'pl', 'compiled', 'form', 'made']
```

```
pt = ctm.get_predicted_topics(testing_dataset, n_samples=1)
print(pt)
```

```
Sampling: [1/1]: : lit [00:00,  5.44it/s][3, 4]
```

Evaluate the model

LDA

We are going to use gensim's LDA implementation and the preprocessed text to learn topics from the abstracts for comparison

```
lda_text = list(map(lambda x : x.split(), training_bow_documents))

id2word = Dictionary(lda_text)
texts = lda_text
corpus = [id2word.doc2bow(text) for text in texts]

lda_model = ldamodel.LdaModel(
   corpus=corpus, id2word=id2word, num_topics=9, passes=10, alpha='auto', per_word_topics=True, iterations=20


lda_topics = []
for i in range(9):
    t = [w[0] for w in lda_model.show_topic(i)[0:10]]
    lda_topics.append(t)


lda_topics[0]
```

```
['model',
 'permanence',
 'streamflow',
 'subduction',
 'channel',
 'predictions',
 'stream',
 'zones',
 'vflux',
 'vertical']
```

coherence score

```
#LDA topics
npmi = CoherenceNPMI(texts=texts, topics=lda_topics)
npmi.score()
```

```
-0.09232825860530004
```

```
#Combine topic score
# TODO should this be lda text?? seems like it needs to be the combined BERT + BoW

npmi = CoherenceNPMI(texts=texts, topics=ctm.get_topic_lists(10))
npmi.score()
# Exception: Words in topics are less than topk
```

```
-0.22333996743381274
```

pyLDAvis

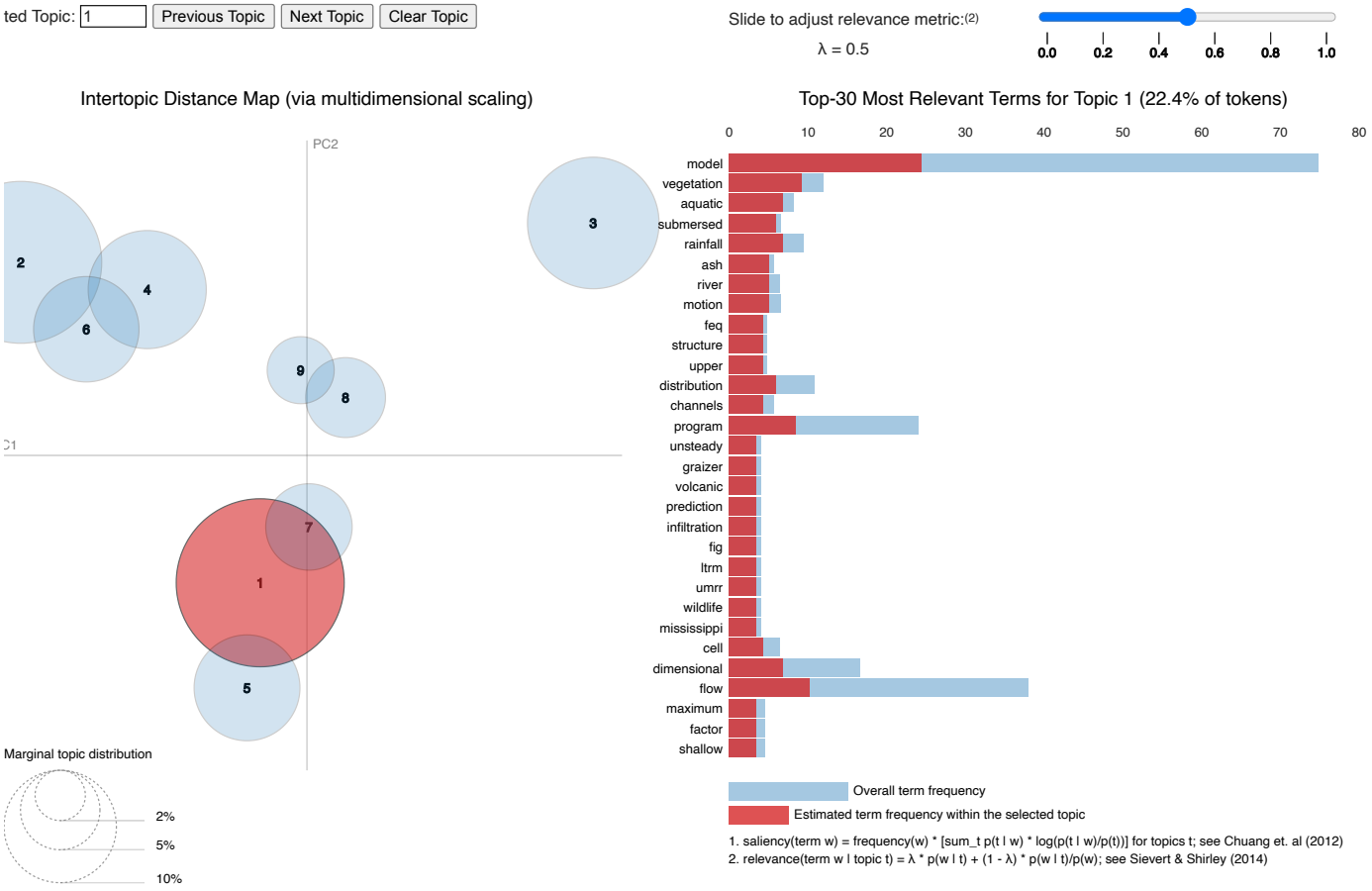
```
# !pip install pyLDAvis==3.3.1


import pyLDAvis.gensim_models as gensimvis
import pyLDAvis
```

```
/usr/local/lib/python3.7/dist-packages/past/types/oldstr.py:5: DeprecationWarning: Using or importing the ABCs from 'collections' instead c
  from collections import Iterable
/usr/local/lib/python3.7/dist-packages/past/builtins/misc.py:4: DeprecationWarning: Using or importing the ABCs from 'collections' instead
  from collections import Mapping
/usr/local/lib/python3.7/dist-packages/sklearn/decomposition/_lda.py:29: DeprecationWarning: `np.float` is a deprecated alias for the built
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  EPS = np.finfo(np.float).eps
```

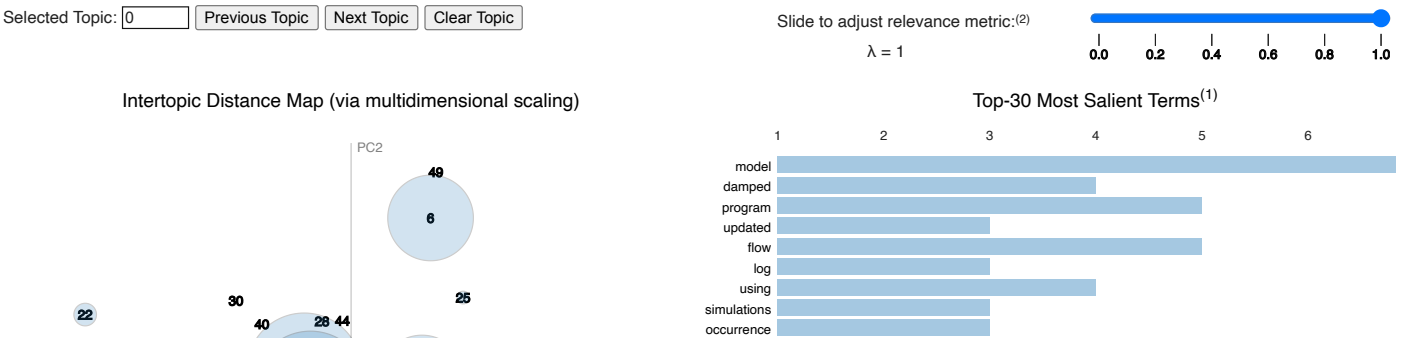
```
# https://github.com/bmabey/pyLDAvis/blob/master/notebooks/Gensim%20Newsgroup.ipynb
vis_data = gensimvis.prepare(lda_model, corpus, id2word)
```

```
pyLDAvis.display(vis_data)
```




```
from gensim import models

hdp = models.hdpmodel.HdpModel(corpus, id2word, T=50)
vis_data = gensimvis.prepare(hdp, corpus, id2word)
pyLDAvis.display(vis_data)
```




```
# hdp = models.hdpmodel.HdpModel(corpus, id2word, T=50)
vis_data = gensimvis.prepare(ctm, corpus, id2word)
pyLDAvis.display(vis_data)
# AttributeError: 'CombinedTM' object has no attribute 'num_topics'
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call
last)
<ipython-input-31-53351963741f> in <module>()
      1 # hdp = models.hdpmodel.HdpModel(corpus, id2word, T=50)
----> 2 vis_data = gensimvis.prepare(ctm, corpus, id2word)
      3 pyLDAvis.display(vis_data)

 1 frames
/usr/local/lib/python3.7/dist-packages/pyLDAvis/gensim_models.py in
_extract_data(topic_model, corpus, dictionary, doc_topic_dists)
     40         num_topics = len(topic_model.lda_alpha)
     41     else:
---> 42         num_topics = topic_model.num_topics
     43
     44     if doc_topic_dists is None:

AttributeError: 'CombinedTM' object has no attribute 'num_topics'
```

10%

```
# data = pyLDAvis.prepare(topic_term_dists, doc_topic_dists, doc_lengths, vocab, term_frequency, R=30, lambda
# # topic_term_dists: Matrix of topic-term probabilities
# # doc_topic_dists: Matrix of document-topic probabilities.
# # doc_lengths: The length of each document, i.e. the number of words in each document. The order of the num
# # vocab: List of all the words in the corpus used to train the model.
# # term_frequency: The count of each particular term over the entire corpus. The ordering of these counts sh
# # R=30: The number of terms to display in the barcharts of the visualization. Default is 30. Recommended to
# pyLDAvis.display(data)
```

✓ 0s completed at 1:08 PM