[Write](https://medium.com/m/signin?operation=register&redirect=https%3A%2F%2Fmedium.com%2Fnew-story&source=---two_column_layout_nav-----------------------new_post_topnav-----------)

[Sign up](https://medium.com/m/signin?operation=register&redirect=https%3A%2F%2Ftowardsdatascience.com%2Fa-beginners-guide-to-octis-optimizing-and-comparing-topic-models-is-simple-590554ec9ba6&source=post_page---two_column_layout_nav-----------------------global_nav-----------)

[Sign In](https://medium.com/m/signin?operation=login&redirect=https%3A%2F%2Ftowardsdatascience.com%2Fa-beginners-guide-to-octis-optimizing-and-comparing-topic-models-is-simple-590554ec9ba6&source=post_page---two_column_layout_nav-----------------------global_nav-----------)



[HANDS-ON TUTORIALS](https://towardsdatascience.com/tagged/hands-on-tutorials)

[**A beginner’s guide to OCTIS vol. 1: Optimizing and Comparing Topic Models Is Simple**](https://towardsdatascience.com/a-beginners-guide-to-octis-optimizing-and-comparing-topic-models-is-simple-590554ec9ba6)

The Python package with the highest number of integrated state-of-the-art topic models.

[[](https://emilrijcken.medium.com/?source=post_page-----590554ec9ba6--------------------------------)](https://emilrijcken.medium.com/?source=post_page-----590554ec9ba6--------------------------------)

[[Towards Data Science](https://towardsdatascience.com/?source=post_page-----590554ec9ba6--------------------------------)](https://towardsdatascience.com/?source=post_page-----590554ec9ba6--------------------------------)

[Emil Rijcken](https://emilrijcken.medium.com/?source=post_page-----590554ec9ba6--------------------------------)

·

[Follow](https://medium.com/m/signin?actionUrl=https%3A%2F%2Fmedium.com%2F_%2Fsubscribe%2Fuser%2F95ae6f4e7791&operation=register&redirect=https%3A%2F%2Ftowardsdatascience.com%2Fa-beginners-guide-to-octis-optimizing-and-comparing-topic-models-is-simple-590554ec9ba6&user=Emil+Rijcken&userId=95ae6f4e7791&source=post_page-95ae6f4e7791----590554ec9ba6---------------------post_header-----------)

Published in

[Towards Data Science](https://towardsdatascience.com/?source=post_page-----590554ec9ba6--------------------------------)

·

7 min read

·

Sep 2, 2021

242

4

Photo by [Universal Eye](https://unsplash.com/@universaleye?utm_source=medium&utm_medium=referral) on [Unsplash](https://unsplash.com/?utm_source=medium&utm_medium=referral" \t "_blank)

Topic models are promising generative statistical methods that aim to extract the hidden topics underlying a collection of documents. Typically, topic models have two matrices as output.

1. Topic-word-matrix (*vocabulary*x *number\_topics*) that indicates the probability of word *i* to appear in topic *k.*

2. Topic-document-matrix (*num\_topics*x *num\_documents*).

Then, the top-n words from this matrix with the highest probability are then used to represent a topic.

The most popular topic modeling method is Latent Dirichlet Allocation, and many articles are written about its workings and implementations. However, focusing on LDA only is restrictive and might be suboptimal for a given corpus. Other topic models might be better for a given corpus.

Recently, a new Python package has been released, OCTIS (Optimizing and Comparing Topic Models is Simple!). Based on its name, it won’t surprise you that OCTIS allows for easy comparison between topic models. The package contains multiple topic models, datasets, evaluation metrics and optimization options.

In this blog, I demonstrate how to get started with the package. In later blogs, I will go into more detail discussing topic model optimization and comparison.

Start by installing the package:

pip install octis

We need a dataset.

from octis.dataset.dataset import Dataset

OCTIS has four built-in datasets:



Let’s go with the smallest one:

dataset = Dataset()dataset.fetch\_dataset('BBC\_news')

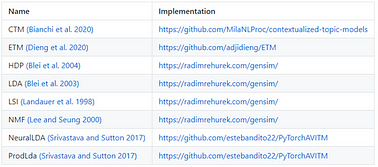
To select the other datasets, choose one of the names from the table (note the spelling of ‘BBC\_news’ in the code differs from the spelling in the table. The other names are the same).

Now, let’s see what the dataset looks like:

len(dataset.\_Dataset\_\_corpus)  
>>> 2225len(dataset.\_Dataset\_\_vocabulary)  
>>> 2949print(dataset.\_Dataset\_\_corpus[0:5])  
>>> [[‘broadband’, ‘ahead’, ‘join’, ‘internet’, ‘fast’, ‘accord’, ‘official’, ‘figure’, ‘number’, ‘business’, ‘connect’, ‘jump’, ‘report’, ‘broadband’, ‘connection’, ‘end’, ‘compare’, ‘nation’, ‘rank’, ‘world’, ‘telecom’, ‘body’, ‘election’, ‘campaign’, ‘ensure’, ‘affordable’, ‘high’, ‘speed’, ‘net’, ‘access’, ‘american’, ‘accord’, ‘report’, ‘broadband’, ‘increasingly’, ‘popular’, ‘research’, ‘shopping’, ‘download’, ‘music’, ‘watch’, ‘video’, ‘total’, ‘number’, ‘business’, ‘broadband’, ‘rise’, ‘end’, ‘compare’, ‘hook’, ‘broadband’, ‘subscriber’, ‘line’, ‘technology’, ‘ordinary’, ‘phone’, ‘line’, ‘support’, ‘high’, ‘data’, ‘speed’, ‘cable’, ‘lead’, ‘account’, ‘line’, ‘broadband’, ‘phone’, ‘line’, ‘connection’, ‘accord’, ‘figure’],[‘plan’, ‘share’, ‘sale’, ‘owner’, ‘technology’, ‘dominate’, ‘index’, ‘plan’, ‘sell’, ‘share’, ‘public’, ‘list’, ‘market’, ‘operate’, ‘accord’, ‘document’, ‘file’, ‘stock’, ‘market’, ‘plan’, ‘raise’, ‘sale’, ‘observer’, ‘step’, ‘close’, ‘full’, ‘public’, ‘icon’, ‘technology’, ‘boom’, ‘recently’, ‘pour’, ‘cold’, ‘water’, ‘suggestion’, ‘company’, ‘sell’, ‘share’, ‘private’, ‘technically’, ‘public’, ‘stock’, ‘start’, ‘trade’, ‘list’, ‘equity’, ‘trade’, ‘money’, ‘sale’, ‘investor’, ‘buy’, ‘share’, ‘private’, ‘filing’, ‘document’, ‘share’, ‘technology’, ‘firm’, ‘company’, ‘high’, ‘growth’, ‘potential’, ‘symbol’, ‘internet’, ‘telecom’, ‘boom’, ‘bubble’, ‘burst’, ‘recovery’, ‘fortune’, ‘tech’, ‘giant’, ‘dot’, ‘revive’, ‘fortune’],[‘mobile’, ‘rack’, ‘mobile’, ‘phone’, ‘celebrate’, ‘anniversary’, ‘weekend’, ‘mobile’, ‘phone’, ‘call’, ‘vodafone’, ‘network’, ‘veteran’, ‘day’, ‘mobile’, ‘phone’, ‘integral’, ‘part’, ‘modern’, ‘life’, ‘briton’, ‘handset’, ‘mobile’, ‘popular’, ‘handset’, ‘phone’, ‘rarely’, ‘call’, ‘portable’, ‘phone’, ‘commercial’, ‘mobile’, ‘service’, ‘launch’, ‘rest’, ‘world’, ‘set’, ‘network’, ‘call’, ‘walk’, ‘call’, ‘office’, ‘house’, ‘day’, ‘vodafone’, ‘firm’, ‘mobile’, ‘network’, ‘launch’, ‘service’, ‘spokesman’, ‘phone’, ‘launch’, ‘size’, ‘cost’, ‘battery’, ‘life’, ‘minute’, ‘hugely’, ‘popular’, ‘mid’, ‘status’, ‘symbol’, ‘young’, ‘business’, ‘fact’, ‘phone’, ‘radio’, ‘signal’, ‘communicate’, ‘easy’, ‘rack’, ‘customer’, ‘month’, ‘easy’, ‘forget’, ‘put’, ‘bid’, ‘document’, ‘forecast’, ‘total’, ‘market’, ‘forecast’, ‘vodafone’, ‘customer’, ‘vodafone’, ‘mobile’, ‘phone’, ‘operator’, ‘launch’, ‘launch’, ‘newcomer’, ‘operate’, ‘mobile’, ‘network’, ‘operator’, ‘technology’, ‘spectrum’, ‘phone’, ‘retire’, ‘call’, ‘global’, ‘system’, ‘mobile’, ‘widely’, ‘phone’, ‘technology’, ‘planet’, ‘call’, ‘digital’, ‘technology’, ‘introduce’, ‘thing’, ‘text’, ‘mobile’, ‘popular’],[‘launch’, ‘reconstruction’, ‘drive’, ‘appeal’, ‘peace’, ‘national’, ‘unity’, ‘important’, ‘find’, ‘solution’, ‘internal’, ‘conflict’, ‘damage’, ‘tsunami’, ‘cut’, ‘percentage’, ‘point’, ‘economic’, ‘growth’, ‘estimate’, ‘wave’, ‘leave’, ‘physical’, ‘damage’, ‘equal’, ‘economy’, ‘separately’, ‘lose’, ‘call’, ‘action’, ‘create’, ‘job’, ‘attend’, ‘ceremony’, ‘southern’, ‘town’, ‘join’, ‘government’, ‘opposition’, ‘politician’, ‘lay’, ‘foundation’, ‘stone’, ‘housing’, ‘project’, ‘intend’, ‘provide’, ‘home’, ‘tsunami’, ‘call’, ‘tragedy’, ‘start’, ‘beginning’, ‘rebuild’, ‘nation’, ‘country’, ‘natural’, ‘resource’, ‘fully’, ‘fight’, ‘add’, ‘due’, ‘arrive’, ‘revive’, ‘peace’, ‘talk’, ‘decade’, ‘long’, ‘conflict’, ‘government’, ‘force’, ‘tiger’, ‘separate’, ‘state’, ‘country’, ‘reconstruction’, ‘effort’, ‘hamper’, ‘tension’, ‘side’, ‘authority’, ‘initial’, ‘estimate’, ‘put’, ‘physical’, ‘damage’, ‘add’, ‘implication’, ‘economy’, ‘wide’, ‘broad’, ‘impact’, ‘substantial’, ‘detail’, ‘difficult’, ‘assess’, ‘early’, ‘stage’, ‘growth’, ‘inflation’, ‘balance’, ‘payment’, ‘foreign’, ‘exchange’, ‘reserve’, ‘expect’, ‘show’, ‘effect’, ‘lose’, ‘business’, ‘reconstruction’, ‘cost’, ‘industry’, ‘agricultural’, ‘production’, ‘affect’, ‘tourism’, ‘suffer’, ‘short’, ‘term’, ‘report’, ‘estimate’, ‘lose’, ‘job’, ‘industry’, ‘earning’, ‘tourism’, ‘expect’, ‘low’, ‘economic’, ‘growth’, ‘expect’, ‘previously’, ‘forecast’, ‘inflation’, ‘climb’, ‘compare’, ‘previous’, ‘estimate’, ‘major’, ‘export’, ‘suffer’, ‘expect’, ‘reconstruction’, ‘effort’, ‘require’, ‘high’, ‘import’, ‘damage’, ‘balance’, ‘payment’, ‘foreign’, ‘exchange’, ‘reserve’, ‘hard’, ‘press’, ‘international’, ‘reserve’, ‘pre’, ‘tsunami’, ‘level’, ‘total’, ‘month’, ‘worth’, ‘import’, ‘week’, ‘approve’, ‘request’, ‘freeze’, ‘loan’, ‘repayment’],[‘buy’, ‘giant’, ‘profit’, ‘soar’, ‘acquisition’, ‘big’, ‘firm’, ‘tax’, ‘profit’, ‘rise’, ‘expect’, ‘solid’, ‘growth’, ‘performance’, ‘sale’, ‘firm’, ‘world’, ‘big’, ‘volume’, ‘buy’, ‘acquisition’, ‘sale’, ‘volume’, ‘grow’, ‘month’, ‘sale’, ‘account’, ‘increase’, ‘sell’, ‘volume’, ‘big’, ‘term’, ‘sale’, ‘continue’, ‘demand’, ‘product’, ‘south’, ‘american’, ‘market’, ‘brazilian’, ‘arm’, ‘popular’, ‘expect’, ‘boost’, ‘turnover’, ‘business’, ‘analyst’, ‘strong’, ‘performance’, ‘boost’, ‘share’, ‘market’, ‘end’, ‘report’, ‘contrast’, ‘volume’, ‘sale’, ‘fall’, ‘central’, ‘european’, ‘sale’, ‘rise’, ‘net’, ‘profit’]]

That’s good; each document is a preprocessed list of tokens, so no need to conduct any preprocessing here.

OCTIS has eight built-in topic models:



(Make sure to cite the right paper if you implement one of the models)

Now, let’s train a NeuralLDA model with this data:

from octis.models.NeuralLDA import NeuralLDA

Other topic models can be imported similarly by replacing ‘NeuralLDA’ with the name from the table above (note, ‘ProdLda’ needs to be ‘ProdLDA’).

Initialize the model:

model = NeuralLDA(num\_topics=20)

All different topic models contain ‘num\_topics’. The other arguments differ per model and can be found on the implementation page. However, checking the source code is quicker to find the arguments:

import inspectprint(inspect.getsource(NeuralLDA))>>> class NeuralLDA(AVITM):def \_\_init\_\_(self, num\_topics=10, activation=’softplus’, dropout=0.2, learn\_priors=True, batch\_size=64, lr=2e-3,momentum=0.99, solver=’adam’, num\_epochs=100, reduce\_on\_plateau=False, prior\_mean=0.0,prior\_variance=None, num\_layers=2, num\_neurons=100, num\_samples=10, use\_partitions=True):…

The arguments in the constructor can be set manually. In this case, these are:

* num\_topics
* activation
* dropout
* learn\_priors
* batch\_size
* lr
* momentum
* solver
* num\_epochs
* reduce\_on\_plateau
* prior\_mean
* prior\_variance
* num\_layers
* num\_neurons
* num\_samples
* use\_partition

For now, we will simply stick with the default settings and set num\_topics= 20:

model = NeuralLDA(num\_topics=20)trained\_model = model.train\_model(dataset)#Now, you should see something like:Epoch: [1/100] Samples: [1557/155700] Train Loss: 987.1886277195729 Time: 0:00:00.352704Epoch: [1/100] Samples: [334/33400] Validation Loss: 982.9130479275823 Time: 0:00:00.020085Epoch: [2/100] Samples: [3114/155700] Train Loss: 990.8275228805395 Time: 0:00:00.362417Epoch: [2/100] Samples: [334/33400] Validation Loss: 982.0897677301647 Time: 0:00:00.010114Epoch: [3/100] Samples: [4671/155700] Train Loss: 981.4996826328677 Time: 0:00:00.362483Epoch: [3/100] Samples: [334/33400] Validation Loss: 965.4536162050898 Time: 0:00:00.010038

The trained\_model is a dictionary, with the following keys:

print(trained\_model.keys())>>> dict\_keys([‘topics’, ‘topic-document-matrix’, ‘topic-word-matrix’, ‘test-topic-document-matrix’])

The keys are the following:

1. *topics*: the list of word topics
2. *topic-word-matrix*: the distribution of the words of the vocabulary for each topic (dimensions: |num topics| x |vocabulary|)
3. *topic-document-matrix*: the distribution of the topics for each document of the training set (dimensions: |num topics| x |training documents|)
4. *test-document-topic-matrix*: the distribution of the topics for each document of the testing set (dimensions: |num topics| x |test documents|)

The first key topics consists of the list with the *n* words with the highest probability per topic from the topic-word-matrix.

Let’s see what these are:

for topic in trained\_model[‘topics’]:  
 print(“ “.join(topic))>>> claim chief meet budget official talk cut raise future economicwin side match final chance team season half great goalcompany market sale month share expect rise fall growth reportcharge competition link victim ago injury anti tragedy decision missgovernment plan party add election public give claim week torygame show give big play add back week put findcompany find user firm technology mobile service phone accord informationcase evidence order trial concern court common law clear powerenjoy tragedy rest carry navigate motion admit date opener technologicalchoose tragedy due motion navigate involve manage suffer stiff technologicalfilm good top award star director movie chance performance careerrival reveal drug short range break challenge battle focus fourthtragedy position seller navigate squeeze motion rare flanker overwhelming technologicalinclude follow number hit man release hold place record bandtragedy navigate seller earn join escape squeeze opener adoption technological

Typically, coherence is used to assess the quality of topics. Let’s check the coherence of this model as well:

from octis.evaluation\_metrics.coherence\_metrics import Coherence

Amongst the different coherence measures, c\_v has the highest correlation with human interpretation.

cv = Coherence(texts=dataset.get\_corpus(),topk=10, measure=’c\_v’)  
print(‘Coherence: ‘ + str(cv.score(trained\_model)))  
>>> Coherence: 0.4963712149432245

(In contrast to the typical topic modeling notation, ‘topk’ indicates the number of words to include in a topic, not the number of topics. Since our topic model has ten words per topic, ‘topk’ cannot exceed 10).

In addition, OCTIS also has the option to calculate the so-called ‘diversity score’, which is the percentage of unique words in the topic. This score is another indication of the quality of the produced topics. If there are not many unique words, then many topics are similar and are not very meaningful.

from octis.evaluation\_metrics.diversity\_metrics import TopicDiversitydiversity = TopicDiversity(topk=10)  
print('Diversity score: '+str(diversity.score(trained\_model)))  
>>> Diversity score: 0.8533333333333334

Both the coherence and the diversity are primarily interesting for comparing the topics of different topic models. In the next blog, I will show how to optimize topic models.

If you like this work, you are likely interested in topic modeling. In that case, you might be interested in the following as well.

**We have created a new topic modeling algorithm called FLSA-W**(the official page is [here](https://ieeexplore.ieee.org/abstract/document/9660139), but you can see the paper [here](https://pure.tue.nl/ws/portalfiles/portal/243684581/A_Comparative_Study_of_Fuzzy_Topic_Models_and_LDA_in_terms_of_Interpretability.pdf)).

**FLSA-W outperforms other state-of-the-art algorithms (such as LDA, ProdLDA, NMF, CTM and more) on several open datasets.**[**This work**](https://pure.tue.nl/ws/files/222725628/Pure_ExperimentalStudyOfFlsa_wForTopicModeling.pdf)**has been submitted but is not peer-reviewed yet.**

**If you want to use FLSA-W, you can download**[**the FuzzyTM package**](https://pypi.org/project/FuzzyTM/)**or the flsamodel in Gensim.** For citations, [please use this paper](https://ieeexplore.ieee.org/abstract/document/9882661?casa_token=UsYg7SvoSioAAAAA%3A3ltCVZexA9-lPveuGVeRDh5VQW6rw0pVRDxmYk39tXbx13u4OuB2sTEFZzIGJCkdRiZBg0eJ).

[Topic Modeling](https://medium.com/tag/topic-modeling?source=post_page-----590554ec9ba6---------------topic_modeling-----------------)

[Machine](https://medium.com/tag/machine-learning?source=post_page-----590554ec9ba6---------------machine_learning-----------------)