In this article, we will go through the evaluation of Topic Modelling by introducing the concept of Topic coherence, as topic models give no guaranty on the interpretability of their output. Topic modeling provides us with methods to organize, understand and summarize large collections of textual information. There are many techniques that are used to obtain topic models. Latent Dirichlet Allocation (LDA) is a widely used topic modeling technique to extract topic from the textual data.

Topic models learn topics—typically represented as sets of important words—automatically from unlabelled documents in an unsupervised way. This is an attractive method to bring structure to otherwise unstructured text data, but Topics are not guaranteed to be well interpretable, therefore, coherence measures have been proposed to distinguish between good and bad topics.

Let’s start learning with a simple example and then we move to a technical part of topic coherence.

*Imagine you are a lead quality analyst sitting at location X at a logistics company and you want to check the quality of your dispatch product at 4 different locations: A, B, C, D. One way is to collect the reviews from various people – for example- “whether they receive product in good condition”, Did they receive on time”.* You may need to improve your process if most people give you bad reviews. So, basically, you are evaluating on the qualitative approach, as there is no quantitative measure involved, which can tell you how much worse your dispatch product quality at A is compared to dispatch quality at B.

To arrive at the quantitative measure, your central lab at X set up 4 different quality lab Kiosk at A, B, C and D to check the dispatch product quality (let’s say quality defined by % of conformance as per some predefined standards). Now, while sitting at the central lab, you can get the quality values from 4 Kiosks and can compute your overall quality. You don’t need to rely on people reviews, as you have a good quantitative measure of quality.

Here the analogy comes in:  
The dispatch product here is the topics from some topic modeling algorithm such as LDA. The qualitative approach is to test the topics on their human interpretability by presenting them to humans and taking their input on them. The quality lab setup is the topic coherence framework, which is grouped into 4 following dimensions:

* Segmentation: A lot of dispatch product divided into different sub-lot sizes, such that each sub-lot product are different.
* Probability Estimation: Quantitative Measurement of sub lot quality.
* Confirmation Measure: Determine quality as per some predefined standard (say % conformance) and assign some number to qualify. For example, 75% of products are good quality as per XXX standard.
* Aggregation: It’s the central lab where you combine all the quality numbers and derive a single number for overall quality.

From a technical point of view, Coherence framework is represented as a composition of parts that can be combined. The parts are grouped into dimensions that span the configuration space of coherence measures. Each dimension is characterized by a set of exchangeable components.

First, the word set t is segmented into a set of pairs of word subsets S. Second, word probabilities P are computed based on a given reference corpus. Both, the set of word subsets S as well as the computed probabilities P are consumed by the confirmation measure to calculate the agreements ϕ of pairs of S. Last, those values are aggregated to a single coherence value c.

There are 2 measures in Topic coherence :

**Intrinsic Measure**

It is represented as UMass. It measures to compare a word only to the preceding and succeeding words respectively, so need ordered word set.It uses as pairwise score function which is the empirical conditional log-probability with smoothing count to avoid calculating the logarithm of zero.

**Extrinsic Measure**

It is represented as UCI. In UCI measure, every single word is paired with every other single word. The UCIcoherence uses pointwise mutual information (PMI).  
Both Intrinsic and Extrinsic measure compute the coherence score c (sum of pairwise scores on the words w1, …, wn used to describe the topic).  
If you are interested to learn in more detail, refer this paper :- [Exploring the Space of Topic Coherence Measures](http://svn.aksw.org/papers/2015/WSDM_Topic_Evaluation/public.pdf)

**Implementation in Python**

Amazon fine food review dataset, publicly available on Kaggle is used for this paper. Since dataset is very huge, only 10,000 reviews are considered. Since we are focusing on topic coherence, I am not going in details for data pre-processing here.

It consists of following steps:

**Step 1**

First step is loading packages, Data and Data pre-processing.  
We created dictionary and corpus required for Topic Modeling: The two main inputs to the LDA topic model are the dictionary and the corpus. Gensim creates a unique id for each word in the document. The produced corpus shown above is a mapping of (word\_id, word\_frequency). For example, (0, 1) below in the output implies, word id 0 occurs once in the first document.

# Import required packages

import numpy as np

import logging

import pyLDAvis.gensim

import json

import warnings

warnings.filterwarnings('ignore') # To ignore all warnings that arise here to enhance clarity

from gensim.models.coherencemodel import CoherenceModel

from gensim.models.ldamodel import LdaModel

from gensim.corpora.dictionary import Dictionary

from numpy import array

# Import dataset

p\_df = pd.read\_csv('C:/Users/kamal/Desktop/R project/Reviews.csv')

# Create sample of 10,000 reviews

p\_df = p\_df.sample(n = 10000)

# Convert to array

docs =array(p\_df['Text'])

# Define function for tokenize and lemmatizing

from nltk.stem.wordnet import WordNetLemmatizer

from nltk.tokenize import RegexpTokenizer

def docs\_preprocessor(docs):

tokenizer = RegexpTokenizer(r'\w+')

for idx in range(len(docs)):

docs[idx] = docs[idx].lower() # Convert to lowercase.

docs[idx] = tokenizer.tokenize(docs[idx]) # Split into words.

# Remove numbers, but not words that contain numbers.

docs = [[token for token in doc if not token.isdigit()] for doc in docs]

# Remove words that are only one character.

docs = [[token for token in doc if len(token) > 3] for doc in docs]

# Lemmatize all words in documents.

lemmatizer = WordNetLemmatizer()

docs = [[lemmatizer.lemmatize(token) for token in doc] for doc in docs]

return docs

# Perform function on our document

docs = docs\_preprocessor(docs)

#Create Biagram & Trigram Models

from gensim.models import Phrases

# Add bigrams and trigrams to docs,minimum count 10 means only that appear 10 times or more.

bigram = Phrases(docs, min\_count=10)

trigram = Phrases(bigram[docs])

for idx in range(len(docs)):

for token in bigram[docs[idx]]:

if '\_' in token:

# Token is a bigram, add to document.

docs[idx].append(token)

for token in trigram[docs[idx]]:

if '\_' in token:

# Token is a bigram, add to document.

docs[idx].append(token)

#Remove rare & common tokens

# Create a dictionary representation of the documents.

dictionary = Dictionary(docs)

dictionary.filter\_extremes(no\_below=10, no\_above=0.2)

#Create dictionary and corpus required for Topic Modeling

corpus = [dictionary.doc2bow(doc) for doc in docs]

print('Number of unique tokens: %d' % len(dictionary))

print('Number of documents: %d' % len(corpus))

print(corpus[:1])

*Number of unique tokens: 4214*

*Number of documents: 10000*

*[[(0, 1), (1, 1), (2, 2), (3, 1), (4, 1), (5, 1), (6, 4), (7, 2), (8, 1), (9, 2), (10, 5), (11, 8), (12, 3), (13, 1), (14, 2), (15, 1), (16, 2), (17, 2), (18, 3), (19, 3), (20, 1), (21, 1), (22, 1), (23, 2), (24, 1), (25, 1), (26, 1), (27, 1), (28, 1), (29, 2), (30, 1), (31, 2), (32, 2), (33, 1), (34, 1), (35, 1), (36, 1), (37, 2), (38, 3)]]*

**Step 2**

We have everything required to train the LDA model. In addition to the corpus and dictionary, we need to provide the number of topics as well.Set number of topics=5.

# Set parameters.

num\_topics = 5

chunksize = 500

passes = 20

iterations = 400

eval\_every = 1

# Make a index to word dictionary.

temp = dictionary[0] # only to "load" the dictionary.

id2word = dictionary.id2token

lda\_model = LdaModel(corpus=corpus, id2word=id2word, chunksize=chunksize, \

alpha='auto', eta='auto', \

iterations=iterations, num\_topics=num\_topics, \

passes=passes, eval\_every=eval\_every)

# Print the Keyword in the 5 topics

print(lda\_model.print\_topics())

*[(0, '0.067\*"coffee" + 0.012\*"strong" + 0.011\*"green" + 0.010\*"vanilla" + 0.009\*"just\_right" + 0.009\*"cup" + 0.009\*"blend" + 0.008\*"drink" + 0.008\*"http\_amazon" + 0.007\*"bean"'), (1, '0.046\*"have\_been" + 0.030\*"chocolate" + 0.021\*"been" + 0.021\*"gluten\_free" + 0.018\*"recommend" + 0.017\*"highly\_recommend" + 0.014\*"free" + 0.012\*"cooky" + 0.012\*"would\_recommend" + 0.011\*"have\_ever"'), (2, '0.028\*"food" + 0.018\*"amazon" + 0.016\*"from" + 0.015\*"price" + 0.012\*"store" + 0.011\*"find" + 0.010\*"brand" + 0.010\*"time" + 0.010\*"will" + 0.010\*"when"'), (3, '0.019\*"than" + 0.013\*"more" + 0.012\*"water" + 0.011\*"this\_stuff" + 0.011\*"better" + 0.011\*"sugar" + 0.009\*"much" + 0.009\*"your" + 0.009\*"more\_than" + 0.008\*"which"'), (4, '0.010\*"would" + 0.010\*"little" + 0.010\*"were" + 0.009\*"they\_were" + 0.009\*"treat" + 0.009\*"really" + 0.009\*"make" + 0.008\*"when" + 0.008\*"some" + 0.007\*"will"')]*

**Step 3**

LDA is an unsupervised technique, meaning that we don’t know prior to running the model how many topics exits in our corpus.You can use LDA visualization tool pyLDAvis, tried a few numbers of topics and compared the results. Topic coherence is one of the main techniques used to estimate the number of topics.We will use both UMass and c\_v measure to see the coherence score of our LDA model.

**Using c\_v Measure**

# Compute Coherence Score using c\_v

coherence\_model\_lda = CoherenceModel(model=lda\_model, texts=docs, dictionary=dictionary, coherence='c\_v')

coherence\_lda = coherence\_model\_lda.get\_coherence()

print('\nCoherence Score: ', coherence\_lda)

*Coherence Score: 0.359704263036*

**Using UMass Measure**

# Compute Coherence Score using UMass

coherence\_model\_lda = CoherenceModel(model=lda\_model, texts=docs, dictionary=dictionary, coherence="u\_mass")

coherence\_lda = coherence\_model\_lda.get\_coherence()

print('\nCoherence Score: ', coherence\_lda)

*Coherence Score: -2.60591638507*

**Step 4**

The last step is to find the optimal number of topics.We need to build many LDA models with different values of the number of topics (k) and pick the one that gives the highest coherence value. Choosing a ‘k’ that marks the end of a rapid growth of topic coherence usually offers meaningful and interpretable topics. Picking an even higher value can sometimes provide more granular sub-topics. If you see the same keywords being repeated in multiple topics, it’s probably a sign that the ‘k’ is too large.

**Using c\_v Measure**

def compute\_coherence\_values(dictionary, corpus, texts, limit, start=2, step=3):

"""

Compute c\_v coherence for various number of topics

Parameters:

----------

dictionary : Gensim dictionary

corpus : Gensim corpus

texts : List of input texts

limit : Max num of topics

Returns:

-------

model\_list : List of LDA topic models

coherence\_values : Coherence values corresponding to the LDA model with respective number of topics

"""

coherence\_values = []

model\_list = []

for num\_topics in range(start, limit, step):

model=LdaModel(corpus=corpus, id2word=dictionary, num\_topics=num\_topics)

model\_list.append(model)

coherencemodel = CoherenceModel(model=model, texts=texts, dictionary=dictionary, coherence='c\_v')

coherence\_values.append(coherencemodel.get\_coherence())

return model\_list, coherence\_values

Create a model list and plot Coherence score against a number of topics

model\_list, coherence\_values = compute\_coherence\_values(dictionary=dictionary, corpus=corpus, texts=docs, start=2, limit=40, step=6)

# Show graph

import matplotlib.pyplot as plt

limit=40; start=2; step=6;

x = range(start, limit, step)

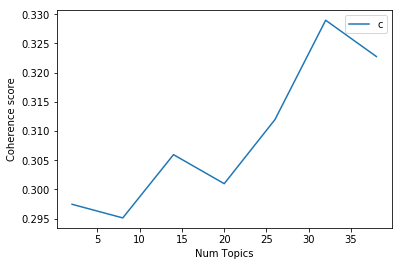
plt.plot(x, coherence\_values)

plt.xlabel("Num Topics")

plt.ylabel("Coherence score")

plt.legend(("coherence\_values"), loc='best')

plt.show()

Gives this plot:  
[](https://datascienceplus.com/wp-content/uploads/2018/05/cv-1.png)

The above plot shows that coherence score increases with the number of topics, with a decline between 15 to 20.Now, choosing the number of topics still depends on your requirement because topic around 33 have good coherence scores but may have repeated keywords in the topic. Topic coherence gives you a good picture so that you can take better decision.

You can try the same with U mass measure.

**Conclusion**

To conclude, there are many other approaches to evaluate Topic models such as Perplexity, but its poor indicator of the quality of the topics.Topic Visualization is also a good way to assess topic models. Topic Coherence measure is a good way to compare difference topic models based on their human-interpretability.The u\_mass and c\_v topic coherences capture the optimal number of topics by giving the interpretability of these topics a number called coherence score.