01-Topic-Modeling

November 6, 2017

```
In [6]: %%capture
        !rm -rf data/*
        !unzip data.zip -d data/
        !pip install --no-cache-dir pyldavis
        from datascience import *
        import numpy as np
        import matplotlib.pyplot as plt
        import pandas as pd
        import pyLDAvis
        import pyLDAvis.sklearn
        import pickle
        %matplotlib inline
```

1 Topic Modeling in Python

In Lisa Rhody's article, "Topic Modeling and Figurative Language", she uses LDA topic modeling to look at ekphrasis poetry. She argues that ekphrasis poetry is particulary well-suited to an LDA analysis because of the assumption of a previously existing set of topics. She's able to extract a number of topics, each constituted of a set of words and probabilities. While we don't have Rhody's corpus, we can use this technique on any large text corpus. We'll use a corpus of novels curated by Andrew Piper.

1.1 Corpus Description

We'll look at an English-language subset of Andrew Piper's novel corpus, totaling 150 novels by British and American authors spanning the years 1771-1930. These texts are each in a separate plaintext file in our `data` folder. Metadata is contained in a spreadsheet distributed with the novel files by the txtLAB at McGill.

The metadata provided describes the corpus that exists as `.txt` files. So let's first read in the metadata:

```
In [7]: metadata_tb = Table.read_table('data/txtlab_Novel150_English.csv')
        metadata_tb.show(5)
```

```
<IPython.core.display.HTML object>
```

Before we go anywhere, let's randomly shuffle the rows so that we don't have them ordered by dates or anything else:

```
In [8]: np.random.seed(0)
        metadata_tb = Table.from_df(metadata_tb.to_df().sample(frac=1))
        metadata_tb.show(5)
```

```
<IPython.core.display.HTML object>
```

We can see the column variables we have in the metadata with the `.labels` attribute:

```
In [10]: metadata_tb.labels
```

```
Out[10]: ('filename',
          'id',
          'language',
          'date',
          'author',
          'title',
          'gender',
          'person',
          'length')
```

To clarify:
`filename`: Name of file on disk
`id`: Unique ID in Piper corpus
`language`: Language of novel
`date`: Initial publication date
`author`: Author's name
`title`: Title of novel
`gender`: Authorial gender
`person`: Textual perspective
`length`: Number of tokens in novel
We see a list of `filenames` in the table, these map into a folder we have called `txtlab_Novel150_English`:

```
In [11]: !ls data/txtlab_Novel150_English/
```

```
EN_1771_Mackenzie,Henry_TheManofFeeling_Novel.txt
EN_1771_Smollett,Tobias_TheExpedictionofHenryClinker_Novel.txt
EN_1778_Burney,Fanny_Evelina_Novel.txt
EN_1782_Burney,Fanny_Cecilia_Novel.txt
EN_1786_Beckford,William_Vathek_Novel.txt
EN_1788_Wollstonecraft,Mary_Mary_Novel.txt
EN_1790_Radcliffe,Ann_ASicilianRomance_Novel.txt
EN_1794_Godwin,William_CalebWilliams_Novel.txt
EN_1794_Radcliffe,Ann_TheMysteriesofUdolpho_Novel.txt
EN_1794_Rowson,Susanna_CharlotteTemple_Novel.txt
```

EN_1795_Lewis,Matthew_TheMonk_Novel.txt
EN_1796_Bonhote,Elizabeth_BungayCastle_Novel.txt
EN_1796_Burney,Fanny_Camilla_Novel.txt
EN_1796_Hays,Mary_EmmaCourtney_Novel.txt
EN_1797_Foster,HannahWebster_TheCoquette_Novel.txt
EN_1798_Brown,CharlesBrockden_Wieland_Novel.txt
EN_1798_Wollstonecraft,Mary_Maria_Novel.txt
EN_1799_Brown,CharlesBrockden_ArthurMervyn_Novel.txt
EN_1800_Edgeworth,Maria_CastleRackrent_Novel.txt
EN_1801_Edgeworth,Maria_Belinda_Novel.txt
EN_1804_Opie,Amelia_AdelineMowbray_Novel.txt
EN_1805_Lewis,Matthew_TheBravoofVenice_Novel.txt
EN_1806_Edgeworth,Maria_Leonora_Novel.txt
EN_1809_More,Hannah_CoelebsinSearchofaWife_Novel.txt
EN_1811_Austen,Jane_SenseandSensibility_Novel.txt
EN_1813_Austen,Jane_PrideandPrejudice_Novel.txt
EN_1814_Austen,Jane_MansfieldPark_Novel.txt
EN_1814_Scott,Walter_Waverley_Novel.txt
EN_1815_Peacock,ThomasLove_HeadlongHall_Novel.txt
EN_1817_Scott,Walter_RobRoy_Novel.txt
EN_1818_Peacock,ThomasLove_NightmareAbbey_Novel.txt
EN_1818_Shelley,Mary_Frankenstein_Novel.txt
EN_1819_Shelley,Mary_Mathilda_Novel.txt
EN_1820_Scott,Walter_Ivanhoe_Novel.txt
EN_1821_Galt,John_AnnalsoftheParish_Novel.txt
EN_1821_Peacock,ThomasLove_MaidMarian_Novel.txt
EN_1822_Hogg,James_ThreePerilsofMan_Novel.txt
EN_1823_Cooper,JamesFenimore_ThePioneers_Novel.txt
EN_1826_Cooper,JameFenimore_TheLastoftheMohicans_Novel.txt
EN_1826_Disraeli,Benjamin_VivianGrey_Novel.txt
EN_1836_Child,Lydia_Philothea_Novel.txt
EN_1837_Disraeli,Benjamin_Venetia_Novel.txt
EN_1837_Trollope,FrancesMilton_TheVicarofWrexham_Novel.txt
EN_1838_Martineau,Harriet_Deerbrook_Novel.txt
EN_1838_Poe,EdgarAllen_TheNarrativeofArthurGordonPym_Novel.txt
EN_1843_Borrow,George_TheBibleinSpain_Novel.txt
EN_1844_Yonge,Charlotte_Abbeychurch_Novel.txt
EN_1847_Aguilar,Grace_HomeInfluence_Novel.txt
EN_1847_Bronte,Charlotte_JaneEyre_Novel.txt
EN_1847_Bronte,Emily_WutheringHeights_Novel.txt
EN_1847_Thackeray,William_VanityFair_Novel.txt
EN_1848_Bronte,Ann_TheTenantofWildfellHall_Novel.txt
EN_1848_Gaskell,Elizabeth_MaryBarton_Novel.txt
EN_1849_Kingsley,Charles_AltonLocke_Novel.txt
EN_1850_Aguilar,Grace_ValeofCedars_Novel.txt
EN_1850_Hawthorne,Nathaniel_TheScarletLetter_Novel.txt
EN_1850_Yonge,Charlotte_Henrietta'sWish_Novel.txt
EN_1851_Hawthorne,Nathaniel_TheHouseoftheSevenGables_Novel.txt

EN_1851_Melville,Hermann_MobyDick_Novel.txt
EN_1852_Collins,Wilkie_Basil_Novel.txt
EN_1853_Craik,Dinah_Agatha'sHusband_Novel.txt
EN_1853_Kingsley,Charles_Hypatia_Novel.txt
EN_1853_Stowe,HarrietBeecher_UncleTom'sCabin_Novel.txt
EN_1853_Yonge,Charlotte_TheHeirofRedcliffe_Novel.txt
EN_1854_Gaskell,Elizabeth_NorthandSouth_Novel.txt
EN_1855_Trollope,FrancesMilton_TheWidowBarnaby_Novel.txt
EN_1856_Craik,Dinah_JohnHalifax_Novel.txt
EN_1857_Trollope,Anthony_BarchesterTowers_Novel.txt
EN_1859_Dickens,Charles_ATaleofTwoCities_Novel.txt
EN_1860_Collins,Wilkie_TheWomaninWhite_Novel.txt
EN_1860_Eliot,George_TheMillontheFloss_Novel.txt
EN_1861_Dickens,Charles_GreatExpectations_Novel.txt
EN_1862_Braddon,Mary_LadyAudley'sSecret_Novel.txt
EN_1862_Eliot,George_Romola_Novel.txt
EN_1864_Braddon,Mary_HenryDunbar_Novel.txt
EN_1865_Carroll,Lewis_Alice'sAdventureinWonderland_Novel.txt
EN_1869_Alcott,Louisa_LittleWomen_Novel.txt
EN_1869_Blackmore,R.D._LornaDoone_Novel.txt
EN_1869_Trollope,Anthony_PhineasFinn_Novel.txt
EN_1871_Carroll,Lewis_ThroughtheLookingGlass.txt
EN_1874_Hardy,Thomas_FarFromtheMaddingCrowd_Novel.txt
EN_1876_Trollope,FrancesEleanor_ACharmingFellow_Novel.txt
EN_1876_Twain,Mark_TheAdventuresofTomSawyer_Novel.txt
EN_1877_Sewell,Anna_BlackBeauty_Novel.txt
EN_1881_James,Henry_PortraitofaLady_Novel.txt
EN_1882_Stevenson,RobertLouis_TreasureIsland_Novel.txt
EN_1883_Braddon,Mary_TheGoldenCalf_Novel.txt
EN_1884_Lyall,Edna_WeTwo_Novel.txt
EN_1884_Twain,Mark_TheAdventuresofHuckleberryFinn_Novel.txt
EN_1885_Barr,Amelia_JanVeeder'sWife_Novel.txt
EN_1886_Stevenson,RobertLouis_JekyllandHyde_Novel.txt
EN_1887_Bellamy,Edward_LookingBackward_Novel.txt
EN_1888_Trollope,FrancesEleanor_ThatUnfortunateMarriage_Novel.txt
EN_1888_Ward,Mrs.Humphry_RobertElsmere_Novel.txt
EN_1889_Doyle,ArthurConan_TheMysteryoftheCloomber_Novel.txt
EN_1890_Broughton,Rhoda_Alas!_Novel.txt
EN_1890_Chopin,Kate_AtFault_Novel.txt
EN_1890_Wilde,Oscar_ThePictureofDorianGray_Novel.txt
EN_1891_Doyle,ArthurConan_TheDoingsofRafflesHaw_Novel.txt
EN_1891_Gissing,George_NewGrubStreet_Novel.txt
EN_1891_Hardy,Thomas_TessoftheD'Urbervilles_Novel.txt
EN_1893_Gissing,George_TheOddWomen_Novel.txt
EN_1893_Grand,Sarah_TheHeavenlyTwins_Novel.txt
EN_1893_Harraden,Beatrice_ShipsThatPassintheNight_Novel.txt
EN_1894_Freeman,MaryWilkins_Pembroke_Novel.txt
EN_1894_Hope,Anthony_ThePrisonerofZenda_Novel.txt

```
EN_1894_Kipling,Rudyard_TheJungleBook_Novel.txt
EN_1895_Crane,Stephen_TheRedBadgeofCourage_Novel.txt
EN_1895_Wells,H.G._TheTimeMachine_Novel.txt
EN_1897_Stoker,Bram_Dracula_Novel.txt
EN_1898_Crockett,SR_TheRedAxe_Novel.txt
EN_1899_Chopin,Kate_TheAwakening_Novel.txt
EN_1899_Conrad,Joseph_HeartofDarkness_Novel.txt
EN_1900_Barr,Amelia_TheMaidofMaidenLane_Novel.txt
EN_1900_Dreiser,Theodore_SisterCarrie_Novel.txt
EN_1900_Kipling,Rudyard_Kim_Novel.txt
EN_1901_Norris,Frank_TheOctopus_Novel.txt
EN_1902_Bellamy,Edward_Eleonora_Novel.txt
EN_1902_Bennett,Arnold_GrandBabylonHotel_Novel.txt
EN_1903_James,Henry_TheAmbassadors_Novel.txt
EN_1903_London,Jack_TheCalloftheWild_Novel.txt
EN_1903_Norris,Frank_ThePit_Novel.txt
EN_1904_Murfree,MaryNoailles_TheFrontiersman_Novel.txt
EN_1905_Orczy,Emma_TheScarletPimpernel_Novel.txt
EN_1905_Wharton,Edith_TheHouseofMirth_Novel.txt
EN_1906_London,Jack_WhiteFang_Novel.txt
EN_1906_Sinclair,Upton_TheJungle_Novel.txt
EN_1906_Stein,Gertrude_ThreeLives_Novel.txt
EN_1908_Forster,E.M._ARoomWithaView_Novel.txt
EN_1910_Forster,E.M._HowardsEnd_Novel.txt
EN_1911_Barrie,J.M._PeterPan_Novel.txt
EN_1911_Wharton,Edith_EthanFrome_Novel.txt
EN_1912_Cather,Willa_Alexander'sBridge_Novel.txt
EN_1912_Dreiser,Theodore_TheFinancier_Novel.txt
EN_1913_Lawrence,D.H._SonsandLovers_Novel.txt
EN_1915_Ford,FordMadox_TheGoodSoldier_Novel.txt
EN_1916_Joyce,James_APortraitoftheArtistasaYoungMan_Novel.txt
EN_1917_Cahan,Abraham_TheRiseofDavidLevinsky_Novel.txt
EN_1917_Lewis,Sinclair_TheInnocents_Novel.txt
EN_1917_Webb,Mary_GonetoEart_Novel.txt
EN_1918_Lewis,Sinclai_TheJob_Nove.txt
EN_1920_DosPassos,John_ThreeSoldiers_Novel.txt
EN_1920_Fitzgerald,FScott_ThisSideofParadise_Novel.txt
EN_1920_Wharton,Edith_TheAgeofInnocence_Novel.txt
EN_1922_Fitzgerald,FScott_TheBeautifulandtheDamned_Novel.txt
EN_1922_Joyce,James_Ulysses_Novel.txt
EN_1925_Woolf,Virginia_Mrs.Dalloway_Novel.txt
EN_1927_Woolf,Virginia_TotheLighthouse_Novel.txt
EN_1928_Woolf,Virginia_Orlando_Novel.txt
EN_1930_Mansfield,Katherine_TheAloe_Novel.txt
```

We can then read in the full text for each novel by iterating through the column, reading each file and appending the string to our `novel_list`:

```
In [12]: # create empty list, entries will be list of tokens from each novel
         novel_list = []

         # iterate through filenames in metadata table
         for filename in metadata_tb['filename']:

             # read in novel text as single string
             with open('data/txtlab_Novel150_English/'+filename, 'r') as f:
                 novel = f.read()

             # clean up (no titles)
             toks = novel.split()  # split to tokens
             toks = [t for t in toks if not t.istitle() and not t.isupper()]  # quick & dirty 
             novel = ' '.join(toks)  # join to single string

             # add string
             novel_list.append(novel)
```

Let's double check they all came through:

```
In [13]: len(novel_list)
```

```
Out[13]: 150
```

And look at the first 200 characters of the fourth novel:

```
In [14]: metadata_tb['author'][3], metadata_tb['title'][3], novel_list[3][:200]
```

```
Out[14]: ('Crane,Stephen',
          'TheRedBadgeofCourage',
          '1 cold passed reluctantly from the earth, and the retiring fogs revealed an army st
```

1.2 Document Term Matrix

Now we need to make a document term matrix, just as we have in the past two classes. We can pull in our `CountVectorizer` from `sklearn` again to create our dtm:

```
In [15]: from sklearn.feature_extraction.text import CountVectorizer
```

While you may not have seen the importance of `max_features`, `max_df` and `min_df` before, for topic modeling this is extremely important, because otherwise your topics will not be super coherent.

Let's start out with this:

- `max_features` = 5000 (i.e. only include 5000 tokens in our dtm)
- `max_df` = .8 (i.e. don't keep any tokens that appear in > 80% of the documents)
- `min_df` = 5 (i.e. only keep the token if it appears in > 5 documents)

We'll add in a `stop_words='english'` too, which automatically uses its own stopwords list to remove from our dtm:

```
In [16]: cv = CountVectorizer(max_features=5000, stop_words='english', max_df=0.80, min_df=5)
```

As with most machine learning approaches, to validate your model you need training and testing partitions. Since we don't have any labels (topic modeling is *unsupervised* machine learning), we just need to do this for the novel strings:

```
In [17]: train = novel_list[:120]
         test = novel_list[120:]
```

Now we can use our `cv` to `fit_transform` our training list of novels (strings!):

```
In [17]: dtm = cv.fit_transform(train)


        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-17-8877df9ffd1f> in <module>()
    ----> 1 dtm = cv.fit_transform(train)


        NameError: name 'cv' is not defined
```

To get our words back out we'll use the method `get_feature_names()`

```
In [19]: dtm_feature_names = cv.get_feature_names()
         dtm_feature_names[:10]

Out[19]: ['abandon',
          'abandoned',
          'abbey',
          'abhorrence',
          'abide',
          'abilities',
          'ability',
          'abode',
          'abominable',
          'abrupt']
```

We can double check that our feature limit was enforced by calling `len` on the `dtm_feature_names`:

```
In [20]: len(dtm_feature_names)

Out[20]: 5000
```

We can throw our dtm into a `Table` like we have before too:

```
In [21]: dtm_tb = Table(dtm_feature_names).with_rows(dtm.toarray())
         dtm_tb.show(5)
```

```
<IPython.core.display.HTML object>
```

1.3 Topic Modeling

1.3.1 Latent Dirichlet Allocation (LDA) Models

LDA reflects an intuition that words in a text are not merely chosen at random but are drawn from underlying concepts (the so-called "latent variables"). The goal of LDA is to look across many texts in order to reverse engineer these concepts by finding words that tend to cluster with one another. For this reason, LDA has been referred to as "the mother of all word collocation techniques."

`sklearn` has the `LatentDirichletAllocation` function:

```
In [22]: from sklearn.decomposition import LatentDirichletAllocation
```

Let's check the doc string:

```
In [23]: LatentDirichletAllocation?
```

Importantly, we'll note:

`n_components`: This is the number of topics. Choosing this is the art of Topic Modeling

`max_iter`: TM initially uses random distribution, and iteratively tweaks model

Let's just say we'll look for 10 topics. We'll do a `max_iter` of 5. Generally, the higher `max_iter` volume the better opportunity to the model has to accurately tune, but it also takes much longer.

```
In [24]: lda = LatentDirichletAllocation(n_components=10, max_iter=5)
```

Before we `fit` the model, we need to remember that with a lot of these probabilistic models random number generators are used to star the algorithm. If we want our results to be reproducible, we need to set the random seed of the math library we use, in this case `numpy`:

```
In [25]: np.random.seed(0)
```

Now we just `fit` the model, as we've done with all `sklearn` models! This may take a while, a lot is going on:

```
In [26]: lda_model = lda.fit(dtm)
```

```
/srv/app/venv/lib/python3.6/site-packages/sklearn/decomposition/online_lda.py:532: Deprecation
  DeprecationWarning)
```

1.3.2 Evaluation

One measure of the model's fit is perplexity, with which we can judge how well the model fits the data. We need to call this on our `test` portion after it's been transformed into a dtm:

```
In [27]: lda_model.perplexity(cv.transform(test))
```

```
Out[27]: 5201.6432940072091
```

NOTE: Currently `sklearns` perplexity algorithm is broken.

The lower the perplexity, the better the fit of the model. So one way to get the optimal number of topics would be to loop through several numbers of topics and minimize the perplexity value.

Unfortunately, it has been shown time and again that minimizing perplexity does not actually separate topics into coherent groups that humans would.

1.3.3 Choosing the best model

Since traditional metrics of evaluating a model's accuracy have not proven to conform to human understanding, a new appraoch was developed by David Minmo in 2011.

> this score measures how much, within the words used to describe a topic, a common word is in average a good predictor for a less common word. (More on topic coherency.)

Here we look for the highest value. This algorithm has only been implemented in the Python `gensim` library. I ran the following code for you on a remote server because it takes a while!

```
import pickle
from joblib import Parallel, delayed
import multiprocessing


def try_topic_number(i):
    lda_model = gensim.models.LdaModel(
        corpus,
        num_topics=i,
        id2word=dictionary,
        iterations=1000,
        alpha='auto',
        passes=4)

    cm = gensim.models.CoherenceModel(
        model=lda_model,
        corpus=corpus,
        dictionary=dictionary,
        coherence='u_mass')

    return cm.get_coherence()
```

```
if __name__ == '__main__':

    num_cores = multiprocessing.cpu_count()

    results = Parallel(n_jobs=num_cores)(delayed(try_topic_number)(i)
                                         for i in try_topic_n)

    pickle.dump(results, open('scores.pkl', 'wb'))
```

You can see above I've dumped the coherence scores into a binary `pickle` file. A `pickle` is simply any Python object that has been saved to a binary file. We can `load` these in too:

```
In [28]: try_topic_n = list(range(5,200,2))
         scores = pickle.load(open('scripts/scores.pkl', 'rb'))
         list(zip(try_topic_n, scores))
```

```
Out[28]: [(5, -1.0719929991318369),
          (7, -0.96332378833297394),
          (9, -1.0370206116730427),
          (11, -0.9391420508605669),
          (13, -0.89941919629993672),
          (15, -0.97948998912659335),
          (17, -0.91916766531351723),
          (19, -0.90498737828676379),
          (21, -0.94107971748945507),
          (23, -0.91727178335847159),
          (25, -0.94560202854227282),
          (27, -0.94721997500437649),
          (29, -0.93943916769500091),
          (31, -0.94078558004436297),
          (33, -0.95568210386623209),
          (35, -0.97168223052159419),
          (37, -0.95621895351631225),
          (39, -0.96524311591964762),
          (41, -0.94611608471546926),
          (43, -0.94697199863616643),
          (45, -0.93911588662731871),
          (47, -0.9498111580639399),
          (49, -0.95470584233581701),
          (51, -0.94337074894135131),
          (53, -0.95659529167559731),
          (55, -0.95964640866303286),
          (57, -0.95353739781735525),
          (59, -0.96554670690867095),
          (61, -0.96413249399173662),
```

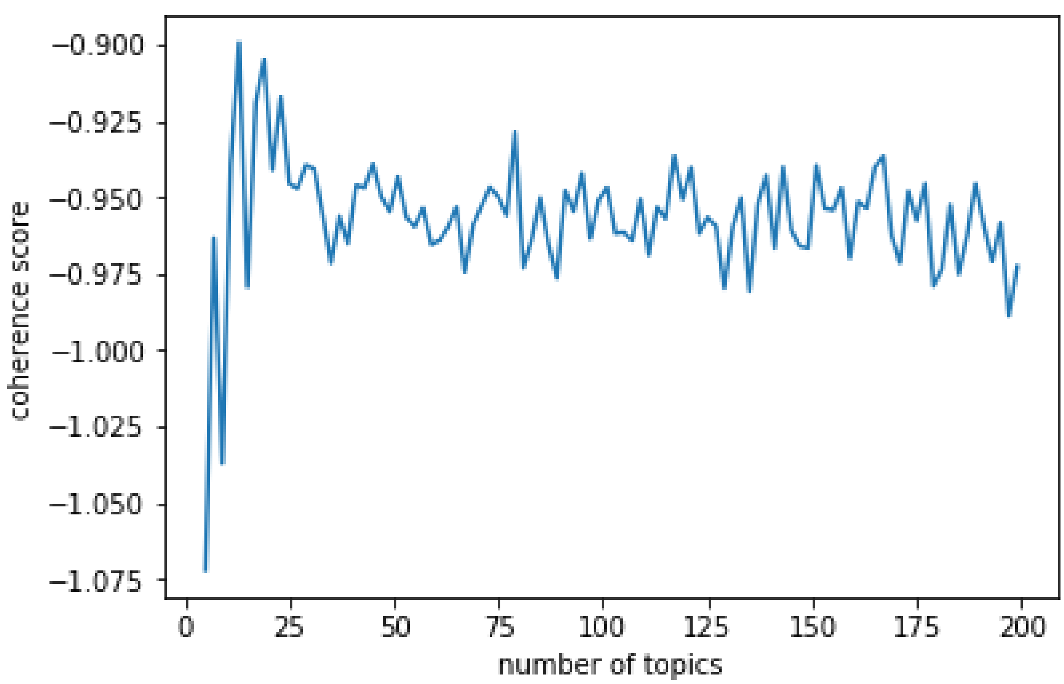
```
(63, -0.9597071349712456),
(65, -0.95322075064367251),
(67, -0.97453921826460621),
(69, -0.95904498451803188),
(71, -0.95284091993987274),
(73, -0.9467063411643819),
(75, -0.94992920286079296),
(77, -0.95610049253624807),
(79, -0.92860299753631115),
(81, -0.97312756908371212),
(83, -0.96377563096911556),
(85, -0.95008465534909625),
(87, -0.96543215191036003),
(89, -0.97662483476026962),
(91, -0.94787857122979113),
(93, -0.95464172951007742),
(95, -0.94219118598190577),
(97, -0.96364966321562884),
(99, -0.9509372259060006),
(101, -0.94685599982300028),
(103, -0.96185709107967166),
(105, -0.9615123885827348),
(107, -0.96423978217584561),
(109, -0.95064836744421155),
(111, -0.96905064541012209),
(113, -0.95325120374870087),
(115, -0.95707205078430169),
(117, -0.93644736640257209),
(119, -0.95073134884943034),
(121, -0.94031496748477417),
(123, -0.9618630838785438),
(125, -0.95664586131528528),
(127, -0.95956694923635066),
(129, -0.97990766834234799),
(131, -0.95982220145807873),
(133, -0.95016330394011927),
(135, -0.98078575773173016),
(137, -0.95221726165387099),
(139, -0.94280735572534335),
(141, -0.96683490205777112),
(143, -0.9400411091460914),
(145, -0.96087720059000503),
(147, -0.96599117549425273),
(149, -0.96675183701243272),
(151, -0.93959412488762073),
(153, -0.95370673198849332),
(155, -0.95425645625470878),
(157, -0.94700901271572846),
```

```
 (159, -0.96994385692302598),
 (161, -0.95150869162452156),
 (163, -0.95369505288969414),
 (165, -0.9401884828661986),
 (167, -0.93656907260895406),
 (169, -0.96267001883782466),
 (171, -0.97164091847540712),
 (173, -0.94802082660578602),
 (175, -0.95767394571051701),
 (177, -0.94555405113902935),
 (179, -0.97919311855470781),
 (181, -0.97356304828972018),
 (183, -0.95241088373517779),
 (185, -0.97524454815721184),
 (187, -0.96263727279370936),
 (189, -0.94559645521049451),
 (191, -0.95909073922754517),
 (193, -0.97108547435530745),
 (195, -0.9584025595723572),
 (197, -0.98878814584420482),
 (199, -0.97270258975592339)]
```

Let's plot these results:

```
In [29]: plt.plot(try_topic_n, [x for x in scores])
         plt.xlabel('number of topics')
         plt.ylabel('coherence score')

Out[29]: <matplotlib.text.Text at 0x7f65e3166400>
```



numpy has a handy argmax or argmin function that returns the index of the highest or lowest value in an array:

```
In [30]: np.argmax(scores)
```

```
Out[30]: 4
```

Then we can just index our topic numbers to get the corresponding number of topics with the highest coherency:

```
In [31]: try_topic_n[np.argmax(scores)]
```

```
Out[31]: 13
```

I've retrained the model for 13 topics and exported as below (note the max_iter=1000 takes a long time, so I've pickled the model again):

```
lda = LatentDirichletAllocation(n_components=13, max_iter=1000)
lda_model = lda.fit(dtm)

pickle.dump((lda, lda_model, dtm, cv), open('13-topics.pkl', 'wb'))
```

We can load in the pre-trained model from the pickle:

```
In [32]: lda, lda_model, dtm, cv = pickle.load(open('scripts/13-topics.pkl', 'rb'))
```

Many papers in the social sciences still don't use a quantitative evaluation metric. Many use the library pyLDAvis to simply visualize the topic distributions, looking for the right size and little overlap in topics as markers of a well-chosen number of topics:

```
In [33]: pyLDAvis.enable_notebook()
         pyLDAvis.sklearn.prepare(lda_model, dtm, cv)
```

```
Out[33]: PreparedData(topic_coordinates=            Freq  cluster  topics         x         y
         topic
         12     13.440292        1       1 -0.023313 -0.104729
         0      12.738078        1       2  0.052580 -0.090617
         1      11.921893        1       3 -0.212336 -0.018731
         8      11.692624        1       4 -0.070277 -0.041536
         6      11.452392        1       5  0.149730  0.007720
         5      10.549177        1       6  0.146266 -0.071886
         3       7.119147        1       7 -0.022566  0.060951
         10      7.064748        1       8  0.008804  0.106346
         7       6.132170        1       9  0.093365  0.079750
```

```
11      4.225619        1      10  0.042795 -0.103414
9       2.853102        1      11 -0.255664  0.049772
2       0.810072        1      12  0.104300  0.141190
4       0.000687        1      13 -0.013683 -0.014816, topic_info=     Category
term
4525  Default  3208.000000       thou  3208.000000  30.0000  30.0000
4772  Default  5790.000000         ve  5790.000000  29.0000  29.0000
4512  Default  2486.000000       thee  2486.000000  28.0000  28.0000
4545  Default  2576.000000        thy  2576.000000  27.0000  27.0000
1272  Default  2667.000000       didn  2667.000000  26.0000  26.0000
323   Default  2289.000000       aunt  2289.000000  25.0000  25.0000
1367  Default  1624.000000     doctor  1624.000000  24.0000  24.0000
4989  Default  2145.000000         ye  2145.000000  23.0000  23.0000
2200  Default  2475.000000     honour  2475.000000  22.0000  22.0000
500   Default  1132.000000       boat  1132.000000  21.0000  21.0000
122   Default  1166.000000        ain  1166.000000  20.0000  20.0000
649   Default  1234.000000     castle  1234.000000  19.0000  19.0000
184   Default  1384.000000  apartment  1384.000000  18.0000  18.0000
4036  Default   784.000000       ship   784.000000  17.0000  17.0000
2480  Default  1605.000000        isn  1605.000000  16.0000  16.0000
4970  Default  1595.000000     wouldn  1595.000000  15.0000  15.0000
624   Default   739.000000    captain   739.000000  14.0000  14.0000
2729  Default  1016.000000      mamma  1016.000000  13.0000  13.0000
1483  Default  1375.000000         em  1375.000000  12.0000  12.0000
4959  Default  1044.000000      woods  1044.000000  11.0000  11.0000
2714  Default   473.000000     maiden   473.000000  10.0000  10.0000
4241  Default   928.000000     squire   928.000000   9.0000   9.0000
1008  Default  1310.000000     couldn  1310.000000   8.0000   8.0000
2000  Default   407.000000       gods   407.000000   7.0000   7.0000
2703  Default  1403.000000         ma  1403.000000   6.0000   6.0000
1372  Default  1007.000000      doesn  1007.000000   5.0000   5.0000
1363  Default   554.000000     divine   554.000000   4.0000   4.0000
2391  Default   887.000000   inquired   887.000000   3.0000   3.0000
1376  Default  1010.000000    dollars  1010.000000   2.0000   2.0000
3644  Default   485.000000   rejoined   485.000000   1.0000   1.0000
...       ...          ...        ...          ...      ...      ...
4772  Topic13     0.001846         ve  5790.217849  -3.0708  -8.5172
2200  Topic13     0.001846     honour  2475.682550  -2.2211  -8.5172
1272  Topic13     0.001846       didn  2667.620084  -2.2958  -8.5172
4467  Topic13     0.001846        tea  2020.804523  -2.0181  -8.5172
323   Topic13     0.001846       aunt  2289.307547  -2.1429  -8.5172
4512  Topic13     0.001846       thee  2486.416197  -2.2255  -8.5172
4525  Topic13     0.001846       thou  3208.143474  -2.4803  -8.5172
1023  Topic13     0.001846     cousin  1757.788287  -1.8787  -8.5172
4545  Topic13     0.001846        thy  2576.348038  -2.2610  -8.5172
4989  Topic13     0.001846         ye  2145.519413  -2.0780  -8.5172
530   Topic13     0.001846       boys  1158.019732  -1.4613  -8.5172
2480  Topic13     0.001846        isn  1605.219104  -1.7879  -8.5172
```

```
3369  Topic13     0.001846  presently  1319.488017  -1.5919  -8.5172
4103  Topic13     0.001846    sisters  1121.791056  -1.4296  -8.5172
2729  Topic13     0.001846      mamma  1016.742696  -1.3312  -8.5172
184   Topic13     0.001846  apartment  1384.654910  -1.6401  -8.5172
1367  Topic13     0.001846     doctor  1624.681350  -1.7999  -8.5172
1183  Topic13     0.001846   demanded   923.831880  -1.2354  -8.5172
2391  Topic13     0.001846   inquired   887.185623  -1.1949  -8.5172
1376  Topic13     0.001846    dollars  1010.168624  -1.3247  -8.5172
2557  Topic13     0.001846        lad   989.688859  -1.3043  -8.5172
2044  Topic13     0.001846       grey  1337.961183  -1.6058  -8.5172
2922  Topic13     0.001846     nation   556.819552  -0.7291  -8.5172
3123  Topic13     0.001846    parents   784.879615  -1.0724  -8.5172
4959  Topic13     0.001846      woods  1044.582998  -1.3582  -8.5172
349   Topic13     0.001846       baby   892.457545  -1.2009  -8.5172
649   Topic13     0.001846     castle  1234.524067  -1.5253  -8.5172
2039  Topic13     0.001846    greatly  1105.064192  -1.4145  -8.5172
2677  Topic13     0.001846       lord  1015.335235  -1.3298  -8.5172
1008  Topic13     0.001846     couldn  1310.029937  -1.5847  -8.5172

[760 rows x 6 columns], token_table=      Topic      Freq        Term
term
2          4  0.097857       abbey
2          7  0.122322       abbey
2          9  0.200608       abbey
2         10  0.577359       abbey
3          1  0.021863  abhorrence
3          2  0.051014  abhorrence
3          5  0.648612  abhorrence
3          6  0.174907  abhorrence
3          8  0.029151  abhorrence
3          9  0.065590  abhorrence
22         1  0.044714     accents
22         2  0.069104     accents
22         4  0.024390     accents
22         5  0.560960     accents
22         6  0.109753     accents
22         7  0.012195     accents
22         9  0.158532     accents
22        12  0.024390     accents
41         2  0.024016    accursed
41         4  0.036024    accursed
41         5  0.156103    accursed
41         8  0.654431    accursed
41         9  0.030020    accursed
41        11  0.084055    accursed
41        12  0.012008    accursed
51         1  0.014241    acquaint
51         5  0.042724    acquaint
```

```
51          6  0.811757    acquaint
51          8  0.014241    acquaint
51          9  0.106810    acquaint
...       ...       ...         ...
4970        4  0.057662      wouldn
4970       11  0.018176      wouldn
4974        2  0.235153      wreath
4974        3  0.036177      wreath
4974        4  0.397951      wreath
4974        5  0.045222      wreath
4974        8  0.072355      wreath
4974        9  0.036177      wreath
4974       11  0.054266      wreath
4974       12  0.117576      wreath
4987        1  0.681421       yacht
4987        6  0.161837       yacht
4987        8  0.144802       yacht
4989        1  0.002797          ye
4989        2  0.005593          ye
4989        3  0.319736          ye
4989        4  0.066651          ye
4989        5  0.007923          ye
4989        6  0.022838          ye
4989        7  0.316940          ye
4989        8  0.217663          ye
4989        9  0.018644          ye
4989       11  0.015847          ye
4989       12  0.005593          ye
4991        3  0.966200         yer
4991        4  0.011367         yer
4991        7  0.018187         yer
4995        3  0.012694          yo
4995        4  0.962661          yo
4995       11  0.023273          yo

[4675 rows x 3 columns], R=30, lambda_step=0.01, plot_opts={'xlab': 'PC1', 'ylab': 'P
```

1.3.4 Topics

To `print` the topics, we can write a function. `display_topics` will print the most probable words to show up in each topic.

```
In [34]: def display_topics(model, feature_names, num_top_words):
             for topic_idx, topic in enumerate(model.components_):
                 print(topic_idx, " ".join([feature_names[i] for i in topic.argsort()[:-num_top
```

Now let's print the top 10 words of the 20 topics for the model we trained, using our `display_topics` function. Have a look through the output and see what topics you can spot:

```
In [35]: display_topics(lda, dtm_feature_names, 10)
```

```
0 aunt mamma cousin sisters tea papa uncle widow pounds everybody
1 ve didn ain em wouldn ye dogs kitchen wheat couldn
2 maiden gods philosopher rejoined divine apartment statue marble inquired thy
3 ye honour king nation army squire government friar lad baron
4 ve honour didn tea aunt thee thou cousin thy ye
5 honour madam uncle ma wholly favour extremely begged behaviour coach
6 thy religion persons anguish apartment beheld principles passions sentiment respecting
7 castle woods mountains scout apartment chateau ma marquis concerning aunt
8 ve tea lad squire th margaret yo grey colour baby
9 ship boat captain doctor deck shore sail whilst vessel island
10 thou thee thy ye hath hast holy minister knight thine
11 laura neighbourhood doctor clerk interview interests boat evidence honour inquired
12 ve didn isn wouldn doesn couldn social haven dollars wasn
```

We can print which topic each novel is closest to by indexing the topic probabilities and using the argmax function:

```
In [36]: doc_topic = lda.transform(dtm)

         for n in range(doc_topic.shape[0]):
             topic_most_pr = doc_topic[n].argmax()
             print(metadata_tb['author'][n], metadata_tb['title'][n])
             print("doc: {} topic: {}\n".format(n,topic_most_pr))
```

```
Dreiser,Theodore SisterCarrie
doc: 0 topic: 12

Stowe,HarrietBeecher UncleTom'sCabin
doc: 1 topic: 1

Scott,Walter Ivanhoe
doc: 2 topic: 10

Crane,Stephen TheRedBadgeofCourage
doc: 3 topic: 1

Godwin,William CalebWilliams
doc: 4 topic: 6

Hardy,Thomas TessoftheD'Urbervilles
doc: 5 topic: 8

Child,Lydia Philothea
doc: 6 topic: 2

Braddon,Mary TheGoldenCalf
```

```
doc: 7 topic: 0

Alcott,Louisa LittleWomen
doc: 8 topic: 0

Dickens,Charles GreatExpectations
doc: 9 topic: 1

Lawrence,D.H. SonsandLovers
doc: 10 topic: 1

Bronte,Ann TheTenantofWildfellHall
doc: 11 topic: 0

Eliot,George Romola
doc: 12 topic: 10

Aguilar,Grace ValeofCedars
doc: 13 topic: 10

Yonge,Charlotte TheHeirofRedcliffe
doc: 14 topic: 0

Cooper,JamesFenimore ThePioneers
doc: 15 topic: 3

Trollope,Anthony PhineasFinn
doc: 16 topic: 0

Stevenson,RobertLouis JekyllandHyde
doc: 17 topic: 9

Borrow,George TheBibleinSpain
doc: 18 topic: 3

Wollstonecraft,Mary Maria
doc: 19 topic: 6

Norris,Frank ThePit
doc: 20 topic: 1

Craik,Dinah JohnHalifax
doc: 21 topic: 8

Austen,Jane SenseandSensibility
doc: 22 topic: 5

Radcliffe,Ann TheMysteriesofUdolpho
```

```
doc: 23 topic: 7

Sinclair,Upton TheJungle
doc: 24 topic: 1

Edgeworth,Maria Leonora
doc: 25 topic: 5

Poe,EdgarAllen TheNarrativeofArthurGordonPym
doc: 26 topic: 9

Wilde,Oscar ThePictureofDorianGray
doc: 27 topic: 12

Ward,Mrs.Humphry RobertElsmere
doc: 28 topic: 8

Austen,Jane MansfieldPark
doc: 29 topic: 0

Cahan,Abraham TheRiseofDavidLevinsky
doc: 30 topic: 12

James,Henry PortraitofaLady
doc: 31 topic: 12

Scott,Walter Waverley
doc: 32 topic: 3

Stein,Gertrude ThreeLives
doc: 33 topic: 1

Cather,Willa Alexander'sBridge
doc: 34 topic: 1

Collins,Wilkie Basil
doc: 35 topic: 11

Edgeworth,Maria CastleRackrent
doc: 36 topic: 3

Sewell,Anna BlackBeauty
doc: 37 topic: 1

Kingsley,Charles Hypatia
doc: 38 topic: 10

Trollope,FrancesEleanor ThatUnfortunateMarriage
```

```
doc: 39 topic: 0

Conrad,Joseph HeartofDarkness
doc: 40 topic: 1

Burney,Fanny Evelina
doc: 41 topic: 5

DosPassos,John ThreeSoldiers
doc: 42 topic: 1

Martineau,Harriet Deerbrook
doc: 43 topic: 0

Lewis,Matthew TheMonk
doc: 44 topic: 6

Craik,Dinah Agatha'sHusband
doc: 45 topic: 8

Norris,Frank TheOctopus
doc: 46 topic: 1

Fitzgerald,FScott TheBeautifulandtheDamned
doc: 47 topic: 12

James,Henry TheAmbassadors
doc: 48 topic: 12

Wells,H.G. TheTimeMachine
doc: 49 topic: 12

Collins,Wilkie TheWomaninWhite
doc: 50 topic: 11

Ford,FordMadox TheGoodSoldier
doc: 51 topic: 12

Yonge,Charlotte Henrietta'sWish
doc: 52 topic: 0

Hardy,Thomas FarFromtheMaddingCrowd
doc: 53 topic: 8

Orczy,Emma TheScarletPimpernel
doc: 54 topic: 8

Dreiser,Theodore TheFinancier
```

doc: 55 topic: 12

Kipling,Rudyard TheJungleBook
doc: 56 topic: 1

Woolf,Virginia Mrs.Dalloway
doc: 57 topic: 12

Thackeray,William VanityFair
doc: 58 topic: 0

Woolf,Virginia TotheLighthouse
doc: 59 topic: 8

Stevenson,RobertLouis TreasureIsland
doc: 60 topic: 9

Peacock,ThomasLove NightmareAbbey
doc: 61 topic: 6

Gissing,George TheOddWomen
doc: 62 topic: 12

Doyle,ArthurConan TheMysteryoftheCloomber
doc: 63 topic: 9

Gaskell,Elizabeth NorthandSouth
doc: 64 topic: 8

Barr,Amelia JanVeeder'sWife
doc: 65 topic: 10

Bellamy,Edward LookingBackward
doc: 66 topic: 12

London,Jack WhiteFang
doc: 67 topic: 1

Bronte,Charlotte JaneEyre
doc: 68 topic: 8

Hays,Mary EmmaCourtney
doc: 69 topic: 6

Chopin,Kate TheAwakening
doc: 70 topic: 12

Broughton,Rhoda Alas!

doc: 71 topic: 8

Opie,Amelia AdelineMowbray
doc: 72 topic: 5

Brown,CharlesBrockden Wieland
doc: 73 topic: 6

Gaskell,Elizabeth MaryBarton
doc: 74 topic: 8

Burney,Fanny Cecilia
doc: 75 topic: 5

Mansfield,Katherine TheAloe
doc: 76 topic: 1

Doyle,ArthurConan TheDoingsofRafflesHaw
doc: 77 topic: 12

Radcliffe,Ann ASicilianRomance
doc: 78 topic: 7

Dickens,Charles ATaleofTwoCities
doc: 79 topic: 11

Stoker,Bram Dracula
doc: 80 topic: 9

Chopin,Kate AtFault
doc: 81 topic: 1

Burney,Fanny Camilla
doc: 82 topic: 5

Grand,Sarah TheHeavenlyTwins
doc: 83 topic: 12

London,Jack TheCalloftheWild
doc: 84 topic: 1

Freeman,MaryWilkins Pembroke
doc: 85 topic: 1

Forster,E.M. ARoomWithaView
doc: 86 topic: 12

Yonge,Charlotte Abbeychurch

doc: 87 topic: 0

Bonhote,Elizabeth BungayCastle
doc: 88 topic: 5

Crockett,SR TheRedAxe
doc: 89 topic: 10

Wharton,Edith TheHouseofMirth
doc: 90 topic: 12

Disraeli,Benjamin Venetia
doc: 91 topic: 11

Woolf,Virginia Orlando
doc: 92 topic: 8

Smollett,Tobias TheExpedictionofHenryClinker
doc: 93 topic: 3

Barr,Amelia TheMaidofMaidenLane
doc: 94 topic: 10

Webb,Mary GonetoEart
doc: 95 topic: 8

Trollope,FrancesMilton TheVicarofWrexham
doc: 96 topic: 0

Beckford,William Vathek
doc: 97 topic: 7

Forster,E.M. HowardsEnd
doc: 98 topic: 12

Brown,CharlesBrockden ArthurMervyn
doc: 99 topic: 6

Cooper,JameFenimore TheLastoftheMohicans
doc: 100 topic: 7

Wollstonecraft,Mary Mary
doc: 101 topic: 6

Kingsley,Charles AltonLocke
doc: 102 topic: 10

Wharton,Edith TheAgeofInnocence

doc: 103 topic: 12

Hope,Anthony ThePrisonerofZenda
doc: 104 topic: 10

Mackenzie,Henry TheManofFeeling
doc: 105 topic: 3

Galt,John AnnalsoftheParish
doc: 106 topic: 3

Peacock,ThomasLove HeadlongHall
doc: 107 topic: 3

Hawthorne,Nathaniel TheScarletLetter
doc: 108 topic: 10

Carroll,Lewis Alice'sAdventureinWonderland
doc: 109 topic: 1

Peacock,ThomasLove MaidMarian
doc: 110 topic: 3

More,Hannah CoelebsinSearchofaWife
doc: 111 topic: 6

Braddon,Mary HenryDunbar
doc: 112 topic: 12

Shelley,Mary Frankenstein
doc: 113 topic: 6

Bennett,Arnold GrandBabylonHotel
doc: 114 topic: 12

Hawthorne,Nathaniel TheHouseoftheSevenGables
doc: 115 topic: 6

Wharton,Edith EthanFrome
doc: 116 topic: 1

Trollope,FrancesMilton TheWidowBarnaby
doc: 117 topic: 0

Shelley,Mary Mathilda
doc: 118 topic: 6

Lewis,Sinclair TheInnocents

```
doc: 119 topic: 1
```

To get the probabilities for each topic for a given book we can print the whole probability list for a given novel:

```
In [37]: metadata_tb['author'][25], metadata_tb['title'][25], doc_topic[25]

Out[37]: ('Edgeworth,Maria',
          'Leonora',
          array([  1.63855409e-01,   1.19987718e-05,   7.96414969e-03,
                   1.14255336e-02,   1.19986082e-05,   3.95844424e-01,
                   3.90338234e-01,   1.26144617e-02,   1.19988124e-05,
                   1.19988548e-05,   1.36693857e-02,   1.19988142e-05,
                   4.22840900e-03]))
```

1.3.5 Challenge

Add these topic assignments back to our `Table` `metadata_tb`

```
In [38]: # YOUR CODE HERE
```

1.4 Interpreting the Model

There are many strategies that can be used to interpret the output of a topic model. In this case, we will look for any correlations between the topic distributions and metadata.

We'll first grab all the topic distributions similar to what we did above. Remember, the order of the novels is still the same!

```
In [39]: list_of_doctopics = [doc_topic[n] for n in range(len(doc_topic))]
         list_of_doctopics[0]

Out[39]: array([  3.74263196e-03,   2.41587636e-01,   5.86537101e-06,
                  5.86538222e-06,   5.86527464e-06,   3.66496255e-02,
                  2.68368745e-02,   5.86538173e-06,   5.86538193e-06,
                  5.86538412e-06,   8.79970468e-04,   5.86538090e-06,
                  6.90262204e-01])
```

We'll make a `DataFrame`, which is similar to a `Table`, with the probabilities for the topics (columns) and documents (rows):

```
In [40]: df = pd.DataFrame(list_of_doctopics)
         df.head()

Out[40]:          0         1         2         3         4         5         6  \
         0  0.003743  0.241588  0.000006  0.000006  0.000006  0.036650  0.026837
         1  0.161628  0.530295  0.000004  0.029890  0.000004  0.000004  0.103822
         2  0.000003  0.000003  0.000003  0.181354  0.000003  0.035976  0.031576
         3  0.000014  0.719892  0.000014  0.021756  0.000014  0.000014  0.052180
```

```
4  0.000005  0.000005  0.000005  0.104593  0.000005  0.167195  0.726078

          7         8         9        10        11        12
0  0.000006  0.000006  0.000006  0.000880  0.000006  0.690262
1  0.000004  0.026528  0.004156  0.130962  0.000004  0.012698
2  0.061129  0.000003  0.000003  0.689937  0.000003  0.000003
3  0.152988  0.049613  0.000014  0.000014  0.000014  0.003472
4  0.000005  0.000005  0.002090  0.000005  0.000005  0.000005
```

We can add these columns to our `metadata_tb` Table:

```
In [41]: meta = metadata_tb.to_df()
         meta[df.columns] = df
         meta.head()
```

```
Out[41]:                                             filename   id language  date  \
         0    EN_1900_Dreiser,Theodore_SisterCarrie_Novel.txt  265  English  1900
         1  EN_1853_Stowe,HarrietBeecher_UncleTom'sCabin_N...  213  English  1853
         2             EN_1820_Scott,Walter_Ivanhoe_Novel.txt  184  English  1820
         3  EN_1895_Crane,Stephen_TheRedBadgeofCourage_Nov...  258  English  1895
         4     EN_1794_Godwin,William_CalebWilliams_Novel.txt  158  English  1794

                         author                title  gender person  length  \
         0     Dreiser,Theodore         SisterCarrie    male  third  156048
         1  Stowe,HarrietBeecher      UncleTom'sCabin  female  third  180498
         2          Scott,Walter              Ivanhoe    male  third  175069
         3        Crane,Stephen  TheRedBadgeofCourage    male  third   46049
         4       Godwin,William        CalebWilliams    male  first  143832

                   0    ...          3         4         5         6         7  \
         0  0.003743    ...   0.000006  0.000006  0.036650  0.026837  0.000006
         1  0.161628    ...   0.029890  0.000004  0.000004  0.103822  0.000004
         2  0.000003    ...   0.181354  0.000003  0.035976  0.031576  0.061129
         3  0.000014    ...   0.021756  0.000014  0.000014  0.052180  0.152988
         4  0.000005    ...   0.104593  0.000005  0.167195  0.726078  0.000005

                   8         9        10        11        12
         0  0.000006  0.000006  0.000880  0.000006  0.690262
         1  0.026528  0.004156  0.130962  0.000004  0.012698
         2  0.000003  0.000003  0.689937  0.000003  0.000003
         3  0.049613  0.000014  0.000014  0.000014  0.003472
         4  0.000005  0.002090  0.000005  0.000005  0.000005

         [5 rows x 22 columns]
```

The `corr()` method will give us a correlation matrix:

```
In [42]: meta.corr()
```

Out[42]:

```
              id      date    length         0         1         2         3  \
id      1.000000  0.989293 -0.160204 -0.152434  0.572221 -0.078462 -0.291158
date    0.989293  1.000000 -0.131386 -0.109032  0.538569 -0.060252 -0.304192
length -0.160204 -0.131386  1.000000  0.263755 -0.244553 -0.062703 -0.042924
0      -0.152434 -0.109032  0.263755  1.000000 -0.181405 -0.057494 -0.068135
1       0.572221  0.538569 -0.244553 -0.181405  1.000000 -0.068212 -0.180374
2      -0.078462 -0.060252 -0.062703 -0.057494 -0.068212  1.000000 -0.039680
3      -0.291158 -0.304192 -0.042924 -0.068135 -0.180374 -0.039680  1.000000
4       0.177250  0.155241 -0.743355 -0.131120  0.329774 -0.025743  0.047297
5      -0.508749 -0.555964  0.275029  0.053067 -0.259983 -0.043849 -0.036884
6      -0.516003 -0.532168 -0.164554 -0.132030 -0.295489 -0.048615 -0.078284
7      -0.302783 -0.309598  0.018215 -0.188447 -0.133439 -0.034832  0.068400
8       0.283322  0.308496  0.062639 -0.035257 -0.052223 -0.066170 -0.165410
9       0.182507  0.205571 -0.207433 -0.189187 -0.018205 -0.038329 -0.047984
10     -0.108731 -0.081166 -0.009373 -0.135976 -0.153772 -0.033162  0.035017
11      0.009553  0.039784  0.144859 -0.050185 -0.122199 -0.013116 -0.061852
12      0.546103  0.536713  0.021067 -0.134841 -0.025223 -0.064700 -0.171921

               4         5         6         7         8         9        10  \
id      0.177250 -0.508749 -0.516003 -0.302783  0.283322  0.182507 -0.108731
date    0.155241 -0.555964 -0.532168 -0.309598  0.308496  0.205571 -0.081166
length -0.743355  0.275029 -0.164554  0.018215  0.062639 -0.207433 -0.009373
0      -0.131120  0.053067 -0.132030 -0.188447 -0.035257 -0.189187 -0.135976
1       0.329774 -0.259983 -0.295489 -0.133439 -0.052223 -0.018205 -0.153772
2      -0.025743 -0.043849 -0.048615 -0.034832 -0.066170 -0.038329 -0.033162
3       0.047297 -0.036884 -0.078284  0.068400 -0.165410 -0.047984  0.035017
4       1.000000 -0.168267 -0.008754 -0.075140 -0.047680  0.170442 -0.104526
5      -0.168267  1.000000  0.108961 -0.028937 -0.199106 -0.151135 -0.130868
6      -0.008754  0.108961  1.000000  0.051374 -0.223339 -0.108129 -0.034809
7      -0.075140 -0.028937  0.051374  1.000000 -0.160707 -0.021450  0.003769
8      -0.047680 -0.199106 -0.223339 -0.160707  1.000000 -0.051862 -0.084348
9       0.170442 -0.151135 -0.108129 -0.021450 -0.051862  1.000000 -0.019541
10     -0.104526 -0.130868 -0.034809  0.003769 -0.084348 -0.019541  1.000000
11     -0.079231 -0.103558 -0.058271 -0.066057 -0.057570  0.056438 -0.083380
12     -0.027975 -0.211949 -0.243017 -0.182510 -0.020239 -0.044767 -0.187641

              11        12
id      0.009553  0.546103
date    0.039784  0.536713
length  0.144859  0.021067
0      -0.050185 -0.134841
1      -0.122199 -0.025223
2      -0.013116 -0.064700
3      -0.061852 -0.171921
4      -0.079231 -0.027975
5      -0.103558 -0.211949
6      -0.058271 -0.243017
7      -0.066057 -0.182510
```

```
8      -0.057570 -0.020239
9       0.056438 -0.044767
10     -0.083380 -0.187641
11      1.000000 -0.011141
12     -0.011141  1.000000
```

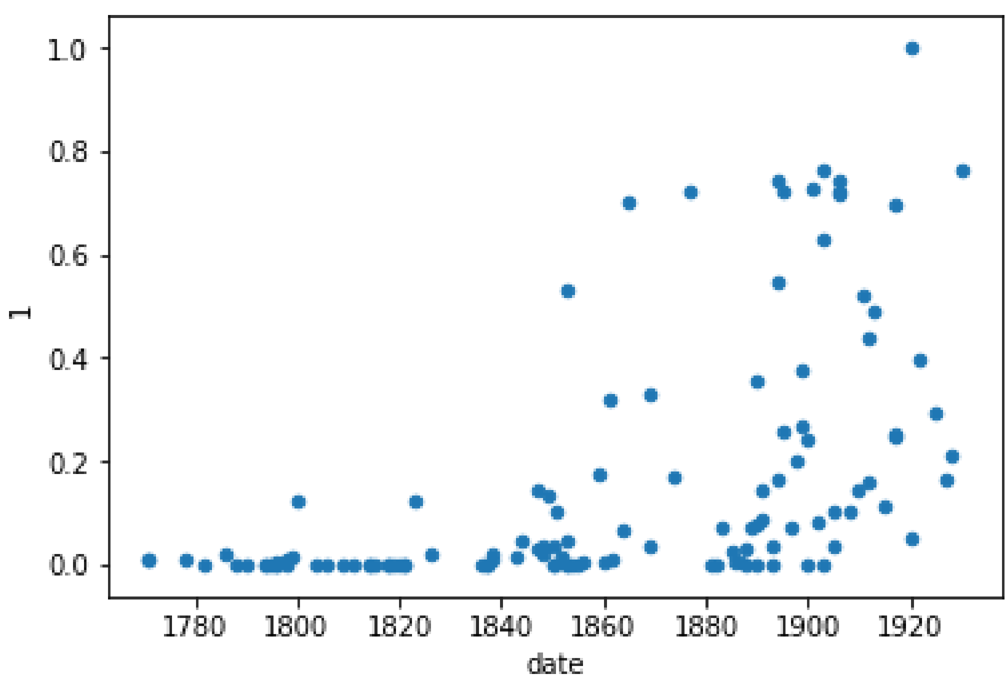
We see some strong correlations of topics with `date`, recall:

```
In [43]: display_topics(lda, dtm_feature_names, 10)
```

```
0 aunt mamma cousin sisters tea papa uncle widow pounds everybody
1 ve didn ain em wouldn ye dogs kitchen wheat couldn
2 maiden gods philosopher rejoined divine apartment statue marble inquired thy
3 ye honour king nation army squire government friar lad baron
4 ve honour didn tea aunt thee thou cousin thy ye
5 honour madam uncle ma wholly favour extremely begged behaviour coach
6 thy religion persons anguish apartment beheld principles passions sentiment respecting
7 castle woods mountains scout apartment chateau ma marquis concerning aunt
8 ve tea lad squire th margaret yo grey colour baby
9 ship boat captain doctor deck shore sail whilst vessel island
10 thou thee thy ye hath hast holy minister knight thine
11 laura neighbourhood doctor clerk interview interests boat evidence honour inquired
12 ve didn isn wouldn doesn couldn social haven dollars wasn
```

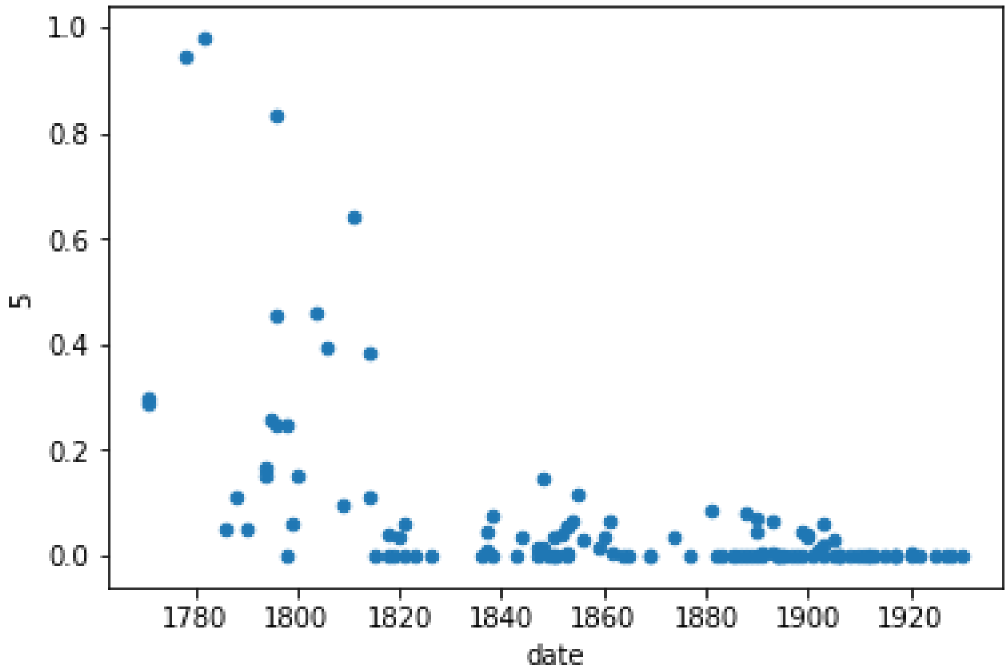
```
In [44]: meta.plot.scatter(x='date', y=1)
```

```
Out[44]: <matplotlib.axes._subplots.AxesSubplot at 0x7f65e3238898>
```



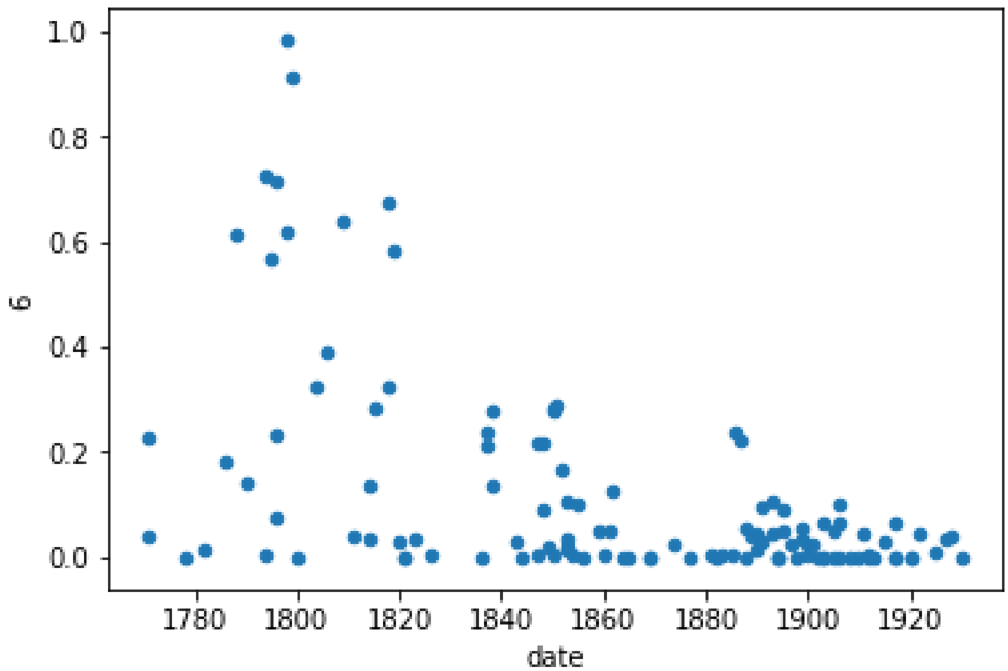
```
In [45]: meta.plot.scatter(x='date', y=5)
```

```
Out[45]: <matplotlib.axes._subplots.AxesSubplot at 0x7f65e3499a58>
```




```
In [46]: meta.plot.scatter(x='date', y=6)
```

```
Out[46]: <matplotlib.axes._subplots.AxesSubplot at 0x7f65e33f7f60>
```




```
In [ ]: meta.plot.scatter(x='date', y=12)
```

Why do you think we see this?

2 Homework

We're going to download the 20 Newsgroups, a widely used corpus for demos of general texts:

> The 20 Newsgroups data set is a collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups. To the best of my knowledge, it was originally collected by Ken Lang, probably for his Newsweeder: Learning to filter netnews paper, though he does not explicitly mention this collection. The 20 newsgroups collection has become a popular data set for experiments in text applications of machine learning techniques, such as text classification and text clustering.

Let's read in the training data:

```
In [11]: train_subset = pickle.load(open('scripts/20-news-train.pkl', 'rb'))
```

Here are the predetermined catgories:

```
In [10]: train_subset.target_names
```

```
Out[10]: ['alt.atheism',
          'comp.graphics',
          'comp.os.ms-windows.misc',
          'comp.sys.ibm.pc.hardware',
          'comp.sys.mac.hardware',
          'comp.windows.x',
          'misc.forsale',
          'rec.autos',
          'rec.motorcycles',
          'rec.sport.baseball',
          'rec.sport.hockey',
          'sci.crypt',
          'sci.electronics',
          'sci.med',
          'sci.space',
          'soc.religion.christian',
          'talk.politics.guns',
          'talk.politics.mideast',
          'talk.politics.misc',
          'talk.religion.misc']
```

Since we're topic modeling, we don't care about what they've been labeled, but it'll be interesting to see how our topics line up with these!

How many documents are there?

```
In [ ]: len(train_subset.data)
```

Let's get a list of documents as strings just like we did with the novels, and then we'll randomly shuffle them in case they're ordered by category already:

```
In [12]: documents_train = train_subset.data
         np.random.shuffle(documents_train)
```

```
In [13]: print(documents_train[0])
```

```
From: nahess@mir.gatech.edu (Nicholas A. Hess)
Subject: Hitatchi Raster Format (HRF)?
Organization: USGS Center for Spatial Analysis Technologies
Lines: 19
NNTP-Posting-Host: mir.gatech.edu
Keywords: HRF

 Our shop uses a package called CADCore - very good - to scan and
subsequently vectorize original maps into digital maps. The problem is that
once the raster file is loaded into the CADCore package, a header is added
to the .HRF file which makes it unreadable by the supplied converter. We
would like to be able to ship some of the already-altered raster images for
further use on our workstations. So, here are my questions:
```

```
 (1) What is the Hitachi format? - I need this format so I can recognize
precisely what to strip out. I strongly suspect that it's a compressed
format - if so, then t might not be possible for me to strip out the
offending header.

 (2) Are there any UNIX packages that read and recognize HRF? It would be
really nice to find some sort of "hrftopbm" converter out there. ;)

 I've already searched some of the more well-known ftp sites which contain
graphics formats documentation, with no luck. So, if you know, or knwo
someone who knows - please email! Thanks.
```

Now we'll do the same for the test set:

```
In [14]: test_subset = pickle.load(open('scripts/20-news-test.pkl', 'rb'))
         documents_test = test_subset.data
         np.random.shuffle(documents_test)
         print(documents_test[0])
```

```
From: ellens@bnr.ca (Chris Ellens)
Subject: Re: Monitors - should they be kept on 24 hours a day???
Nntp-Posting-Host: bcarm422
Organization: Bell-Northern Research
Lines: 10

In article <1r6gis$e46@calvin.NYU.EDU>, roy@mchip00.med.nyu.edu (Roy Smith)
wrote:

>         I wouldn't worry too much about wasting electricity in the winter
> months; that energy is just getting turned into heat.  It may not be as
> efficient a way to heat a building as the central heating plant, but it's

Is there any such thing as in inefficient heater?

Chris Ellens         ellens@bnr.ca
```

2.1 TASK:

You now have two arrays of strings: `documents_train` and `documents_test`. Create a `dtm` and then a topic model for `k` number of topics. Just choose one number of `k` and a very low `iter` value for the training so it doesn't take too long.

See how the topics match up to the annotated categories, and play with different ways of preprocessing the data. Use the `pyLDAvis` library to evaluate your model.

What did you have to do to get decent results?

```
In [19]: dtm = fit_transform(documents_train)
```

```
        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-19-2d1034475376> in <module>()
    ----> 1 dtm = fit_transform(documents_train)


        NameError: name 'fit_transform' is not defined
```

3 BONUS (not assigned)

Create a classifier from this corpus. They're assigned group are in the `target` attribute:

```
In [ ]: train_subset.target
```

```
In [ ]: test_subset.target
```