[1]
```
import os
import re
import operator
import matplotlib.pyplot as plt
import warnings
import gensim
import numpy as np
warnings.filterwarnings('ignore')

%matplotlib inline

from gensim.models import CoherenceModel, LdaModel, LsiModel, HdpModel
from gensim.models.wrappers import LdaMallet
from gensim.corpora import Dictionary
from pprint import pprint

from gensim.utils import lemmatize
from nltk.corpus import stopwords

test_data_dir = '{}'.format(os.sep).join([gensim.__path__[0], 'test', 'te
lee_train_file = test_data_dir + os.sep + 'testfile.cor'

def build_texts(fname):
    with open(fname) as f:
        for line in f:
            yield gensim.utils.simple_preprocess(line, deacc=True, min_le

train_texts = list(build_texts(lee_train_file))

bigram = gensim.models.Phrases(train_texts)

stops = set(stopwords.words('english'))

def process_texts(texts):
    texts = [[word for word in line if word not in stops] for line in tex
    texts = [bigram[line] for line in texts]
    texts = [[word.split('/')[0] for word in lemmatize(' '.join(line), al
    return texts

train_texts = process_texts(train_texts)

dictionary = Dictionary(train_texts)
corpus = [dictionary.doc2bow(text) for text in train_texts]

print '\nLSI Model Started'
lsimodel = LsiModel(corpus=corpus, num_topics=5, id2word=dictionary)
print lsimodel.show_topics(num_topics=5)
lsitopics = lsimodel.show_topics(formatted=False)

print 'LSI Model Ended\n\nHDP Model Started'
```

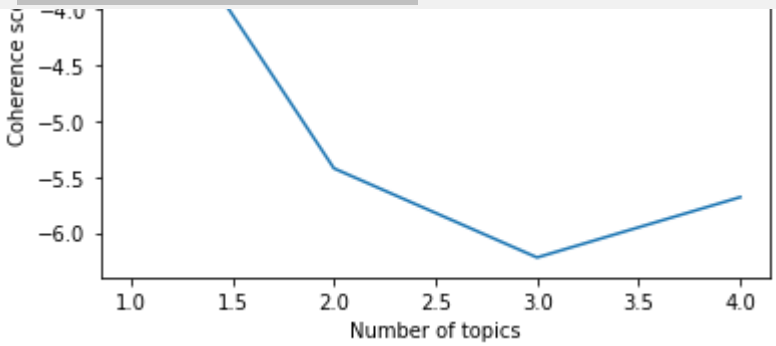
```
    hdpmodel = HdpModel(corpus=corpus, id2word=dictionary)
    print hdpmodel.show_topics()
    hdptopics = hdpmodel.show_topics(formatted=False)

    print 'HDP Model Ended\n\nLDA Model Started'
    ldamodel = LdaModel(corpus=corpus, num_topics=5, id2word=dictionary)
    print ldamodel.show_topics()
    ldatopics = ldamodel.show_topics(formatted=False)
    print 'LDA Model Ended\n'

    print 'Creating graphs...\n'

    def evaluate_graph_general(dictionary, corpus, texts, limit, measure, mod
        coh_measure = []
        lm_list = []
        if model == 'LdaModel':
            print 'LDA Model Detected. Generating graph...'
        elif model == 'HdpModel':
            print 'HDP Model Detected. Generating graph...'
        elif model == 'LsiModel':
            print 'LSI Model Detected. Generating graph...'
        else:
            print 'Invalid Model!'

        for num_topics in range(1, limit):
            if model == 'LdaModel':
                lm = LdaModel(corpus=corpus, num_topics=num_topics, id2word=d
            elif model == 'HdpModel':
                lm = HdpModel(corpus=corpus, id2word=dictionary)
            elif model == 'LsiModel':
                lm = LsiModel(corpus=corpus, num_topics=num_topics, id2word=d
            else:
                print 'Invalid Model!'
            lm_list.append(lm)
            cm = CoherenceModel(model=lm, texts=texts, dictionary=dictionary,
            coh_measure.append(cm.get_coherence())

        # Show graph
        x = range(1, limit)

        if model == 'LdaModel':
            title = 'Dataset 2 (300 Long Articles): LDA'
        elif model == 'HdpModel':
            title = 'Dataset 2 (300 Long Articles): HDP'
        elif model == 'LsiModel':
            title = 'Dataset 2 (300 Long Articles): LSI'
        else:
            title = 'ERROR GRAPH'

        if measure == 'c_v':
            coh_label = 'c_v coherence'
        elif measure == 'c_uci':
```

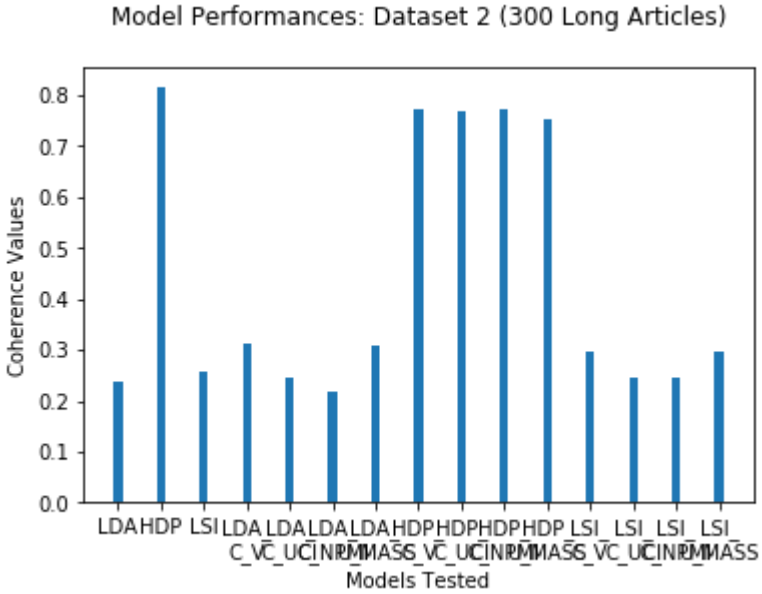
```
            coh_label = 'c_uci coherence'
        elif measure == 'c_npmi':
            coh_label = 'c_npmi coherence'
        elif measure == 'u_mass':
            coh_label = 'u_mass coherence'
        else:
            coh_label = 'ERROR LABEL'

        plt.plot(x, coh_measure, label=coh_label)
        plt.title(title + ' ' + coh_label + '\n')
        plt.xlabel("Number of topics")
        plt.ylabel("Coherence score")
        plt.legend(loc='best')
        plt.show()

        return lm_list, coh_measure

    # ----------------------------------LDA-----------------------------------
    # ----------------------------------c_v-----------------------------------
    ldalist_c_v, lda_c_v = evaluate_graph_general(dictionary=dictionary,
                                                  corpus=corpus,
                                                  texts=train_texts,
                                                  limit=5,
                                                  measure='c_v',
                                                  model='LdaModel')
    ldatopics_c_v = ldalist_c_v[2].show_topics(formatted=False)
    # ----------------------------------c_uci---------------------------------
    ldalist_c_uci, lda_c_uci = evaluate_graph_general(dictionary=dictionary,
                                                  corpus=corpus,
                                                  texts=train_texts,
                                                  limit=5,
                                                  measure='c_uci',
                                                  model='LdaModel')
    ldatopics_c_uci = ldalist_c_uci[2].show_topics(formatted=False)
    # ----------------------------------c_npmi--------------------------------
    ldalist_c_npmi, lda_c_npmi = evaluate_graph_general(dictionary=dictionary
                                                  corpus=corpus,
                                                  texts=train_texts,
                                                  limit=5,
                                                  measure='c_npmi',
                                                  model='LdaModel')
    ldatopics_c_npmi = ldalist_c_npmi[2].show_topics(formatted=False)
    # ----------------------------------u_mass--------------------------------
    ldalist_u_mass, lda_u_mass = evaluate_graph_general(dictionary=dictionary
                                                  corpus=corpus,
                                                  texts=train_texts,
                                                  limit=5,
                                                  measure='u_mass',
                                                  model='LdaModel')
    ldatopics_u_mass = ldalist_u_mass[2].show_topics(formatted=False)
    # ----------------------------------LDA-----------------------------------
```

```
# ----------------------------------HDP-----------------------------------
# ----------------------------------c_v-----------------------------------
hdplist_c_v, hdp_c_v = evaluate_graph_general(dictionary=dictionary,
                                              corpus=corpus,
                                              texts=train_texts,
                                              limit=5,
                                              measure='c_v',
                                              model='HdpModel')
hdptopics_c_v = hdplist_c_v[2].show_topics(formatted=False)
# ----------------------------------c_uci---------------------------------
hdplist_c_uci, hdp_c_uci = evaluate_graph_general(dictionary=dictionary,
                                              corpus=corpus,
                                              texts=train_texts,
                                              limit=5,
                                              measure='c_uci',
                                              model='HdpModel')
hdptopics_c_uci = hdplist_c_uci[2].show_topics(formatted=False)
# ----------------------------------c_npmi--------------------------------
hdplist_c_npmi, hdp_c_npmi = evaluate_graph_general(dictionary=dictionary
                                              corpus=corpus,
                                              texts=train_texts,
                                              limit=5,
                                              measure='c_npmi',
                                              model='HdpModel')
hdptopics_c_npmi = hdplist_c_npmi[2].show_topics(formatted=False)
# ----------------------------------u_mass--------------------------------
hdplist_u_mass, hdp_u_mass = evaluate_graph_general(dictionary=dictionary
                                              corpus=corpus,
                                              texts=train_texts,
                                              limit=5,
                                              measure='u_mass',
                                              model='HdpModel')
hdptopics_u_mass = hdplist_u_mass[2].show_topics(formatted=False)
# ----------------------------------HDP-----------------------------------

# ----------------------------------LSI-----------------------------------
# ----------------------------------c_v-----------------------------------
lsilist_c_v, lsi_c_v = evaluate_graph_general(dictionary=dictionary,
                                              corpus=corpus,
                                              texts=train_texts,
                                              limit=5,
                                              measure='c_v',
                                              model='LsiModel')
lsitopics_c_v = lsilist_c_v[2].show_topics(formatted=False)
# ----------------------------------c_uci---------------------------------
lsilist_c_uci, lsi_c_uci = evaluate_graph_general(dictionary=dictionary,
                                              corpus=corpus,
                                              texts=train_texts,
                                              limit=5,
                                              measure='c_uci',
                                              model='LsiModel')
lsitopics_c_uci = lsilist_c_uci[2].show_topics(formatted=False)
```

```
    # ----------------------------------c_npmi-------------------------------
    lsilist_c_npmi, lsi_c_npmi = evaluate_graph_general(dictionary=dictionary
                                                        corpus=corpus,
                                                        texts=train_texts,
                                                        limit=5,
                                                        measure='c_npmi',
                                                        model='LsiModel')
    lsitopics_c_npmi = lsilist_c_npmi[2].show_topics(formatted=False)
    # ----------------------------------u_mass-------------------------------
    lsilist_u_mass, lsi_u_mass = evaluate_graph_general(dictionary=dictionary
                                                        corpus=corpus,
                                                        texts=train_texts,
                                                        limit=5,
                                                        measure='u_mass',
                                                        model='LsiModel')
    lsitopics_u_mass = lsilist_u_mass[2].show_topics(formatted=False)
    # ----------------------------------LSI----------------------------------

    ldatopics = [[word for word, prob in topic] for topicid, topic in ldatopi
    hdptopics = [[word for word, prob in topic] for topicid, topic in hdptopi
    lsitopics = [[word for word, prob in topic] for topicid, topic in lsitopi

    ldatopics_c_v = [[word for word, prob in topic] for topicid, topic in lda
    ldatopics_c_uci = [[word for word, prob in topic] for topicid, topic in l
    ldatopics_c_npmi = [[word for word, prob in topic] for topicid, topic in
    ldatopics_u_mass = [[word for word, prob in topic] for topicid, topic in

    hdptopics_c_v = [[word for word, prob in topic] for topicid, topic in hdp
    hdptopics_c_uci = [[word for word, prob in topic] for topicid, topic in h
    hdptopics_c_npmi = [[word for word, prob in topic] for topicid, topic in
    hdptopics_u_mass = [[word for word, prob in topic] for topicid, topic in

    lsitopics_c_v = [[word for word, prob in topic] for topicid, topic in lsi
    lsitopics_c_uci = [[word for word, prob in topic] for topicid, topic in l
    lsitopics_c_npmi = [[word for word, prob in topic] for topicid, topic in
    lsitopics_u_mass = [[word for word, prob in topic] for topicid, topic in

    lda_coherence = CoherenceModel(topics=ldatopics, texts=train_texts, dicti
    hdp_coherence = CoherenceModel(topics=hdptopics[:2], texts=train_texts, d
    lsi_coherence = CoherenceModel(topics=lsitopics[:2], texts=train_texts, d

    lda_c_v_coherence = CoherenceModel(topics=ldatopics_c_v, texts=train_text
    lda_c_uci_coherence = CoherenceModel(topics=ldatopics_c_uci, texts=train_
    lda_c_npmi_coherence = CoherenceModel(topics=ldatopics_c_npmi, texts=trai
    lda_u_mass_coherence = CoherenceModel(topics=ldatopics_u_mass, texts=trai

    hdp_c_v_coherence = CoherenceModel(topics=hdptopics_c_v, texts=train_text
    hdp_c_uci_coherence = CoherenceModel(topics=hdptopics_c_uci, texts=train_
    hdp_c_npmi_coherence = CoherenceModel(topics=hdptopics_c_npmi, texts=trai
    hdp_u_mass_coherence = CoherenceModel(topics=hdptopics_u_mass, texts=trai

    lsi_c_v_coherence = CoherenceModel(topics=lsitopics_c_v, texts=train_text
```

```
lsi_c_uci_coherence = CoherenceModel(topics=lsitopics_c_uci, texts=train_
lsi_c_npmi_coherence = CoherenceModel(topics=lsitopics_c_npmi, texts=trai
lsi_u_mass_coherence = CoherenceModel(topics=lsitopics_u_mass, texts=trai

def evaluate_bar_graph(coherences, indices):
    assert len(coherences) == len(indices)
    n = len(coherences)
    x = np.arange(n)
    plt.bar(x, coherences, width=0.2, tick_label=indices, align='center')
    plt.title('Model Performances: Dataset 2 (300 Long Articles)\n')
    plt.xlabel('Models Tested')
    plt.ylabel('Coherence Values')

evaluate_bar_graph([lda_coherence, hdp_coherence, lsi_coherence,
                    lda_c_v_coherence, lda_c_uci_coherence, lda_c_npmi_co
                    hdp_c_v_coherence, hdp_c_uci_coherence, hdp_c_npmi_co
                    lsi_c_v_coherence, lsi_c_uci_coherence, lsi_c_npmi_co
                   ['LDA', 'HDP', 'LSI', 'LDA_\nC_V', 'LDA_\nC_UCI', 'LDA
```





[2]

```
import os
import re
import operator
import matplotlib.pyplot as plt
import warnings
import gensim
import numpy as np
warnings.filterwarnings('ignore')

%matplotlib inline

from gensim.models import CoherenceModel, LdaModel, LsiModel, HdpModel
from gensim.models.wrappers import LdaMallet
from gensim.corpora import Dictionary
from pprint import pprint

from gensim.utils import lemmatize
from nltk.corpus import stopwords

test_data_dir = '{}'.format(os.sep).join([gensim.__path__[0], 'test', 'te
lee_train_file = test_data_dir + os.sep + 'testfile.cor'

def build_texts(fname):
    with open(fname) as f:
        for line in f:
            yield gensim.utils.simple_preprocess(line, deacc=True, min_le

train_texts = list(build_texts(lee_train_file))

bigram = gensim.models.Phrases(train_texts)

stops = set(stopwords.words('english'))

def process_texts(texts):
    texts = [[word for word in line if word not in stops] for line in tex
    texts = [bigram[line] for line in texts]
    texts = [[word.split('/')[0] for word in lemmatize(' '.join(line), al
    return texts

train_texts = process_texts(train_texts)

dictionary = Dictionary(train_texts)
corpus = [dictionary.doc2bow(text) for text in train_texts]

print '\nLSI Model Started'
lsimodel = LsiModel(corpus=corpus, num_topics=10, id2word=dictionary)
print lsimodel.show_topics(num_topics=5)
lsitopics = lsimodel.show_topics(formatted=False)

print 'LSI Model Ended\n\nHDP Model Started'
hdpmodel = HdpModel(corpus=corpus, id2word=dictionary)
print hdpmodel.show_topics()
```

```
    hdptopics = hdpmodel.show_topics(formatted=False)

    print 'HDP Model Ended\n\nLDA Model Started'
    ldamodel = LdaModel(corpus=corpus, num_topics=10, id2word=dictionary)
    print ldamodel.show_topics()
    ldatopics = ldamodel.show_topics(formatted=False)
    print 'LDA Model Ended\n'

    print 'Creating graphs...\n'

    def evaluate_graph_general(dictionary, corpus, texts, limit, measure, mod
        coh_measure = []
        lm_list = []
        if model == 'LdaModel':
            print 'LDA Model Detected. Generating graph...'
        elif model == 'HdpModel':
            print 'HDP Model Detected. Generating graph...'
        elif model == 'LsiModel':
            print 'LSI Model Detected. Generating graph...'
        else:
            print 'Invalid Model!'

        for num_topics in range(1, limit):
            if model == 'LdaModel':
                lm = LdaModel(corpus=corpus, num_topics=num_topics, id2word=d
            elif model == 'HdpModel':
                lm = HdpModel(corpus=corpus, id2word=dictionary)
            elif model == 'LsiModel':
                lm = LsiModel(corpus=corpus, num_topics=num_topics, id2word=d
            else:
                print 'Invalid Model!'
            lm_list.append(lm)
            cm = CoherenceModel(model=lm, texts=texts, dictionary=dictionary,
            coh_measure.append(cm.get_coherence())

        # Show graph
        x = range(1, limit)

        if model == 'LdaModel':
            title = 'Dataset 2 (300 Long Articles): LDA'
        elif model == 'HdpModel':
            title = 'Dataset 2 (300 Long Articles): HDP'
        elif model == 'LsiModel':
            title = 'Dataset 2 (300 Long Articles): LSI'
        else:
            title = 'ERROR GRAPH'

        if measure == 'c_v':
            coh_label = 'c_v coherence'
        elif measure == 'c_uci':
            coh_label = 'c_uci coherence'
        elif measure == 'c_npmi':
```

```
            coh_label = 'c_npmi coherence'
        elif measure == 'u_mass':
            coh_label = 'u_mass coherence'
        else:
            coh_label = 'ERROR LABEL'

        plt.plot(x, coh_measure, label=coh_label)
        plt.title(title + ' ' + coh_label + '\n')
        plt.xlabel("Number of topics")
        plt.ylabel("Coherence score")
        plt.legend(loc='best')
        plt.show()

        return lm_list, coh_measure

    # ----------------------------------LDA---------------------------------
    # ----------------------------------c_v---------------------------------
    ldalist_c_v, lda_c_v = evaluate_graph_general(dictionary=dictionary,
                                                  corpus=corpus,
                                                  texts=train_texts,
                                                  limit=11,
                                                  measure='c_v',
                                                  model='LdaModel')
    ldatopics_c_v = ldalist_c_v[5].show_topics(formatted=False)
    # ----------------------------------c_uci---------------------------------
    ldalist_c_uci, lda_c_uci = evaluate_graph_general(dictionary=dictionary,
                                                  corpus=corpus,
                                                  texts=train_texts,
                                                  limit=11,
                                                  measure='c_uci',
                                                  model='LdaModel')
    ldatopics_c_uci = ldalist_c_uci[5].show_topics(formatted=False)
    # ----------------------------------c_npmi---------------------------------
    ldalist_c_npmi, lda_c_npmi = evaluate_graph_general(dictionary=dictionary
                                                  corpus=corpus,
                                                  texts=train_texts,
                                                  limit=11,
                                                  measure='c_npmi',
                                                  model='LdaModel')
    ldatopics_c_npmi = ldalist_c_npmi[5].show_topics(formatted=False)
    # ----------------------------------u_mass---------------------------------
    ldalist_u_mass, lda_u_mass = evaluate_graph_general(dictionary=dictionary
                                                  corpus=corpus,
                                                  texts=train_texts,
                                                  limit=11,
                                                  measure='u_mass',
                                                  model='LdaModel')
    ldatopics_u_mass = ldalist_u_mass[5].show_topics(formatted=False)
    # ----------------------------------LDA---------------------------------

    # ----------------------------------HDP---------------------------------
    # ----------------------------------c_v---------------------------------
```

```
    hdplist_c_v, hdp_c_v = evaluate_graph_general(dictionary=dictionary,
                                                  corpus=corpus,
                                                  texts=train_texts,
                                                  limit=11,
                                                  measure='c_v',
                                                  model='HdpModel')
    hdptopics_c_v = hdplist_c_v[5].show_topics(formatted=False)
    # ---------------------------------c_uci-------------------------------
    hdplist_c_uci, hdp_c_uci = evaluate_graph_general(dictionary=dictionary,
                                                  corpus=corpus,
                                                  texts=train_texts,
                                                  limit=11,
                                                  measure='c_uci',
                                                  model='HdpModel')
    hdptopics_c_uci = hdplist_c_uci[5].show_topics(formatted=False)
    # ---------------------------------c_npmi-------------------------------
    hdplist_c_npmi, hdp_c_npmi = evaluate_graph_general(dictionary=dictionary
                                                  corpus=corpus,
                                                  texts=train_texts,
                                                  limit=11,
                                                  measure='c_npmi',
                                                  model='HdpModel')
    hdptopics_c_npmi = hdplist_c_npmi[5].show_topics(formatted=False)
    # ---------------------------------u_mass-------------------------------
    hdplist_u_mass, hdp_u_mass = evaluate_graph_general(dictionary=dictionary
                                                  corpus=corpus,
                                                  texts=train_texts,
                                                  limit=11,
                                                  measure='u_mass',
                                                  model='HdpModel')
    hdptopics_u_mass = hdplist_u_mass[5].show_topics(formatted=False)
    # ---------------------------------HDP----------------------------------

    # ---------------------------------LSI----------------------------------
    # ---------------------------------c_v----------------------------------
    lsilist_c_v, lsi_c_v = evaluate_graph_general(dictionary=dictionary,
                                                  corpus=corpus,
                                                  texts=train_texts,
                                                  limit=11,
                                                  measure='c_v',
                                                  model='LsiModel')
    lsitopics_c_v = lsilist_c_v[5].show_topics(formatted=False)
    # ---------------------------------c_uci--------------------------------
    lsilist_c_uci, lsi_c_uci = evaluate_graph_general(dictionary=dictionary,
                                                  corpus=corpus,
                                                  texts=train_texts,
                                                  limit=11,
                                                  measure='c_uci',
                                                  model='LsiModel')
    lsitopics_c_uci = lsilist_c_uci[5].show_topics(formatted=False)
    # ---------------------------------c_npmi-------------------------------
    lsilist_c_npmi, lsi_c_npmi = evaluate_graph_general(dictionary=dictionary
```

```
                                                corpus=corpus,
                                                texts=train_texts,
                                                limit=11,
                                                measure='c_npmi',
                                                model='LsiModel')
    lsitopics_c_npmi = lsilist_c_npmi[5].show_topics(formatted=False)
    # ----------------------------------u_mass-------------------------------
    lsilist_u_mass, lsi_u_mass = evaluate_graph_general(dictionary=dictionary
                                                corpus=corpus,
                                                texts=train_texts,
                                                limit=11,
                                                measure='u_mass',
                                                model='LsiModel')
    lsitopics_u_mass = lsilist_u_mass[5].show_topics(formatted=False)
    # ----------------------------------LSI----------------------------------

    ldatopics = [[word for word, prob in topic] for topicid, topic in ldatopi
    hdptopics = [[word for word, prob in topic] for topicid, topic in hdptopi
    lsitopics = [[word for word, prob in topic] for topicid, topic in lsitopi

    ldatopics_c_v = [[word for word, prob in topic] for topicid, topic in lda
    ldatopics_c_uci = [[word for word, prob in topic] for topicid, topic in l
    ldatopics_c_npmi = [[word for word, prob in topic] for topicid, topic in
    ldatopics_u_mass = [[word for word, prob in topic] for topicid, topic in

    hdptopics_c_v = [[word for word, prob in topic] for topicid, topic in hdp
    hdptopics_c_uci = [[word for word, prob in topic] for topicid, topic in h
    hdptopics_c_npmi = [[word for word, prob in topic] for topicid, topic in
    hdptopics_u_mass = [[word for word, prob in topic] for topicid, topic in

    lsitopics_c_v = [[word for word, prob in topic] for topicid, topic in lsi
    lsitopics_c_uci = [[word for word, prob in topic] for topicid, topic in l
    lsitopics_c_npmi = [[word for word, prob in topic] for topicid, topic in
    lsitopics_u_mass = [[word for word, prob in topic] for topicid, topic in

    lda_coherence = CoherenceModel(topics=ldatopics, texts=train_texts, dicti
    hdp_coherence = CoherenceModel(topics=hdptopics[:5], texts=train_texts, d
    lsi_coherence = CoherenceModel(topics=lsitopics[:5], texts=train_texts, d

    lda_c_v_coherence = CoherenceModel(topics=ldatopics_c_v, texts=train_text
    lda_c_uci_coherence = CoherenceModel(topics=ldatopics_c_uci, texts=train_
    lda_c_npmi_coherence = CoherenceModel(topics=ldatopics_c_npmi, texts=trai
    lda_u_mass_coherence = CoherenceModel(topics=ldatopics_u_mass, texts=trai

    hdp_c_v_coherence = CoherenceModel(topics=hdptopics_c_v, texts=train_text
    hdp_c_uci_coherence = CoherenceModel(topics=hdptopics_c_uci, texts=train_
    hdp_c_npmi_coherence = CoherenceModel(topics=hdptopics_c_npmi, texts=trai
    hdp_u_mass_coherence = CoherenceModel(topics=hdptopics_u_mass, texts=trai

    lsi_c_v_coherence = CoherenceModel(topics=lsitopics_c_v, texts=train_text
    lsi_c_uci_coherence = CoherenceModel(topics=lsitopics_c_uci, texts=train_
    lsi_c_npmi_coherence = CoherenceModel(topics=lsitopics_c_npmi, texts=trai
```

```
lsi_u_mass_coherence = CoherenceModel(topics=lsitopics_u_mass, texts=trai

def evaluate_bar_graph(coherences, indices):
    assert len(coherences) == len(indices)
    n = len(coherences)
    x = np.arange(n)
    plt.bar(x, coherences, width=0.2, tick_label=indices, align='center')
    plt.title('Model Performances: Dataset 2 (300 Long Articles)\n')
    plt.xlabel('Models Tested')
    plt.ylabel('Coherence Values')

evaluate_bar_graph([lda_coherence, hdp_coherence, lsi_coherence,
                    lda_c_v_coherence, lda_c_uci_coherence, lda_c_npmi_co
                    hdp_c_v_coherence, hdp_c_uci_coherence, hdp_c_npmi_co
                    lsi_c_v_coherence, lsi_c_uci_coherence, lsi_c_npmi_co
                   ['LDA', 'HDP', 'LSI', 'LDA_\nC_V', 'LDA_\nC_UCI', 'LDA
```

```
LSI Model Started
[(0, u'0.743*"person" + 0.238*"doctor" + 0.234*"healthcare" +
0.171*"patient" + 0.165*"money" + 0.155*"insurance" + 0.142*"health" +
0.135*"time" + 0.124*"government" + 0.118*"cost"'), (1, u'0.724*"doctor"
+ -0.453*"person" + 0.388*"patient" + 0.152*"hospital" + 0.134*"time" +
0.113*"nurse" + 0.095*"money" + 0.085*"cost" + 0.072*"insurance" +
0.053*"service"'), (2, u'0.842*"healthcare" + -0.338*"person" +
0.179*"insurance" + -0.145*"doctor" + -0.100*"patient" +
-0.085*"hospital" + 0.083*"cost" + 0.083*"service" + 0.080*"government"
+ 0.078*"education"'), (3, u'0.814*"insurance" + -0.294*"healthcare" +
-0.253*"patient" + 0.172*"pay" + 0.171*"service" + -0.125*"time" +
0.089*"care" + 0.087*"health" + -0.085*"person" + 0.080*"bill"'), (4,
u'0.388*"health" + -0.381*"doctor" + 0.309*"money" + -0.265*"healthcare"
+ -0.259*"person" + 0.250*"government" + 0.218*"patient" + 0.214*"time"
+ 0.202*"school" + 0.184*"education"')]
LSI Model Ended

HDP Model Started
[(0, u'0.005*environment + 0.004*calculate + 0.003*life_support +
0.003*run + 0.003*stream + 0.003*explosion + 0.003*train + 0.003*share +
0.003*worker + 0.003*benefit + 0.003*cough + 0.003*privilege +
0.003*england + 0.003*paper + 0.003*sterling + 0.003*empoyee +
0.003*backup + 0.003*one + 0.002*lack + 0.002*pig'), (1, u'0.003*grant +
0.003*policy + 0.003*concentration + 0.003*record + 0.003*middle_class +
0.003*bag + 0.003*soda + 0.003*insert + 0.003*base + 0.003*resource +
0.003*outrage + 0.003*things_like + 0.002*incline + 0.002*combo +
0.002*thing + 0.002*lobbying + 0.002*suffering + 0.002*situation +
0.002*document + 0.002*employer'), (2, u'0.004*bottle +
0.004*psychologist + 0.004*tab + 0.004*tree + 0.004*max + 0.003*home +
0.003*govenment + 0.003*radiation + 0.003*approach + 0.003*puncture +
0.003*reliability + 0.003*restrict + 0.003*marathon + 0.003*jack +
0.003*control + 0.003*food + 0.003*awareness + 0.002*mass +
0.002*hospital + 0.002*member'), (3, u'0.005*gov + 0.005*break +
```

```
[ ]
```