Kérjük, fejtse ki a hibát:

ValueError: This topic model is not currently supported. Supported topic models should implement the `get\_topics` method.

spark

Hypothesis

The error arises because the CoherenceModel in Gensim expects a topic model that has a get\_topics method. The custom RandomTopicModel class defined in the code doesn't have this method, which leads to the ValueError: This topic model is not currently supported. Supported topic models should implement the 'get\_topics' method.

Suggested Changes

library\_add

content\_copy

import numpy as np  
from gensim.models import CoherenceModel  
  
# Assuming you have already:  
# - 'lda\_model': Your trained LDA model  
# - 'corpus': Your document-term matrix  
# - 'dictionary': Your vocabulary dictionary  
# - 'texts': Your original document texts (dict)  
  
# 1. Random Topic Assignment  
num\_topics = lda\_model.num\_topics  
num\_documents = len(corpus)  
random\_topics = np.random.randint(0, num\_topics, num\_documents)  
  
# 2. Calculate Coherence for LDA Model  
lda\_coherence\_model = CoherenceModel(model=lda\_model, texts=list(texts.values()), dictionary=dictionary, coherence='c\_v')  
lda\_coherence = lda\_coherence\_model.get\_coherence()  
  
# 3. Calculate Coherence for Random Baseline  
# Create a dummy model with random topic assignments and a get\_topics method  
class RandomTopicModel:  
    def \_\_init\_\_(self, num\_topics, random\_topics):  
        self.num\_topics = num\_topics  
        self.random\_topics = random\_topics  
  
    def get\_document\_topics(self, bow, minimum\_probability=None):  
        doc\_id = bow[0][0] if isinstance(bow, list) and bow else 0 # Handle different bow formats  
          
        topic\_id = self.random\_topics[doc\_id % len(self.random\_topics)] # Ensure doc\_id is within bounds  
        return [(topic\_id, 1.0)]  # Assign probability 1.0 to the random topic  
      
    def get\_topics(self):  
        # Return a list of topics, each represented as a list of (word\_id, probability) tuples.  
        # Since this is a random model, we'll just return empty topics.  
        return [[] for \_ in range(self.num\_topics)]    
  
random\_topic\_model = RandomTopicModel(num\_topics, random\_topics)  
random\_coherence\_model = CoherenceModel(model=random\_topic\_model, texts=list(texts.values()), dictionary=dictionary, coherence='c\_v')  
random\_coherence = random\_coherence\_model.get\_coherence()  
  
# 4. Compare Coherence Scores  
print(f"LDA Model Coherence: {lda\_coherence}")  
print(f"Random Baseline Coherence: {random\_coherence}")  
  
# You can further visualize the comparison:  
import matplotlib.pyplot as plt  
plt.bar(['LDA', 'Random'], [lda\_coherence, random\_coherence])  
plt.title('Topic Model Coherence Comparison')  
plt.ylabel('Coherence Score')  
plt.show()

2025.05.06