Latent_dirichlet_allocation.py

```
#!/usr/bin/env python
# coding: utf-8

# ## Step 0: Latent Dirichlet Allocation ##

# ## Step 1: Load the dataset
'''1. Load the dataset from the CSV and save it to data_text'''
import pandas as pd
data = pd.read_csv('abcnews-date-text.csv', error_bad_lines=False)
data_text = data[:300000][['headline_text']]
data_text['index'] = data_text.index
documents = data_text

'''2. Data Preprocessing'''
# Loading Gensim and nltk libraries
import gensim
from gensim.utils import simple_preprocess
from gensim.parsing.preprocessing import STOPWORDS
from nltk.stem import WordNetLemmatizer, SnowballStemmer
from nltk.stem.porter import *
import numpy as np
np.random.seed(400)

import nltk
nltk.download('wordnet')

# Write a function to perform the pre processing steps on the entire dataset
stemmer = SnowballStemmer("english")
def lemmatize_stemming(text):
    return stemmer.stem(WordNetLemmatizer().lemmatize(text, pos='v'))

## Tokenize and lemmatize
def preprocess(text):
    result=[]
    for token in gensim.utils.simple_preprocess(text) :
        if token not in gensim.parsing.preprocessing.STOPWORDS and len(token) > 3:
            result.append(lemmatize_stemming(token))
    return result

# Preprocess all the news headlines with map
# **Note**: This may take a few minutes 0.5min in my laptop)
processed_docs = documents['headline_text'].map(preprocess)

'''Bag of words on the dataset'''
# create a dictionary
dictionary = gensim.corpora.Dictionary(processed_docs) #Create a dictionary from 'processed_docs' containing the number of times a word a

# Remove very rare and very common words (OPTIONAL STEP)
dictionary.filter_extremes(no_below=15, no_above=0.1, keep_n=100000)

# doc2bow → BoW
bow_corpus = [dictionary.doc2bow(doc) for doc in processed_docs]

lda_model = gensim.models.LdaModel(bow_corpus,
                                   num_topics = 10,
                                   id2word = dictionary,
                                   passes = 5)

# print each topic's words & weight
for idx, topic in lda_model.print_topics(-1): #idx指的是topic index,topic指的是词&权重构成的向量
    print("Topic: {} \nWords: {}".format(idx, topic))
    print("\n")
'''Output:
    Topic: 0
    Words: 0.021*"health" + 0.021*"nation" + 0.016*"school" + 0.015*"indigen" + 0.013*"communiti" + 0.013*"nuclear" + 0.013*"servic" + 0.
    Topic: 1
    Words: 0.019*"closer" + 0.016*"price" + 0.015*"record" + 0.014*"opposit" + 0.014*"win" + 0.012*"year" + 0.012*"high" + 0.012*"lead" +
    Topic: 2
    Words: 0.042*"plan" + 0.040*"water" + 0.024*"council" + 0.023*"govt" + 0.018*"fund" + 0.014*"concern" + 0.013*"group" + 0.011*"resid'
    Topic: 3
    Words: 0.054*"urg" + 0.028*"govt" + 0.016*"industri" + 0.014*"farmer" + 0.012*"worker" + 0.011*"centr" + 0.011*"stand" + 0.009*"sale'
    Topic: 4
    Words: 0.028*"warn" + 0.020*"iraq" + 0.016*"return" + 0.015*"troop" + 0.013*"leader" + 0.013*"say" + 0.012*"green" + 0.010*"author" +
    Topic: 5
    Words: 0.024*"drought" + 0.019*"death" + 0.013*"bushfir" + 0.013*"compani" + 0.012*"prepar" + 0.012*"toll" + 0.011*"student" + 0.011*
    Topic: 6
    Words: 0.018*"elect" + 0.017*"secur" + 0.015*"blaze" + 0.015*"rule" + 0.013*"chief" + 0.013*"labor" + 0.011*"timor" + 0.011*"firefig
    Topic: 7
    Words: 0.073*"polic" + 0.029*"kill" + 0.028*"crash" + 0.025*"charg" + 0.020*"investig" + 0.019*"attack" + 0.016*"drug" + 0.015*"jail'
    Topic: 8
    Words: 0.032*"court" + 0.030*"claim" + 0.030*"face" + 0.023*"accus" + 0.023*"reject" + 0.019*"defend" + 0.015*"govt" + 0.014*"deni" +
    Topic: 9
    Words: 0.027*"miss" + 0.023*"seek" + 0.018*"search" + 0.014*"begin" + 0.014*"work" + 0.013*"safeti" + 0.013*"continu" + 0.012*"guilt
    '''

'''Testing model'''
for index, score in sorted(lda_model[bow_corpus[document_num]], key=lambda tup: -1*tup[1]):
    print("\nScore: {}\t \nTopic: {}".format(score, lda_model.print_topic(index, 10)))

# Testing model on new data
unseen_document = "My favorite sports activities are running and swimming."

# Data preprocessing step for the unseen document
bow_vector = dictionary.doc2bow(preprocess(unseen_document))
for index, score in sorted(lda_model[bow_vector], key=lambda tup: -1*tup[1]):
    print("Score: {}\t Topic: {}".format(score, lda_model.print_topic(index, 5)))
# The model correctly classifies the unseen document with 'x'% probability to the X category.
```