МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“Київський політехнічний інститут ім. Ігоря Сікорського”
ФАКУЛЬТЕТ Інформатики та Обчислювальної Техніки
КАФЕДРА Інформаційних систем та технологій

Звіт до лабораторної роботи №5

з предмету: Обробка та Аналіз текстових даних на мові Python

Перевірила:
Тимофєєва Ю.С.

Виконли:
студенти групи ІК-01
Філоненко І. Р.
Гацан С. Ю.

КИЇВ - 2023

Тема: Моделювання тем

Мета: Ознайомитись з вирішенням задач пошуку ключових слів та моделювання тем.

Варіант: 11

Завдання:

Створити програму, яка зчитує заданий набір даних з попередньої лабораторної роботи, виділяє текстову частину даних (вони розгляжаються як документи), виконує попередню обробку та завдання відповідно до варіанту. Якщо недостатньо ресурсів для роботи з повним набором даних, можна виділити частину, але таким чином, щоб були присутні усі класи.

1. Застосувати прихований розподіл Діріхле бібліотеки Gensim для моделювання тем. Обрати оптимальну кількість тем, оцінивши та порівнявши різні моделі. Вивести терми, що описують кожну з тем.
2. Використати текст *chesterton-thursday.txt* з корпусу *gutenberg* бібліотеки *nltk* та вивести ключові триграми.

Код програми

```
#!/usr/bin/env python3

import re
import csv
import numpy as np
import pandas as pd

from matplotlib import pyplot as plt

import nltk
from nltk.tokenize import WordPunctTokenizer
from nltk.corpus import stopwords, gutenberg
from nltk.collocations import TrigramCollocationFinder
from nltk.collocations import TrigramAssocMeasures

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import TruncatedSVD, LatentDirichletAllocation, NMF

import gensim

nltk.data.path.append("../Lab2/nltk_data")

prepared_corpus = []

skip_generation = False

ans = input("Do you want to load already prepared corpus? (Y/N) ").lower()

if ans == 'y':
    try:
```

```
        with open("prepared_corpus.txt", 'r') as file:
            for line in file:
                prepared_corpus.append(line.strip())
        skip_generation = True
    except FileNotFoundError:
        print("Prepared corpus not found, generating it.")

if skip_generation is False:
    file = open("news.csv", 'r')
    next(file)  # To skip headers row
    csv = csv.DictReader(file, delimiter=',', quotechar='"', fieldnames=['text',
'label'])
    texts = {i: v for i, v in enumerate(csv)}
    labels = [texts[i]['label'] for i in range(len(texts))]

    wpt = WordPunctTokenizer()

    # Preparing corpus
    corpus = [texts[i]['text'] for i in range(len(texts)) if len(texts[i]
['text']) != 0]


    def preprocess_sentence(sentence):
        sentence = re.sub(r'[^a-zA-Z\s]|http\S+', '', sentence, re.I | re.A)
        sentence = sentence.lower()
        sentence = sentence.strip()
        tokens = wpt.tokenize(sentence)
        filtered_tokens = [token for token in tokens if token not in
stopwords.words('english')]
        sentence = ' '.join(filtered_tokens)
        return sentence

    for sentence in corpus:
        prepared_corpus.append(preprocess_sentence(sentence))

    with open("prepared_corpus.txt", 'w') as file2:
        for sentence in prepared_corpus:
            file2.write(sentence + '\n')

# print("Підготовлений корпус: ")
# print(prepared_corpus)
# print(len(prepared_corpus))

# Limiting corpus (for faster results)

prepared_corpus = prepared_corpus[:len(prepared_corpus)]

# --------
# Creating id2word dictionary

norm_papers = [sentence.split() for sentence in prepared_corpus]

bigram = gensim.models.Phrases(norm_papers, min_count=10, threshold=20,
delimiter='_')
bigram_model = gensim.models.phrases.Phraser(bigram)

print(bigram_model)

norm_corpus_bigrams = [bigram_model[doc] for doc in norm_papers]
dictionary = gensim.corpora.Dictionary(norm_corpus_bigrams)

print(dictionary)

# --------
```

```
# Creating BOW (Bag Of Words)

bow_corpus = [dictionary.doc2bow(text) for text in norm_corpus_bigrams]

# --------
# Task 1. Latent Dirichlet Allocation
if 1:
    print("LDA Gensim:")
    Total_topics = 10
    lda_model = gensim.models.LdaModel(
        corpus = bow_corpus,       # +
        id2word = dictionary,      # +
        chunksize=1416,            # +
        alpha = 'auto',            # +
        eta = 'auto',              # +
        random_state = 0,          # +
        iterations=500,            # +
        num_topics = Total_topics, # +
        passes = 20,               # +
        eval_every=None            # +
    )

    for topic_id, topic in lda_model.print_topics(num_topics=10, num_words=20):
        print('Topic #'+str(topic_id+1)+':')
        print(topic)

    print("Coherence: ", end='')
    topics_coherences = lda_model.top_topics(bow_corpus, topn=20)
    avg_coherence_score = np.mean([item[1] for item in topics_coherences])
    print(avg_coherence_score)
    ## --------
    ## Analysing coherence
    cv_coherence_model_lda = gensim.models.CoherenceModel(
        model=lda_model, corpus=bow_corpus, texts=norm_corpus_bigrams,
        dictionary=dictionary, coherence='c_v')
    avg_coherence_cv = cv_coherence_model_lda.get_coherence()
    print("Avg. Coherence Score (CV): " + str(avg_coherence_cv))

    umass_coherence_model_lda = gensim.models.CoherenceModel(
        model=lda_model, corpus=bow_corpus, texts=norm_corpus_bigrams,
        dictionary=dictionary, coherence='u_mass')
    avg_coherence_umass = umass_coherence_model_lda.get_coherence()
    print("Avg. Coherence Score (U-Mass): " + str(avg_coherence_umass))

    perplexity = lda_model.log_perplexity(bow_corpus)
    print("Model Perplexity: " + str(perplexity))
# --------
# Now compare it to other Total_topics numbers
x = []
y1 = []
y2 = []
y3 = []
ans = input("Do you want to load already prepared corpus? (Y/N) ").lower()
skip_generation = False
if ans == 'y':
    try:
        with open("prepared_graph.txt", 'r') as file:
            line = []
            for l in file:
                line.append(l.strip())

            i = 0
            while line[i] != '---':
                y1.append(float(line[i]))
```

```
                i = i + 1

            i = i + 1
            while line[i] != '---':
                y2.append(float(line[i]))
                i = i + 1

            i = i + 1
            while line[i] != '---':
                y3.append(float(line[i]))
                i = i + 1

            for v in range(1, len(y1)+1):
                x.append(v)

        skip_generation = True
    except FileNotFoundError:
        print("Prepared corpus not found, generating it.")

print(x)
print(y1)
print(y2)
print(y3)

if skip_generation is False:
    for i in range(1, 20 +1) :
        print("i = " + str(i) + ":")
        x.append(i)
        lda_model = gensim.models.LdaModel(
            corpus = bow_corpus,       # +
            id2word = dictionary,      # +
            chunksize=1416,            # +
            alpha = 'auto',            # +
            eta = 'auto',              # +
            random_state = 0,          # +
            iterations=500,            # +
            num_topics = i,            # +
            passes = 20,               # +
            eval_every=None            # +
        )

        print("Coherence: ", end='')
        topics_coherences = lda_model.top_topics(bow_corpus, topn=20)
        avg_coherence_score = np.mean([item[1] for item in topics_coherences])
        print(avg_coherence_score)
        y1.append(avg_coherence_score)
        ## --------
        ## Analysing coherence
        cv_coherence_model_lda = gensim.models.CoherenceModel(
            model=lda_model, corpus=bow_corpus, texts=norm_corpus_bigrams,
            dictionary=dictionary, coherence='c_v')
        avg_coherence_cv = cv_coherence_model_lda.get_coherence()
        print("Avg. Coherence Score (CV): " + str(avg_coherence_cv))
        y2.append(avg_coherence_cv)

        umass_coherence_model_lda = gensim.models.CoherenceModel(
            model=lda_model, corpus=bow_corpus, texts=norm_corpus_bigrams,
            dictionary=dictionary, coherence='u_mass')
        avg_coherence_umass = umass_coherence_model_lda.get_coherence()
        print("Avg. Coherence Score (U-Mass): " + str(avg_coherence_umass))
        y3.append(avg_coherence_umass)

        perplexity = lda_model.log_perplexity(bow_corpus)
        print("Model Perplexity: " + str(perplexity))
```

```
    with open("prepared_graph.txt", 'w') as file2:
        for y in y1:
            file2.write(str(y) + '\n')
        file2.write("---\n")
        for y in y2:
            file2.write(str(y) + '\n')
        file2.write("---\n")
        for y in y3:
            file2.write(str(y) + '\n')
        file2.write("---\n")

fig, axs = plt.subplots(2)
axs[0].plot(x, y2, marker = 'o')
axs[1].plot(x, y3, marker = 'o')
plt.show()

# --------
# Task 2. NLTK Gutenberg Trigrams
print("Gutenberg trigrams:")

gutenberg_sents = gutenberg.sents(fileids="chesterton-thursday.txt")
gutenberg_sents = [' '.join(gutenberg_sents[i]) for i in
range(len(gutenberg_sents))]

prepared_corpus = []
wpt = WordPunctTokenizer()

def preprocess_sentence(sentence):
    sentence = re.sub(r'[^a-zA-Z\s]+', '', sentence, re.I | re.A)
    sentence = sentence.lower()
    sentence = sentence.strip()
    tokens = wpt.tokenize(sentence)
    filtered_tokens = [token for token in tokens if token not in
stopwords.words('english')]
    sentence = filtered_tokens
    return sentence

for sentence in gutenberg_sents:
    prepared_corpus.append(preprocess_sentence(sentence))

finder = TrigramCollocationFinder.from_documents(prepared_corpus)
trigram_measures = TrigramAssocMeasures()

print("Top 10 raw frequency: ", end='')
print(finder.nbest(trigram_measures.raw_freq, 10))

print("Top 10 PMI: ", end='')
print(finder.nbest(trigram_measures.pmi, 10))

print("Top 10 Likelihood Ratio: ", end='')
print(finder.nbest(trigram_measures.likelihood_ratio, 10))

# --------
```

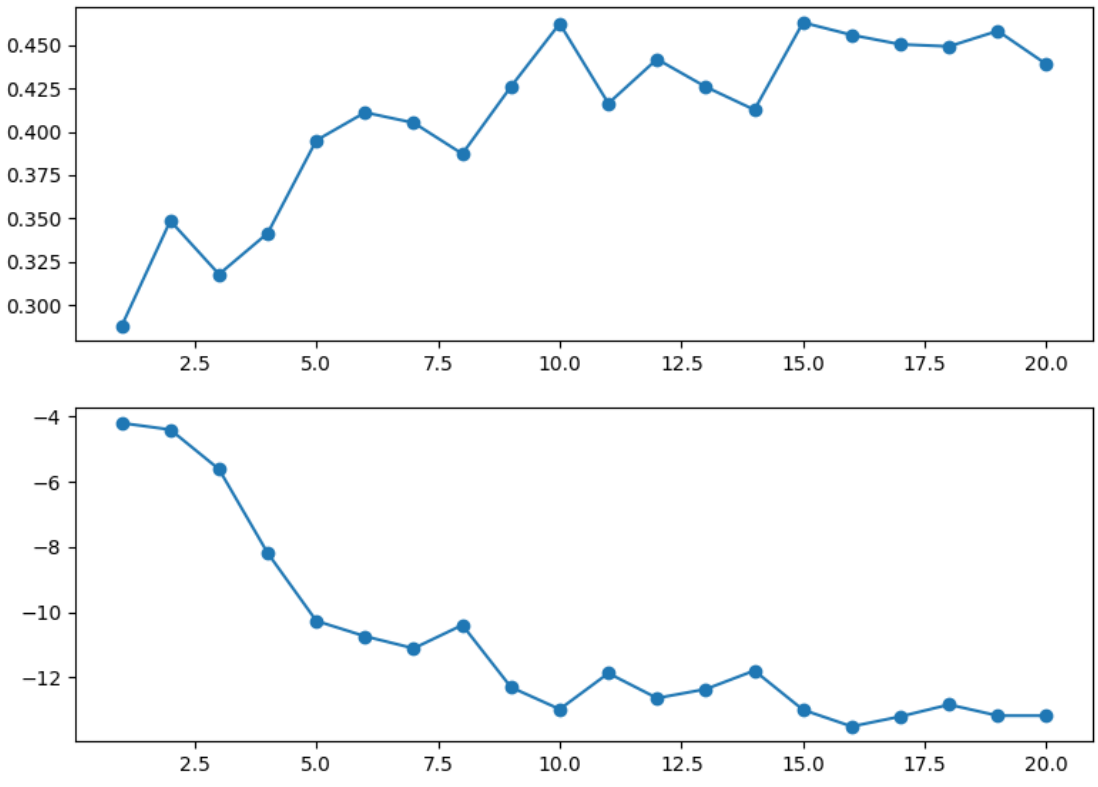
Результат роботи програми

```
Do you want to load already prepared corpus? (Y/N) Y
LDA Gensim:
Topic #1:
0.017*"cpi" + 0.014*"volume" + 0.011*"meeting" + 0.011*"sa" + 0.010*"national" +
0.010*"house" + 0.010*"concerns" + 0.009*"spy_spx" + 0.009*"stop" +
0.008*"shows" + 0.008*"johnson" + 0.007*"made" + 0.007*"already" +
0.007*"remain" + 0.007*
"consumers" + 0.007*"warns" + 0.007*"federal" + 0.006*"trying" + 0.006*"aug" +
0.006*"focus"
Topic #2:
0.041*"pre" + 0.026*"higher" + 0.016*"ahead" + 0.013*"low" + 0.012*"si" +
0.011*"index" + 0.010*"chinese" + 0.009*"gap" + 0.009*"supply" + 0.009*"rally" +
0.009*"break" + 0.008*"calls" + 0.008*"buying" + 0.007*"holding" +
0.007*"become" +
0.007*"nations" + 0.007*"resistance" + 0.006*"daily" + 0.006*"money" +
0.006*"past"
Topic #3:
0.027*"stocks" + 0.023*"markets" + 0.020*"economy" + 0.020*"business" +
0.019*"finance" + 0.017*"investing" + 0.013*"stock" + 0.011*"market" +
0.010*"shares" + 0.008*"buy" + 0.008*"company" + 0.008*"investors" +
0.007*"could" + 0.007*"earn
ings" + 0.007*"prices" + 0.007*"growth" + 0.007*"back" + 0.007*"china" +
0.007*"uk" + 0.006*"report"
Topic #4:
0.017*"reports" + 0.015*"strong" + 0.015*"chinas" + 0.012*"results" +
0.010*"close" + 0.010*"bonds" + 0.010*"short" + 0.009*"b" + 0.008*"holdings" +
0.007*"way" + 0.006*"bond" + 0.006*"policy" + 0.006*"bottom" +
0.006*"expectations" + 0.00
6*"mortgage" + 0.006*"nflx" + 0.005*"current" + 0.005*"countrys" + 0.005*"puts"
+ 0.005*"net"
Topic #5:
0.017*"since" + 0.014*"spy" + 0.012*"crypto" + 0.009*"rising" + 0.008*"point" +
0.008*"well" + 0.007*"costs" + 0.007*"qqq" + 0.007*"fuel" + 0.007*"august" +
0.007*"ecb" + 0.007*"president_biden" + 0.007*"breakingviews" + 0.006*"tuesday"
+
0.006*"euro" + 0.006*"rishi_sunak" + 0.006*"draghi" + 0.006*"white_house" +
0.006*"india" + 0.006*"esf"
Topic #6:
0.024*"president" + 0.013*"rate" + 0.012*"appoints" + 0.012*"rates" +
0.010*"fed" + 0.010*"crisis" + 0.009*"nasdaq" + 0.009*"points" + 0.008*"dont" +
0.007*"earlier" + 0.007*"political" + 0.007*"stake" + 0.007*"put" + 0.007*"nept"
```

+ 0.007*
"need" + 0.006*"bid" + 0.006*"sold" + 0.005*"fight" + 0.005*"squeeze" +
0.005*"bounce"
Topic #7:
0.023*"announces" + 0.014*"amp" + 0.014*"trade" + 0.013*"million" + 0.012*"inc"
+ 0.008*"bill" + 0.008*"firm" + 0.008*"likely" + 0.007*"eu" + 0.007*"months" +
0.007*"health" + 0.007*"dollar" + 0.007*"partners" + 0.006*"acquisition" + 0.006
"agreement" + 0.006"leader" + 0.006*"senate" + 0.006*"cost" + 0.005*"digital"
+ 0.005*"lead"
Topic #8:
0.041*"us" + 0.035*"trading" + 0.024*"inflation" + 0.024*"stockmarket" +
0.015*"year" + 0.014*"may" + 0.013*"june" + 0.012*"energy" + 0.012*"says" +
0.011*"high" + 0.010*"billion" + 0.010*"according" + 0.010*"next" + 0.009*"said"
+ 0.009*"
day" + 0.008*"support" + 0.008*"price" + 0.008*"debt" + 0.007*"time" +
0.007*"key"
Topic #9:
0.042*"new" + 0.019*"global" + 0.011*"economic" + 0.010*"open" +
0.008*"european" + 0.008*"power" + 0.007*"credit" + 0.007*"people" +
0.007*"news" + 0.007*"services" + 0.007*"officer" + 0.006*"industry" +
0.006*"long" + 0.006*"europe" + 0.
006*"second" + 0.006*"make" + 0.006*"state" + 0.006*"country" + 0.006*"prev" +
0.005*"go"
Topic #10:
0.021*"government" + 0.013*"prime_minister" + 0.013*"last" +
0.012*"economy_stockmarket" + 0.012*"spy_qqq" + 0.012*"guidance" + 0.010*"green"
+ 0.007*"hold" + 0.007*"court" + 0.006*"saudi_arabia" + 0.006*"election" +
0.006*"japan" + 0.006*
"dia_spx" + 0.006*"red" + 0.005*"natural_gas" + 0.005*"italys" + 0.005*"expect"
+ 0.005*"compq_djia" + 0.005*"meet" + 0.005*"hot"

Coherence: -12.973130077975323
Avg. Coherence Score (CV): 0.46245879325437916
Avg. Coherence Score (U-Mass): -12.973130077975323
Model Perplexity: -10.00772374278859

```
Do you want to load already prepared corpus? (Y/N) Y
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
[-4.197212753317936, -4.399437544135994, -5.612731257340049, -8.183914223300842,
-10.2650781363116, -10.736559614639928, -11.106826676002369, -
10.392253231313301, -12.30008504540332, -12.973130077975323, -
11.869918124751326, -12.6325036258
73987, -12.358675068770244, -11.789734455871468, -12.99213642625416, -
13.50172047720238, -13.19710990086702, -12.832014206005596, -13.171256716036988,
-13.171010940496322]
[0.2878849710656254, 0.34858587424225995, 0.3177355155022696,
0.3414336581735825, 0.39503505396958427, 0.4111593867341233, 0.4051744632266407,
0.3872694070746636, 0.42611897403642174, 0.46245879325437916,
0.4163911628432528, 0.441929031607
35075, 0.4259473690137853, 0.4127013409105927, 0.4630693582067981,
0.45589323972344575, 0.450474470186558, 0.4492931533666931, 0.45823259648465064,
0.43898705960095813]
[-4.197212753317936, -4.399437544135994, -5.612731257340049, -8.183914223300842,
-10.265078136311601, -10.736559614639928, -11.106826676002369, -
10.392253231313301, -12.300085045403323, -12.973130077975323, -
11.869918124751326, -12.6325036
25873985, -12.358675068770243, -11.789734455871468, -12.992136426254165, -
13.50172047720238, -13.19710990086702, -12.832014206005598, -13.171256716036988,
-13.171010940496322]
```



Gutenberg trigrams:
Top 10 raw frequency: [('said', 'dr', 'bull'), ('professor', 'de', 'worms'), ('well', 'said', 'syme'), ('yes', 'said', 'syme'), ('could', 'almost', 'fancy'), ('de', 'st', 'eustache'), ('man', 'dark', 'room'), ('said', 'syme', 'seriously'),
('asked', 'syme', 'sort'), ('blue', 'card', 'pocket')]
Top 10 PMI: [('acquainted', 'secretarial', 'duties'), ('allies', 'providential', 'stoppage'), ('archangel', 'judging', 'justly'), ('argent', 'chevron', 'gules'), ('arts', 'occasionally', 'political'), ('barnum', 'freak', 'physically'), ('b ellegarde', 'baron', 'zumpt'), ('bite', 'slice', 'bread'), ('blasphemy', 'blend', 'angel'), ('bludgeon', 'fist', 'fisherman')]
Top 10 Likelihood Ratio: [('said', 'dr', 'bull'), ('dr', 'bull', 'desperately'), ('dr', 'bull', 'smiled'), ('countenance', 'dr', 'bull'), ('dr', 'bull', 'bubbling'), ('dr', 'bull', 'precision'), ('dr', 'bull', 'sphinx'), ('wheeler', 'dr',
'bull'), ('dr', 'bull', 'suddenly'), ('dr', 'bull', 'adamantine')]