Министерство науки и высшего образованияРоссийской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Рязанский государственный радиотехнический университет

имени В.Ф.Уткина»

Кафедра «Электронные вычислительные машины»

Отчет по практическому занятию №11

на тему

«Введение в обработку естественного языка»

по дисциплине  
**«Машинное обучение»**

Выполнили:

Студенты группы №140

Бригада 7

Сафаров Д. А.

Тимохин Е. М.

Проверила:  
ас. каф. ЭВМ Панина И.С.

**Цель работы**

Получение студентом навыков реализации базовых методов обработки естественного языка, включая предобработку текста, формирование «мешка слов» («bag-of-words»), выделение стоп-слов и наиболее важных слов в документе, создание тематических моделей.

**Задание**

Сгенерируйте по 3 примера текстов для отзывов на 5 звезд и на 3 звезды. Сравните полученные результаты. Сделайте выводы. Самостоятельно приведите сгенерированный набор к семантически верному виду.

**Практическая часть**

Код программы представлен в приложении А. Результат её работы представлен на рисунке 1.

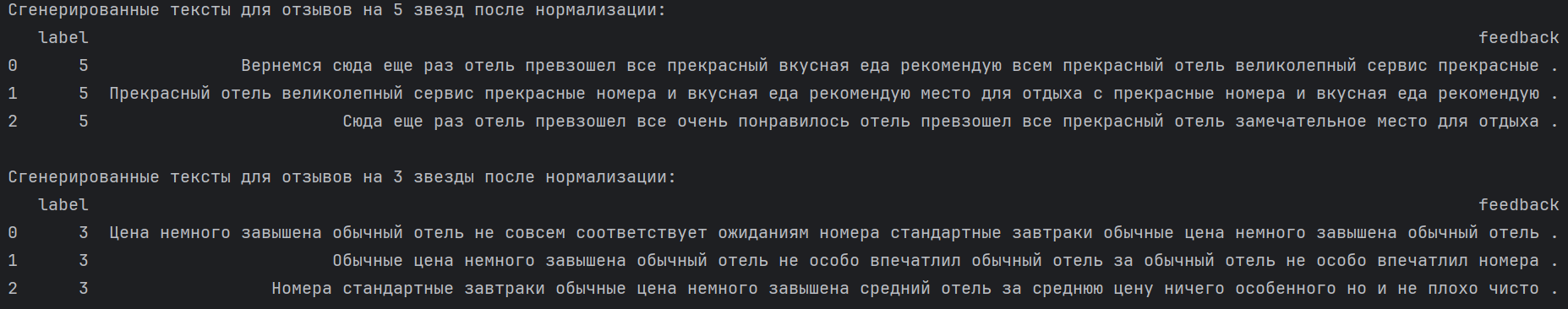


Рисунок 1 – Результат выполнения программы

Приложение А. Текст программы

import pandas as pd  
import numpy as np  
import sklearn  
import re  
import nltk  
  
from nltk import tokenize  
from nltk.tokenize import RegexpTokenizer  
  
import random  
from collections import Counter, defaultdict  
  
nltk.download('punkt')  
def get\_text\_for\_label(df, label):#Функция получает тексты для заданной категории (label) из DataFrame.  
 label\_query = 'label == ' + str(label)  
 df\_label = df.query(label\_query).drop(['label'], axis=1)  
 return df\_label.feedback.values.tolist()  
  
def tokenize\_sentences(text\_corp):#Функция токенизирует предложения из списка текстов.  
 token\_corp = []  
 tokenizer = RegexpTokenizer(r'\w+')  
 for text in text\_corp:  
 sentences = tokenize.sent\_tokenize(text)  
 for sent in sentences:  
 tokens = tokenizer.tokenize(sent)  
 token\_corp.extend(tokens)  
 token\_corp.append('END\_SENT\_START') # В конце каждого предложения добавляем фиктивный токен  
 return token\_corp  
  
def get\_bigrams(token\_list):#Функция создает список биграмм из списка токенов  
 bigram\_corp = []  
 for i in range(len(token\_list) - 1):  
 bigram = token\_list[i] + ' ' + token\_list[i + 1]  
 bigram\_corp.append(bigram) # Получим список биграмм  
 return bigram\_corp  
  
'''  
Функция генерирует тексты на основе списка токенов и биграмм.  
Биграмма - это последовательность из двух слов, которые встречаются рядом друг с другом в тексте.  
Она принимает параметры: метку (label),   
количество генерируемых текстов (count\_text),   
количество предложений в каждом тексте (count\_sent),   
количество слов в каждом предложении (count\_word).  
'''  
  
  
'''  
Генерация текстов:  
  
Для каждого текста генерируется случайное начальное слово.  
Затем для каждого предложения в тексте генерируются случайные слова на основе биграмм (пар последовательных слов) из корпуса.  
Если для текущего слова не найдено подходящее следующее слово в биграммах, выбирается случайное слово из всего корпуса.  
Сгенерированный текст добавляется в список текстов.  
Создается DataFrame из списка сгенерированных текстов, включая метку (label).  
'''  
def generate\_texts(token\_list, bigram\_list, label, count\_text, count\_sent, count\_word):  
 # Создаем словарь для подсчета биграмм "исключений"  
 exceptions\_bigramm = defaultdict(int)  
 texts = []  
 for it\_text in range(count\_text): # Цикл с диапазоном кол-ва текстов  
 text = ''  
 for it\_sent in range(count\_sent): # Цикл с диапазоном кол-ва предложений в тексте  
 final\_sent = ''  
 # Генерируем случайное слово для начала предложения для обеспечения стохастического процесса генерации предложения  
 start\_word = random.choice(token\_list)  
 final\_sent += start\_word.capitalize()  
 for step in range(count\_word - 1):  
 next\_word = None  
 # Формируем список биграмм, начинающихся с текущего слова  
 possible\_bigrams = [bigram for bigram in bigram\_list if bigram.startswith(start\_word)]  
 if possible\_bigrams:  
 # Случайно выбираем следующее слово из списка биграмм  
 next\_bigram = random.choice(possible\_bigrams)  
 next\_word = next\_bigram.split()[1]  
 if next\_word is None:  
 # Если не удалось найти подходящее слово, выбираем случайное слово из всего корпуса  
 next\_word = random.choice(token\_list)  
 final\_sent += ' ' + next\_word  
 start\_word = next\_word  
 final\_sent += '. '  
 text += final\_sent  
 texts.append(text)  
 generation\_text\_df = pd.DataFrame(texts, columns=['feedback']) # Формируем фрейм из списка  
 generation\_text\_df['label'] = label  
 return generation\_text\_df[['label', 'feedback']]  
  
df\_fin\_feedback = pd.DataFrame({  
 'feedback': [  
 # Объемные отзывы для 5 звезд  
 "Отличный отель! Великолепный сервис, прекрасные номера и вкусная еда. Рекомендую всем!",  
 "Прекрасный отель! Замечательное место для отдыха с семьей. Бассейн, детский клуб, все очень понравилось.",  
 "Отель превзошел все ожидания! Уютные номера, великолепный вид, вежливый персонал. Обязательно вернемся сюда еще раз.",  
 # Объемные отзывы для 4 звезд  
 "Хороший отель. В целом все понравилось, но были небольшие недочеты. Номера чистые, персонал внимательный.",  
 "Приятный отель. Неплохое расположение, удобные номера, но немного шумно. Завтраки вкусные.",  
 "Хороший отель за свою цену. Номера комфортные, персонал дружелюбный. Но есть небольшие проблемы с чистотой.",  
 # Объемные отзывы для 3 звезд  
 "Средний отель. Не совсем соответствует ожиданиям. Номера чистые, но не очень уютные. Персонал отзывчивый.",  
 "Обычный отель. Не особо впечатлил. Номера стандартные, завтраки обычные. Цена немного завышена.",  
 "Средний отель за среднюю цену. Ничего особенного, но и не плохо. Чисто, но нет ничего выдающегося."  
 ],  
 'label': [5, 5, 5, 4, 4, 4, 3, 3, 3] # Указываем соответствующие рейтинги для каждого отзыва  
})  
  
def normalize\_text(text):  
 # Удаление фиктивного токена  
 text = text.replace('end\_sent\_start', '')  
 # Удаление лишних символов и пробелов  
 text = re.sub(r'[^\w\s]', '', text)  
 # Коррекция заглавных букв  
 text = text.capitalize()  
 # Добавление точки в конце предложения, если ее нет  
 if not text.endswith('.'):  
 text += '.'  
 return text  
  
# Получаем тексты и биграммы для отзывов на 5 звезд  
feedback\_label\_5\_stars = get\_text\_for\_label(df\_fin\_feedback, 5)  
token\_label\_5\_stars = tokenize\_sentences(feedback\_label\_5\_stars)  
bigram\_label\_5\_stars = get\_bigrams(token\_label\_5\_stars)  
  
# Получаем тексты и биграммы для отзывов на 3 звезды  
feedback\_label\_3\_stars = get\_text\_for\_label(df\_fin\_feedback, 3)  
token\_label\_3\_stars = tokenize\_sentences(feedback\_label\_3\_stars)  
bigram\_label\_3\_stars = get\_bigrams(token\_label\_3\_stars)  
  
generated\_texts\_label\_5\_stars = generate\_texts(token\_label\_5\_stars, bigram\_label\_5\_stars, label=5, count\_text=3, count\_sent=2, count\_word=10)  
  
generated\_texts\_label\_5\_stars['feedback'] = generated\_texts\_label\_5\_stars['feedback'].apply(normalize\_text)  
  
  
generated\_texts\_label\_3\_stars = generate\_texts(token\_label\_3\_stars, bigram\_label\_3\_stars, label=3, count\_text=3, count\_sent=2, count\_word=10)  
  
generated\_texts\_label\_3\_stars['feedback'] = generated\_texts\_label\_3\_stars['feedback'].apply(normalize\_text)  
  
  
# Убираем фиктивный токен из сгенерированных текстов перед их обработкой  
generated\_texts\_label\_5\_stars['feedback'] = generated\_texts\_label\_5\_stars['feedback'].str.replace(' end\_sent\_start', '')  
generated\_texts\_label\_3\_stars['feedback'] = generated\_texts\_label\_3\_stars['feedback'].str.replace(' end\_sent\_start', '')  
# Устанавливаем максимальную ширину столбца feedback равной None, чтобы выводить полный текст  
pd.set\_option('display.max\_colwidth', None)  
print("Сгенерированные тексты для отзывов на 5 звезд после нормализации:")  
print(generated\_texts\_label\_5\_stars)  
  
print("\nСгенерированные тексты для отзывов на 3 звезды после нормализации:")  
print(generated\_texts\_label\_3\_stars)  
  
'''  
ВЫВОД ПО ГЕНЕРАЦИИ   
Судя по результатам, тексты для отзывов на 5 звезд выглядят более позитивными что указывает на положительный опыт пребывания в отеле.   
С другой стороны, тексты для отзывов на 3 звезды выражают более сдержанную оценку и содержат слова, указывающие на недостатки.  
Таким образом, генерация текстов отражает различия в отзывах на отели с разными оценками, и можно сделать вывод,   
что модель генерации текстов учитывает контекст и генерирует соответствующие оценке отзывы.  
'''

**Вывод**

Были получены студентом навыки реализации базовых методов обработки естественного языка, включая предобработку текста, формирование «мешка слов» («bag-of-words»), выделение стоп-слов и наиболее важных слов в документе, создание тематических моделей.