**ОТЧЕТ К ЛАБОРАТОРНОЙ РАБОТЕ № 3**

**Лабораторная работа № 3**

**Модель открытого текста**

**Вариант №5**

Ф. И. О. студента: Гюнтер Тимофей Вячеславович

Группа: ФИТ-221

Проверил: Дата:

**Основные сведения**

Энтропия открытого текста определяется формулой:

**Результаты**

Энтропия для k-грамм открытого текста:

График зависимости /k от k:

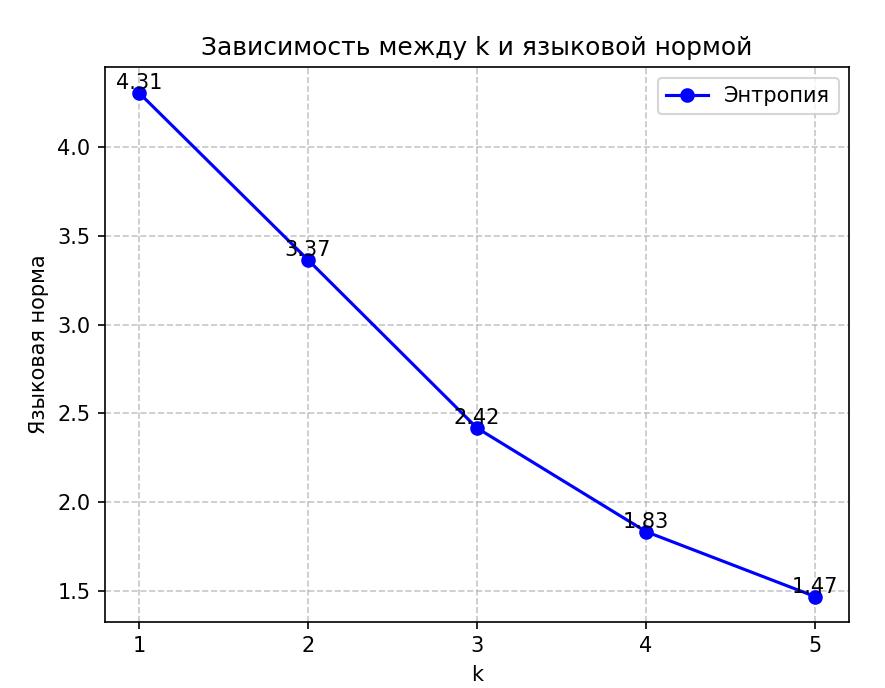


Рисунок 1- График зависимости k от /k

Асимптотическое значение /k при :

**Код программы**

from lab1 import Caesar, alphabet

preprocess = Caesar.preprocessing

rus\_alphabet = alphabet

text = 'Я был разбужен спозаранку Щелчком оконного стекла. Размокшей \

каменной баранкой В воде Венеция плыла. Все было тихо, и, однако, Во \

сне я слышал крик, и он Подобьем смолкнувшего знака Еще тревожил \

небосклон'

# def k\_gramms

preprocessed\_text = preprocess(text)

import numpy as np

def get\_k\_gramms\_entries(text: str, k: int):

    k\_gramms =  [text[i : i + k] for i in range(len(text) - k + 1)]

    n = len(k\_gramms)

    # print('n', n, k\_gramms)

    k\_gramms\_freq = {gramm: text.count(gramm)/n for gramm in set(k\_gramms)}

    return dict(sorted(k\_gramms\_freq.items(), key=lambda x: x[1], reverse=True))

# s = 'разбиениеби'

# example\_text = preprocess(s)

# k\_gramms\_freq\_dict = get\_k\_gramms\_entries(example\_text, 2)

# # calculate\_entropy\_and\_language\_norma()

# print(k\_gramms\_freq\_dict)

# grams = k\_grams(text, k)

# frequences = Counter()

def get\_k\_gramms\_frequency\_dict(text: str):

    '''Метод для получения частотного словаря для данного текста'''

    k\_interval = range(1, 5+1)

    k\_gramms\_dict = {key: {} for key in k\_interval}

    for k in k\_interval:

        gramm\_dict = {}

        for bias in range(k):

            for i in range(bias, len(text)-k+1, k):

                k\_gramm = text[i:i+k]

                if all(symbol in rus\_alphabet for symbol in k\_gramm):

                    if k\_gramm not in gramm\_dict: gramm\_dict[k\_gramm] = 1

                    else: gramm\_dict[k\_gramm] += 1

        k\_gramms\_dict[k] = gramm\_dict

    for k in k\_gramms\_dict:

        k\_gramms\_dict[k] = dict(sorted(k\_gramms\_dict[k].items(), key=lambda x: x[1], reverse=True))

        n = sum(list(k\_gramms\_dict[k].values()))

        for key in k\_gramms\_dict[k]:

            frequency = k\_gramms\_dict[k][key]/n

            k\_gramms\_dict[k][key] = frequency

    return k\_gramms\_dict

# dict\_ans = get\_k\_gramms\_frequency\_dict(example\_text)

def calculate\_entropy\_and\_language\_norma(k: int, k\_gramm\_dict: list):

   entropy = -sum(k\_gramm\_freq\*np.log2(k\_gramm\_freq) for k\_gramm\_freq in k\_gramm\_dict)

   lang\_norma = -sum(k\_gramm\_freq\*np.log2(k\_gramm\_freq)/k for k\_gramm\_freq in k\_gramm\_dict)

   return entropy, lang\_norma

k\_gramms\_dict = get\_k\_gramms\_frequency\_dict(preprocessed\_text)

entropy\_dict = {}

lang\_norma = []

for k in range(1, 5+1):

    entropy, lang\_norma\_value = calculate\_entropy\_and\_language\_norma(k, k\_gramms\_dict[k].values())

    entropy\_dict[k] = entropy

    lang\_norma.append(lang\_norma\_value)

# calculate\_entropy(k\_gramms.values())

import matplotlib.pyplot as plt

def k\_entropy\_graphic(k, lang\_norma):

    plt.plot(k, lang\_norma, label='Энтропия', color='b', marker='o')  # Кривая с маркерами

    for i in range(len(k)):

        plt.text(k[i], lang\_norma[i], f'{lang\_norma[i]:.2f}', ha='center', va='bottom', fontsize=10)

    plt.xlabel('k')

    plt.ylabel('Языковая норма')

    plt.title('Зависимость между k и языковой нормой')

    plt.xticks(k)

    plt.grid(True, linestyle='--', alpha=0.7)

    plt.legend()

    plt.show()

def display\_graphic():

    k\_entropy\_graphic(list(entropy\_dict.keys()), lang\_norma)

def entropy\_lang\_norm\_interface():

    for k in range(1, 5+1):

        entropy, lang\_norma\_value = calculate\_entropy\_and\_language\_norma(k, k\_gramms\_dict[k].values())

        entropy\_dict[k] = entropy

        print('---------------------------------------------')

        print(f'k: {k}, энтропия: {round(entropy, 3)}, языковая норма: {round(lang\_norma\_value, 3)}')

# entropy\_lang\_norm\_interface()