**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский Авиационный Институт»**

**(Национальный Исследовательский Университет)**

**Институт: №8 «Информационные технологии   
и прикладная математика»   
Кафедра: 806 «Вычислительная математика   
и программирование»**

Лабораторная работа № 3   
по курсу «Криптография»

Группа: М8О-312Б-22

Студент: Куценко М.Д.

Преподаватель: А. В. Борисов

Оценка:

Дата: 16.04.2025

Москва, 2025

ОГЛАВЛЕНИЕ

[1 Тема 3](#__RefHeading___Toc272_1095258672)

[2 Задание 3](#__RefHeading___Toc274_1095258672)

[3 Теория 4](#__RefHeading___Toc276_1095258672)

[4 Ход лабораторной работы 5](#__RefHeading___Toc278_1095258672)

[5 Выводы 11](#__RefHeading___Toc280_1095258672)

[6 Список используемой литературы 12](#__RefHeading___Toc282_1095258672)

# **Тема**

Сравнение осмысленных текстов и случайных текстов.

# **Задание**

Сравнить 1) два осмысленных текста на естественном языке, 2) осмысленный текст и текст из случайных букв, 3) осмысленный текст и текст из случайных слов, 4) два текста из случайных букв, 5) два текста из случайных слов.

Считать процент совпадения букв в сравниваемых текстах – получить дробное значение от 0 до 1 как результат деления количества совпадений на общее число букв. Расписать подробно в отчёте алгоритм сравнения и приложить сравниваемые тексты в отчёте хотя бы для одного запуска по всем пяти случаям. Осознать какие значения получаются в этих пяти случаях. Привести соображения о том почему так происходит.

Длина сравниваемых текстов должна совпадать. Привести соображения о том какой длины текста должно быть достаточно для корректного сравнения

# **Теория**

Сравнение текстов — определение различий и сходств среди двух данных последовательностей символов.

# **Ход лабораторной работы**

Для осмысленных текстов были взяты первые 3000 символов из начала произведений «Гарри Поттер» и «Властелин Колец» на английском языке.

Для случайных слов я использовал генерацию с использованием LLM, сделав два набора по 3000 символов.

Для случайных букв я сделал простой случайный генератор на языке Python:

import random  
import string  
  
def generate\_random\_text(length):  
  letters = string.ascii\_letters  
    
  random\_text = ''.join(random.choice(letters) for \_ in range(length))  
    
  return random\_text

Далее я выбрал 4 функции сравнения букв в текстах:

from collections import Counter  
  
def compare\_texts\_1(text1, text2, length):  
  counter = 0  
  for i in range (length):  
      if text1[i] == text2[i]:  
          counter +=1  
  return counter/length  
  
def compare\_texts\_2(text1, text2, length):  
  text1\_new = text1.lower()  
  text2\_new = text2.lower()  
  return compare\_texts\_1(text1\_new, text2\_new, length)  
  
def filter\_text(text):  
  chars\_to\_remove = {' ', ":", ".", "\"", "'", ";", ''', '\n', '-', ''', ','}  
  text\_new = text.lower()  
  for char in chars\_to\_remove:  
      text\_new = text\_new.replace(char, "")  
  return text\_new  
  
def compare\_texts\_3(text1, text2, length):  
  return compare\_texts\_1(filter\_text(text1), filter\_text(text2), length)  
  
def compare\_texts\_4(text1, text2, length):  
  count1 = Counter(filter\_text(text1)[:length])  
  count2 = Counter(filter\_text(text2)[:length])  
  return sum((count1 & count2).values())/length

compare\_texts\_1 сравнивает все символы двух текстов последовательно.

compare\_texts\_2 сравнивает все символы двух текстов последовательно без учёта регистра.

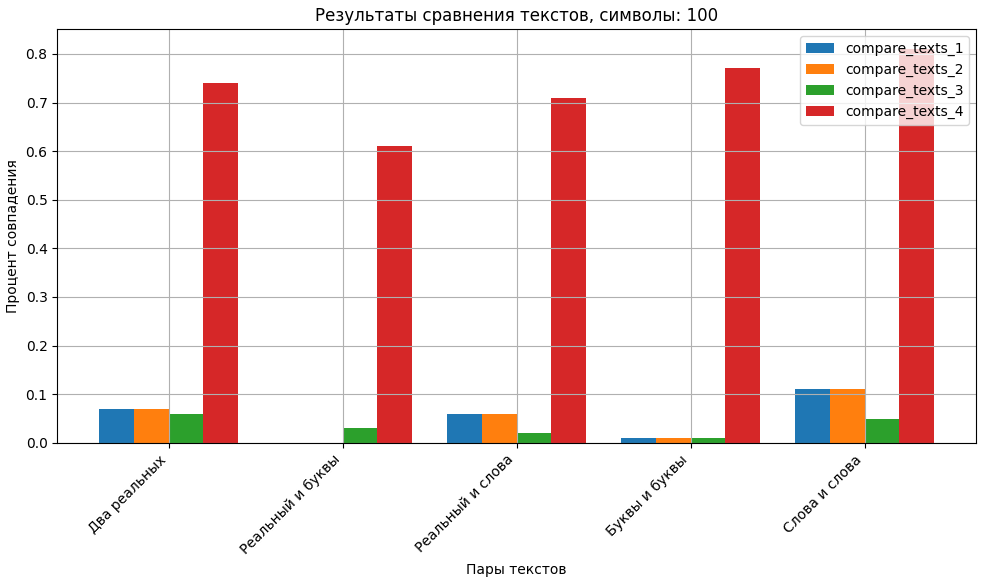
compare\_texts\_3 сравнивает все символы двух текстов последовательно без учёта регистра, а также с удалением пробелов и знаков препинания. Фильтрация текстов производится через filter\_text.

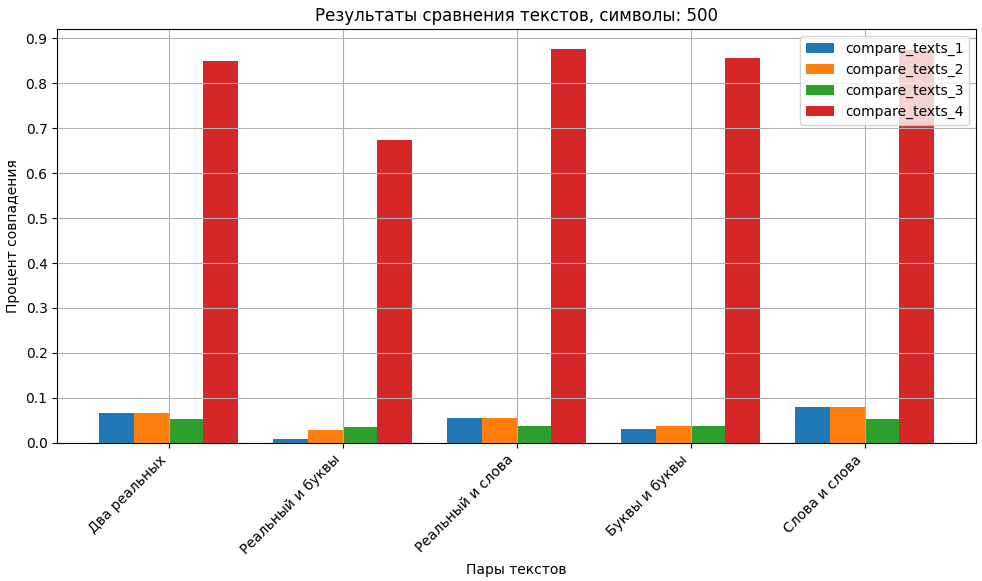
compare\_texts\_4 сравнивает все символы двух текстов без учёта последовательности символов, без учёта регистра, а также с удалением пробелов и знаков препинания. В этой функции в словарь заносится вся информация об символах из двух строк, после чего находится общая часть двух словарей и считается отношение для неё.

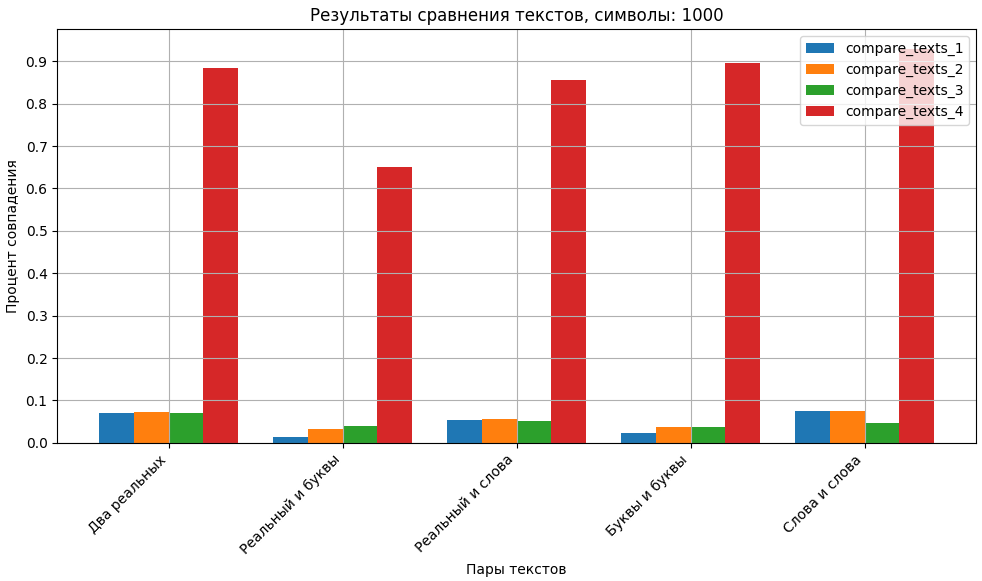
Далее при помощи следующей функции производится отрисовка графиков:

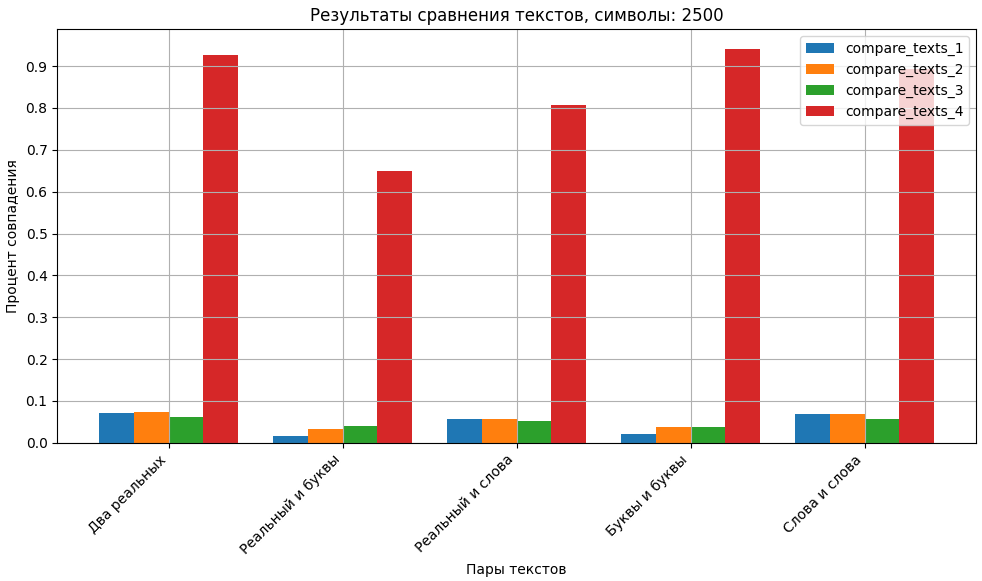
import matplotlib.pyplot as plt  
from matplotlib.ticker import MultipleLocator  
import numpy as np  
  
functions = [compare\_texts\_1, compare\_texts\_2, compare\_texts\_3, compare\_texts\_4]  
  
def plot\_comparison\_results(written, letter, word, comparison\_functions, length):  
  
  pair\_labels = ["Два реальных", "Реальный и буквы", "Реальный и слова", "Буквы и буквы", "Слова и слова"]  
  text\_pairs = [(written[0], written[1]), (written[0], letter[0]), (written[0], word[0]), (letter[0], letter[1]), (word[0], word[1])]  
  
  results = {metric: [] for metric in [func.\_\_name\_\_ for func in comparison\_functions]}  
  for a, b in text\_pairs:  
      for func in comparison\_functions:  
          results[func.\_\_name\_\_].append(func(a, b, length))  
    
  n\_metrics = len(comparison\_functions)  
  n\_pairs = len(text\_pairs)  
  bar\_width = 0.8 / n\_metrics  
  index = np.arange(n\_pairs)  
    
  plt.figure(figsize=(10, 6))  
    
  for i, (metric, values) in enumerate(results.items()):  
      plt.bar(index + i \* bar\_width, values, bar\_width, label=metric)  
    
  plt.xlabel('Пары текстов')  
  plt.ylabel('Процент совпадения')  
  plt.title(f'Результаты сравнения текстов, символы: {length}')  
  plt.xticks(index + bar\_width \* (n\_metrics - 1) / 2, pair\_labels, rotation=45, ha='right')  
  plt.gca().yaxis.set\_major\_locator(MultipleLocator(0.1))  
  plt.legend()  
  plt.grid()  
  plt.tight\_layout()  
  plt.show()

Посмотрим на результаты для разного числа символов:







Как мы видим, общая тенденция заметна даже на небольшом числе символов, однако взятие больших объёмов текста позволяет получить более точные результаты анализа.

Так, для compare\_texts\_1 - compare\_texts\_3 наименьшие результаты также прослеживаются для сравнений, использующих случайные буквы. Это позволяет сделать вывод что в реальных текстах, в отличии от текстов из случайных букв, как правило одни буквы следуют за другими.

Наименьшее значение compare\_texts\_4 прослеживается для сравнения рукописного текста с текстом из случайно сгенерированных букв. Это показывает что в реальных текстах в отличии от текстов, состоящих из случайных букв, частота использования различных букв неодинакова.

# **Выводы**

На примере анализа естественных и случайно сгенерированных текстов нашёл следующие две закономерности:

* В естественных текстах частота использования различных букв неодинакова.
* В естественных текстах есть закономерности следования одних букв за другими.

# **Список используемой литературы**

1. Применко Э.А. Алгебраические основы криптографии: Учебное пособие. — М.: Книжный дом «ЛИБРОКОМ», 2013. — 288 с.
2. Грушо А.А., Применко Э.А., Тимонина Е.Е. Теоретические основы компьютерной безопасности: учеб. пособие для студентов высш. учеб. Заведений. — М.: Издательский центр «Академия», 2009. — 272 с.