**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(ка): Л. Д, Андрюшин

Преподаватель: А. В. Борисов

Оценка:

Дата: 17.04.2025

Москва, 2025

**ОГЛАВЛЕНИЕ**

[1 Тема 3](#_Toc158983147)

[2 Задание 3](#_Toc158983148)

[3 Теория 4](#_Toc158983149)

[4 Ход лабораторной работы 5](#_Toc158983150)

[5 Выводы 9](#_Toc158983151)

# **Тема**

Лабораторная работа №3

# **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**

Для сравнения текстов существует много разных способов. Можно, например, использовать расстояние Левенштейна — это когда считается, сколько символов нужно вставить, удалить или заменить, чтобы один текст превратился в другой. Есть более «умные» методы вроде косинусного сходства, когда тексты превращают в векторы и смотрят, насколько они похожи по направлению. Также можно сравнивать тексты с помощью n-грамм, частотных характеристик, распределения символов и других статистических приёмов.

Я решил не усложнять и использовать обычное посимвольное сравнение. Суть в том, что я просто беру два текста одинаковой длины и считаю, сколько букв на одинаковых позициях совпадают. Получаю количество совпадений и делю на общее количество букв — так выходит число от 0 до 1. Такой способ простой, но наглядный и вполне подходит для целей этой лабораторной.

# **Ход лабораторной работы**

Для начала я взял три «осмысленных текста». Это просто случайные главы из различных английских книжек и привел их к нужному мне формату. Я убрал все символы кроме букв и оставил их только 1000 из каждого текста. Я решил что 1000 символов вполне достаточно и они уже могут обеспечить нормальную точность. Вот код, с помощью которого я делал «осмысленные тексты»:

import re

def extract\_letters(input\_path, output\_path, letter\_limit=1000):

    with open(input\_path, 'r', encoding='utf-8') as infile:

        text = infile.read().lower()  # читаем и приводим к нижнему регистру

    letters = re.findall(r'[a-z]', text)

    limited = letters[:letter\_limit]

    result = ''.join(limited)

    with open(output\_path, 'w', encoding='utf-8') as outfile:

        outfile.write(result)

input\_files = ['book1.txt', 'book2.txt', 'book3.txt']

output\_files = ['text1\_clean.txt', 'text2\_clean.txt', 'text3\_clean.txt']

for input\_path, output\_path in zip(input\_files, output\_files):

    extract\_letters(input\_path, output\_path)

print("Готово! Тексты сохранены.")

Вот код, с помощью которого я уже непосредственно сравивал количество совпадающих букв:

import random

import string

import nltk

from nltk.corpus import words

nltk.download('words')

word\_list = words.words()

def generate\_random\_letters(length=1000):

    random.seed()

    return ''.join(random.choices(string.ascii\_lowercase, k=length))

def generate\_random\_word\_string(length=1000):

    random.seed()

    result = ''

    while len(result) < length:

        word = random.choice(word\_list).lower()

        word = ''.join([c for c in word if c.isalpha()])

        result += word

    return result[:length]

def compare\_texts(text1, text2):

    matches = sum(c1 == c2 for c1, c2 in zip(text1, text2))

    return matches / len(text1)

def read\_text(path):

    with open(path, 'r', encoding='utf-8') as f:

        text = f.read().lower()

        text = ''.join(filter(str.isalpha, text))

        return text[:1000]

text1 = read\_text('text1\_clean.txt')

text2 = read\_text('text2\_clean.txt')

text3 = read\_text('text3\_clean.txt')

random\_letters = generate\_random\_letters()

random\_words = generate\_random\_word\_string()

print("1) Осмысленный текст vs осмысленный текст:")

print("   →", compare\_texts(text1, text2))

print("2) Осмысленный текст vs случайные буквы:")

print("   →", compare\_texts(text1, random\_letters))

print("3) Осмысленный текст vs случайные слова:")

print("   →", compare\_texts(text1, random\_words))

print("4) Случайные буквы vs случайные буквы:")

print("   →", compare\_texts(random\_letters, generate\_random\_letters()))

print("5) Случайные слова vs случайные слова:")

print("   →", compare\_texts(random\_words, generate\_random\_word\_string()))

В начале кода подключаются нужные библиотеки: random используется для генерации случайных значений, string нужен, чтобы получить алфавит, а nltk — это библиотека для работы с текстами. Из неё мы берём список английских слов (nltk.corpus.words), чтобы потом использовать для генерации случайных слов. Сразу загружается словарь через nltk.download('words'), и все слова сохраняются в переменную word\_list.

Функция generate\_random\_letters(length=1000) просто создаёт строку из 1000 случайных букв. Каждая буква выбирается из английского алфавита в нижнем регистре. Сначала вызывается random.seed(), чтобы обеспечить случайность, если код запускается несколько раз.

Функция generate\_random\_word\_string(length=1000) делает похожее, только вместо отдельных букв она берёт случайные английские слова из словаря. Каждое слово приводится к нижнему регистру и очищается от символов, которые не являются буквами (на всякий случай). Цикл продолжается, пока общая длина строки не станет хотя бы 1000 символов, и в конце результат обрезается до ровно 1000 символов.

Функция compare\_texts(text1, text2) — основа всего сравнения. Она берёт два текста одинаковой длины, сравнивает каждый символ на соответствующих позициях и считает, сколько из них совпадают. Потом это количество делится на длину текста, и в итоге получается значение от 0 до 1 — процент совпадений.

Функция read\_text(path) открывает указанный текстовый файл, считывает из него весь текст, переводит его в нижний регистр, убирает всё, кроме букв, и берёт только первые 1000 букв. Это нужно, чтобы привести все входные данные к одинаковому виду и длине.

Далее идёт основная часть: из трёх заранее подготовленных файлов считываются осмысленные тексты. Также создаются две случайные строки — одна из случайных букв, другая из случайных слов. После этого проводится 5 сравнений.

Пример сравнения двух осознанных текстов:

Текст 1:

Hediscomposureofspiritswhichthisextraordinaryvisitthrewelizabethintocouldnotbeeasilyovercomenorcouldsheformanyhourslearntothinkofitlessthanincessantlyladycatherineitappearedhadactuallytakenthetroubleofthisjourneyfromrosingsforthesolepurposeofbreakingoffhersupposedengagementwithmrdarcyitwasarationalschemetobesurebutfromwhatthereportoftheirengagementcouldoriginateelizabethwasatalosstoimaginetillsherecollectedthathisbeingtheintimatefriendofbingleyandherbeingthesisterofjanewasenoughatatimewhentheexpectationofoneweddingmadeeverybodyeagerforanothertosupplytheideashehadnotherselfforgottentofeelthatthemarriageofhersistermustbringthemmorefrequentlytogetherandherneighboursatlucaslodgethereforeforthroughtheircommunicationwiththecollinsesthereportsheconcludedhadreachedladycatherinehadonlysetthatdownasalmostcertainandimmediatewhichshehadlookedforwardtoaspossibleatsomefuturetimeinrevolvingladycatherinesexpressionshowevershecouldnothelpfeelingsomeuneasinessastothepossibleconsequenceofherpersistingint (это текст уже после обработки, что очевидно по внешнему виду)

Текст 2:

itisnoonanddoughboythestewardthrustinghispaleloafofbreadfacefromthecabinscuttleannouncesdinnertohislordandmasterwhosittingintheleequarterboathasjustbeentakinganobservationofthesunandisnowmutelyreckoningthelatitudeonthesmoothmedallionshapedtabletreservedforthatdailypurposeontheupperpartofhisivorylegfromhiscompleteinattentiontothetidingsyouwouldthinkthatmoodyahabhadnotheardhismenialbutpresentlycatchingholdofthemizenshroudsheswingshimselftothedeckandinanevenunexhilaratedvoicesayingdinnermrstarbuckdisappearsintothecabinwhenthelastechoofhissultansstephasdiedawayandstarbuckthefirstemirhaseveryreasontosupposethatheisseatedthenstarbuckrousesfromhisquietudetakesafewturnsalongtheplanksandafteragravepeepintothebinnaclesayswithsometouchofpleasantnessdinnermrstubbanddescendsthescuttlethesecondemirloungesabouttheriggingawhileandthenslightlyshakingthemainbracetoseewhetheritwillbeallrightwiththatimportantropehelikewisetakesuptheoldburdenandwitharapiddinnermrflaskfollowsafterhispredecessorsbutthethirde

# **Выводы**

Результаты показывают интересную закономерность. Наибольший процент совпадений наблюдается при сравнении осмысленного текста с другим осмысленным текстом (примерно 6.5%) и при сравнении случайных слов между собой (около 6.6%). Это можно объяснить тем, что в обоих случаях используются реальные английские слова, а значит — буквы распределяются не случайным образом, а с определённой логикой. Например, часто встречаются сочетания вроде "th", "in", "er", и т.д., и они могут совпадать на одних и тех же позициях даже при разных текстах.

При сравнении осмысленного текста со случайными словами совпадений оказалось тоже довольно много — 6.1%. Это логично, так как случайные слова тоже состоят из настоящих английских слов, пусть и с другой смысловой структурой. Их распределение букв всё ещё ближе к естественному языку, чем у просто случайных букв.

Сравнение осмысленного текста со случайными буквами даёт гораздо меньший процент совпадений (примерно 3.3%). Это ожидаемо, потому что случайные буквы генерируются полностью без логики и без учёта частоты появления букв в английском языке. Совпадения тут скорее случайны и редки.

Наконец, при сравнении случайных букв между собой получаем примерно 3.9%, что чуть больше, чем при сравнении случайных букв с осмысленным текстом. Это тоже объясняется вероятностью случайных совпадений, но без какой-либо структуры они происходят достаточно редко.

Таким образом, можно сделать вывод, что тексты, в которых сохраняется структура языка (даже если они сгенерированы случайно), совпадают друг с другом чаще, чем тексты, построенные из случайных букв. Сами значения совпадений невысоки — это ожидаемо, так как точное посимвольное совпадение между двумя разными текстами встречается довольно редко.

# **Список используемой литературы**

1. <https://gutenberg.org/ebooks/48320?utm_source=chatgpt.com>
2. <https://www.gutenberg.org/ebooks/2701/?utm_source=chatgpt.com>
3. <https://gutenberg.org/ebooks/1342?utm_source=chatgpt.com>