КОМП’ЮТЕРНИЙ ПРАКТИКУМ №1

Гоголєва Поліна ФБ-12

Експериментальна оцінка ентропії на символ джерела відкритого тексту

Мета роботи: Засвоєння понять ентропії на символ джерела та його надлишковості, вивчення та порівняння різних моделей джерела відкритого тексту для наближеного визначення ентропії, набуття практичних навичок щодо оцінки ентропії на символ джерела.

Хід роботи

0. Уважно прочитати методичні вказівки до виконання комп’ютерного практикуму.

Єдиний пункт, з котрим не було проблем і складнощів! (формули налякали not gonna lie)

1. Написати програми для підрахунку частот букв і частот біграм в тексті, а також підрахунку H1 та H2 за безпосереднім означенням. Підрахувати частоти букв та біграм, а також значення H1 та H2 на довільно обраному тексті російською мовою достатньої довжини (щонайменше 1Мб), де імовірності замінити відповідними частотами. Також одержати значення H1 та H2 на тому ж тексті, в якому вилучено всі пробіли.

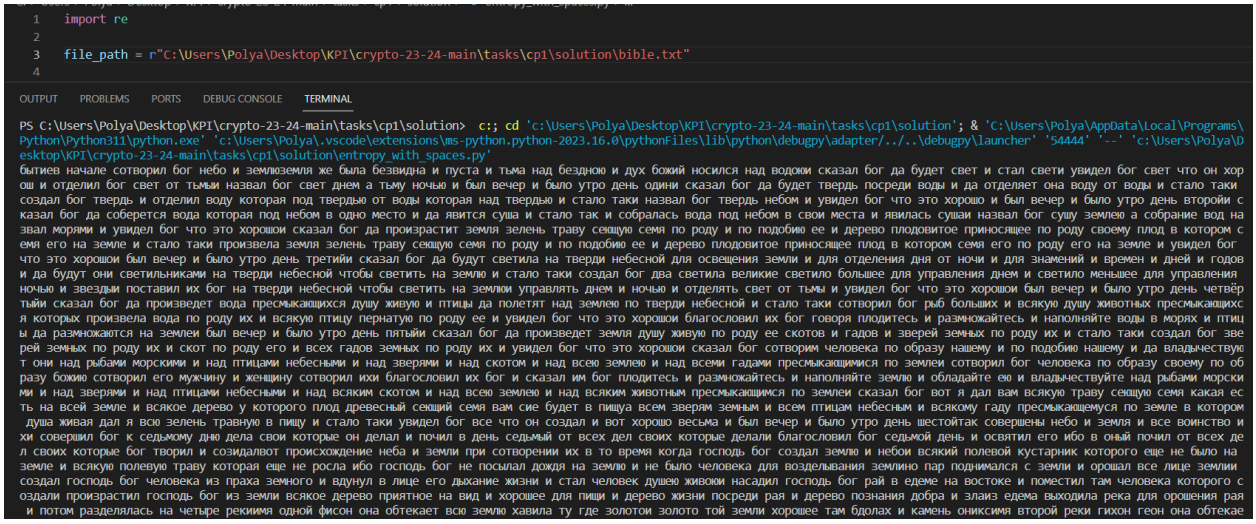
Для початку напишемо частину коду, що буде загружати текст з файлу та фільтрувати його відносно вимог методички, цитую:
«всі символи, окрім текстових, повинні вилучатись або замінюватись на пробіли; прописні літери – замінюватись на відповідні стрічні; послідовність пробілів (або інших розділових знаків, наприклад, символів кінця рядку) повинна трактуватись як один пробіл або вилучатись, якщо пробіл не входить до алфавіту.»

Як текстовий файл я обрала біблію, там води точно більше ніж на 1Мб.

```
import re

file_path = r"C:\Users\Polya\Desktop\KPI\crypto-23-24-
main\tasks\cp1\solution\bible.txt"

with open(file_path, "r") as file:

    cleaned_text = ''

    for line in file:

        line = re.sub(r'[^а-яА-ЯёЁ]', ' ', line)

        # Видалення зайвих пробілів

        line = ' '.join(line.split())
```

```
        cleaned_text += line

print (cleaned_text)
```

print написала просто щоб потестити чи все правильно працює, бідний пітон ледь не вмер друкуючи всю біблію, але все працює, а всі послідовності символів і пробілів замінились на один пробіл.



Тепер рахуватимемо частоти букв і біграм.

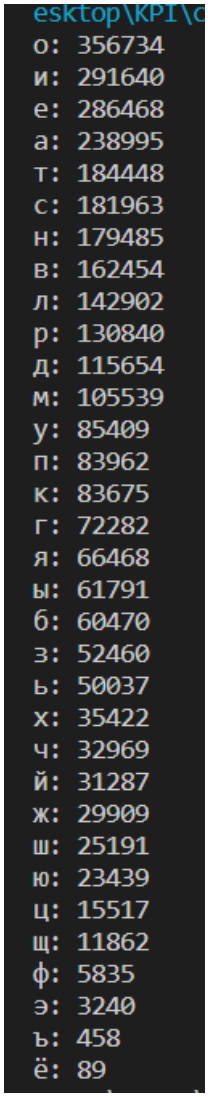
Отримуємо частотний аналіз літер у Біблії (моя віруюча бабуся зараз напевно у захваті, що я цікавлюсь Біблією). Для цього створимо словник зі значенням літера:її кількість у тексті, прогоним весь алфавіт через цикл і додаватимо до countera одиничку щоразу, як зустрічатимемо літеру.

```
import re

file_path = r"C:\Users\Polya\Desktop\KPI\crypto-23-24-
main\tasks\cp1\solution\bible.txt"

with open(file_path, "r") as file:

    cleaned_text = ''

    for line in file:

        line = re.sub(r'[^a-яА-ЯёЁ]', ' ', line)

        # Видалення зайвих пробілів

        line = ' '.join(line.split())

        cleaned_text += line.lower()

print (cleaned_text)


letter_counts={}

for letter in cleaned_text:

    if letter in letter_counts:

        letter_counts[letter] += 1
```

```
    else:
        letter_counts[letter] = 1
sorted_letter_counts = dict(sorted(letter_counts.items(), key=lambda item:
item[1], reverse=True))
```

(тут я поняла, що одна літера upper case і та ж літера lower case вважається за дві різні літери алфавіту і спростила собі життя, переводячи весь текст у нижній регістр).

Тож так ми дізнались, що літера «о» зустрічається у Біблії 356734 разів:



Тепер рахуватимемо частотність біграм з перетином і без, принцип як і у минулому пункті, але беремо по два символи і ітеруємо через кожні 2(у випадку без перетину) або 1(у випадку з перетином) символ.

Нюанси: При обрахунках ми від довжини тексту віднімаємо 1, оскільки для підрахунку біграм потрібно розглядати пари символів, і останній символ не може бути початком нової біграми.
Також для підрахунту частот біграм з перетином ми витягаємо пару символів з тексту, розпочинаючи з індексу і і закінчуючи індексом і + 2. Це створює

біграму з перетином, оскільки дві послідовні пари символів будуть мати одну спільну літеру (перший символ наступного біграму є останнім символом попереднього).

Вийшло щось отаке:

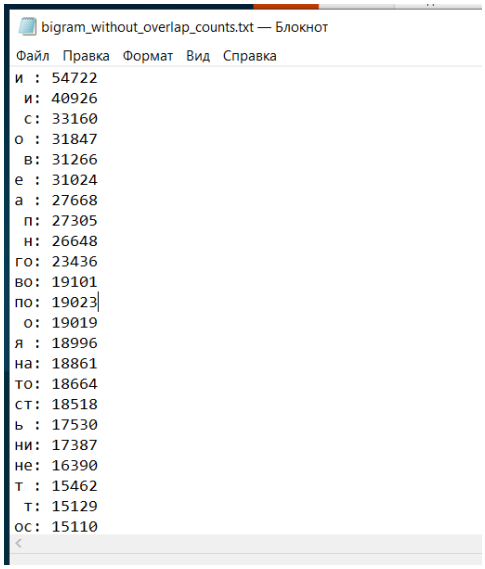
```
bigram_with_overlap_counts = {}

bigram_without_overlap_counts = {}

for i in range(len(cleaned_text) - 1):

    #Підрахуємо частоту біграм з перетином

    bigram_with_overlap = cleaned_text[i:i + 2]

    if bigram_with_overlap in bigram_with_overlap_counts:

        bigram_with_overlap_counts[bigram_with_overlap] += 1

    else:

        bigram_with_overlap_counts[bigram_with_overlap] = 1

    if i % 2 == 0:

        #І без перетину

        bigram_without_overlap = cleaned_text[i:i + 2]

        if bigram_without_overlap in bigram_without_overlap_counts:

            bigram_without_overlap_counts[bigram_without_overlap] += 1

        else:

            bigram_without_overlap_counts[bigram_without_overlap] = 1

sorted_bigram_with_overlap_counts =
dict(sorted(bigram_with_overlap_counts.items(), key=lambda item: item[1],
reverse=True))

sorted_bigram_without_overlap_counts =
dict(sorted(bigram_without_overlap_counts.items(), key=lambda item: item[1],
reverse=True))print("Частотний аналіз біграм з перетином:")

for bigram, count in sorted_bigram_with_overlap_counts.items():

    print(f"{bigram}: {count}")

print("Частотний аналіз біграм без перетину:")

for bigram, count in sorted_bigram_without_overlap_counts.items():

    print(f"{bigram}: {count}")
```

Вивід виявився величеньким і не вміщався у термінал, тому я закинула його у текстові файли, щоб була візуалізація і було з чого робити таблички частотності і прибрала прінт, приблизно отак це виглядало для кожної з функцій

```
with open("letters_count.txt", "w") as letters_file:
```

```
    letters_file.write("Частотний аналіз літер у тексті:\n")
    for letter, count in sorted_letter_counts.items():
        letters_file.write(f"{letter}: {count}\n")
```

Тому тепер у мене є файлики з частотним аналізом біграм і літер. Так як ми вважаємо, що пробіл є частиною нашого алфавіту, то його частотність теж підраховується і біграми формату «пробіл-літера» теж існують.



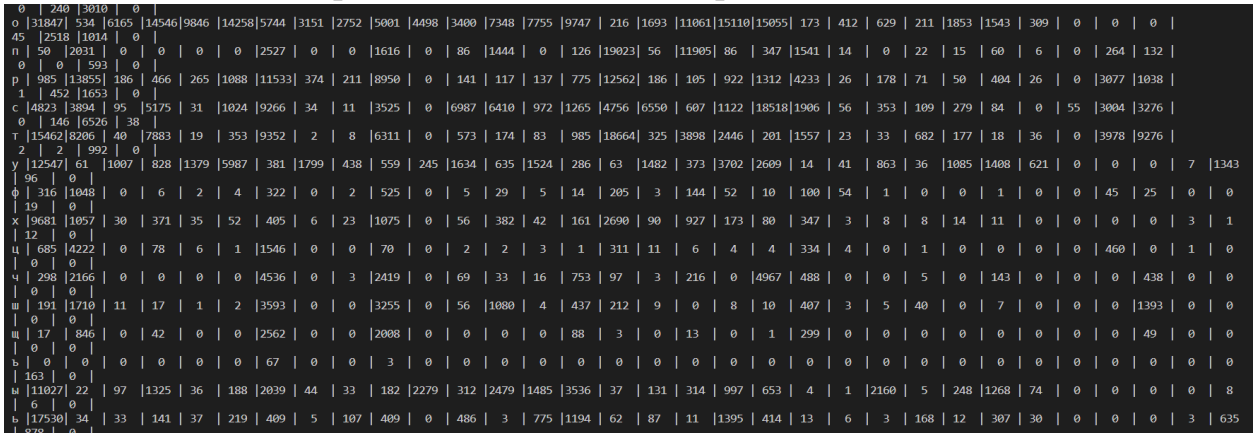
Ну тепер треба створити таблички частот букв та біграм, для більш зручної візуалізації і та сприйняття даних.

Таблиця частот букв:

Символ	Частотність
пробіл	640177
о	356734
и	291640
е	286468
а	238995
т	184448
с	181963
н	179485
в	162454
л	142902
р	130840
д	115654
м	105539
у	85409
п	83962
к	83675
г	72282
я	66468
ы	61791

б	60470
з	52460
ь	50037
х	35422
ч	32969
й	31287
ж	29909
ш	25191
ю	23439
ц	15517
щ	11862
ф	5835
э	3240
ъ	458
ё	89

Таблицю частот біграм так просто не перепишеш, тому я модифікувала код так, щоб він сам створював та виводив матрицю. Але вивід виявився нечитабельним, бо просто не вміщався на екран

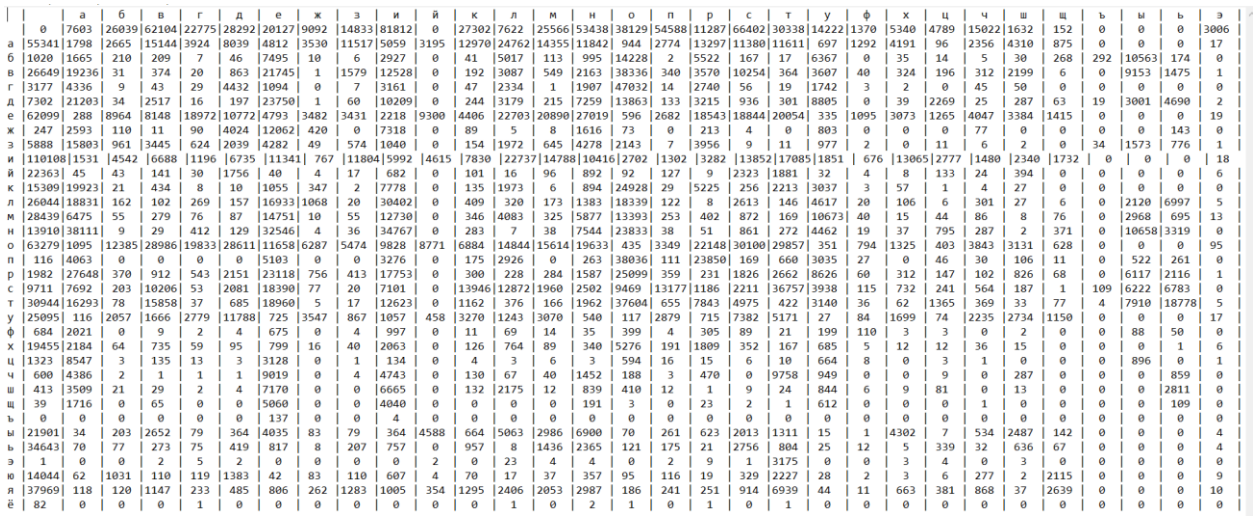


Тому я вирішила експортувати результати у файли, додавши у код отаку функцію

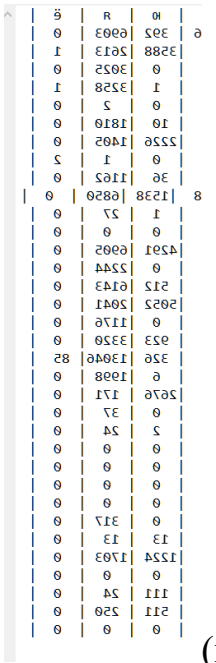
```
def save_matrix_to_file(matrix, file_name):
    with open(file_name, "w") as output_file:
        alphabet = sorted(sorted_letter_counts.keys())
        output_file.write(f"  |" + "|".join(f"{letter:^5}" for letter in
alphabet) + "|\n")
        for letter1 in alphabet:
            row = []
            for letter2 in alphabet:
                bigram = letter1 + letter2
                count = matrix.get(bigram, 0)
```

```
                row.append(f"{count:^5}")

            output_file.write(f"{letter1} |" + "|".join(row) + "|\n")


# Зберігаємо матрицю біграм з перетином

save_matrix_to_file(bigram_with_overlap_counts,
"bigram_with_overlap_matrix.txt")


# Зберігаємо матрицю біграм без перетину

save_matrix_to_file(bigram_without_overlap_counts,
"bigram_without_overlap_matrix.txt")
```

Запустили і отримали такі результати для біграм з перетином:



	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э
	0	7603	26039	62104	22775	28292	20127	9092	14833	81812	0	27302	7622	25566	53438	38129	54588	11287	66402	30338	14222	1370	5340	4789	15022	1632	152	0	0	0	3006
а	55341	1798	2665	15144	3924	8039	4812	3530	11517	5059	3195	12970	24762	14355	11842	944	2774	13297	11380	11611	697	1292	4191	96	2356	4310	875	0	0	0	17
б	1020	1665	210	209	7	46	7495	10	6	2927	0	41	5017	113	995	14228	2	5522	167	17	6367	0	35	14	5	30	268	292	10563	174	0
в	26649	19236	31	374	20	863	21745	1	1579	12528	0	192	3087	549	2163	38336	340	3570	10254	364	3607	40	324	196	312	2199	6	0	9153	1475	1
г	3177	4336	9	43	29	4432	1094	0	7	3161	0	47	2334	1	1907	47032	14	2740	56	19	1742	3	2	0	45	50	0	0	0	0	0
д	7302	21203	34	2517	16	197	23750	1	60	10209	0	244	3179	215	7259	13863	133	3215	936	301	8805	0	39	2269	25	287	63	19	3001	4690	2
е	62099	288	8964	8148	18972	10772	4793	3482	3431	2218	9300	4406	22703	20890	27019	596	2682	18543	18844	20054	335	1095	3073	1265	4047	3384	1415	0	0	0	19
ж	247	2593	110	11	90	4024	12062	420	0	7318	0	89	5	8	1616	73	0	213	4	0	803	0	0	0	77	0	0	0	0	143	0
з	5888	15803	961	3445	624	2039	4282	49	574	1040	0	154	1972	645	4278	2143	7	3956	9	11	977	2	0	11	6	2	0	34	1573	776	1
и	110108	1531	4542	6688	1196	6735	11341	767	11804	5992	4615	7830	22737	14788	10416	2702	1302	3282	13852	17085	1851	676	13065	2777	1480	2340	1732	0	0	0	18
й	22363	45	43	141	30	1756	40	4	17	682	0	101	16	96	892	92	127	9	2323	1881	32	4	8	133	24	394	0	0	0	0	6
к	15309	19923	21	434	8	10	1055	347	2	7778	0	135	1973	6	894	24928	29	5225	256	2213	3037	3	57	1	4	27	0	0	0	0	0
л	26044	18831	162	102	269	157	16933	1068	20	30402	0	409	320	173	1383	18339	122	8	2613	146	4617	20	106	6	301	27	6	0	2120	6997	5
м	28439	6475	55	279	76	87	14751	10	55	12730	0	346	4083	325	5877	13393	253	402	872	169	10673	40	15	44	86	8	76	0	2968	695	13
н	13910	38111	9	29	412	129	32546	4	36	34767	0	283	7	38	7544	23833	38	51	861	272	4462	19	37	795	287	2	371	0	10658	3319	0
о	63279	1095	12385	28986	19833	28611	11658	6287	5474	9828	8771	6884	14844	15614	19633	435	3349	22148	30100	29857	351	794	1325	403	3843	3131	628	0	0	0	95
п	116	4063	0	0	0	0	5103	0	0	3276	0	175	2926	0	263	38036	111	23850	169	660	3035	27	0	46	30	106	11	0	522	261	0
р	1982	27648	370	912	543	2151	23118	756	413	17753	0	300	228	284	1587	25099	359	231	1826	2662	8626	60	312	147	102	826	68	0	6117	2116	1
с	9711	7692	203	10206	53	2081	18390	77	20	7101	0	13946	12872	1960	2502	9469	13177	1186	2211	36757	3938	115	732	241	564	187	1	109	6222	6783	0
т	30944	16293	78	15858	37	685	18960	5	17	12623	0	1162	376	166	1962	37604	655	7843	4975	422	3140	36	62	1365	369	33	77	4	7910	18778	5
у	25095	116	2057	1666	2779	11788	725	3547	867	1057	458	3270	1243	3070	540	117	2879	715	7382	5171	27	84	1699	74	2235	2734	1150	0	0	0	17
ф	684	2021	0	9	2	4	675	0	4	997	0	11	69	14	35	399	4	305	89	21	199	110	3	3	0	2	0	0	88	50	0
х	19455	2184	64	735	59	95	799	16	40	2063	0	126	764	89	340	5276	191	1809	352	167	685	5	12	12	36	15	0	0	0	1	6
ц	1323	8547	3	135	13	3	3128	0	1	134	0	4	3	6	3	594	16	15	6	10	664	8	0	3	1	0	0	0	896	0	1
ч	600	4386	2	1	1	1	9019	0	4	4743	0	130	67	40	1452	188	3	470	0	9758	949	0	0	9	0	287	0	0	0	859	0
ш	413	3509	21	29	2	4	7170	0	0	6665	0	132	2175	12	839	410	12	1	9	24	844	6	9	81	0	13	0	0	0	2811	0
щ	39	1716	0	65	0	0	5060	0	0	4040	0	0	0	0	191	3	0	23	2	1	612	0	0	0	1	0	0	0	0	109	0
ъ	0	0	0	0	0	0	137	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ы	21901	34	203	2652	79	364	4035	83	79	364	4588	664	5063	2986	6900	70	261	623	2013	1311	15	1	4302	7	534	2487	142	0	0	0	4
ь	34643	70	77	273	75	419	817	8	207	757	0	957	8	1436	2365	121	175	21	2756	804	25	12	5	339	32	636	67	0	0	0	4
э	1	0	0	2	5	2	0	0	0	0	2	0	23	4	4	0	2	9	1	3175	0	0	3	4	0	3	0	0	0	0	0
ю	14044	62	1031	110	119	1383	42	83	110	607	4	70	17	37	357	95	116	19	329	2227	28	2	3	6	277	2	2115	0	0	0	9
я	37969	118	120	1147	233	485	806	262	1283	1005	354	1295	2406	2053	2987	186	241	251	914	6939	44	11	663	381	868	37	2639	0	0	0	10
ё	82	0	0	0	1	0	0	0	0	0	0	0	1	0	2	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0



(на один екран все одно не вмістилось вибачте)

І для біграм без перетину:

bigram_without_overlap_matrix.txt — Блокнот

Файл Правка Формат Вид Справка

	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э
	0	3784	13007	31266	11464	14170	9982	4615	7387	40926	0	13713	3856	12780	26648	19019	27305	5633	33160	15129	7092	675	2641	2394	7446	825	69	0	0	0	1494
а	27668	861	1353	7709	1996	4024	2422	1781	5758	2488	1642	6438	12448	7161	5972	451	1375	6593	5635	5851	338	635	2094	43	1230	2190	436	0	0	0	7
б	521	873	108	114	4	19	3712	5	5	1501	0	16	2562	56	493	7160	0	2866	84	9	3092	0	16	6	3	12	136	143	5230	84	0
в	13261	9593	18	183	9	446	10761	0	802	6133	0	98	1507	254	1061	19101	168	1761	5114	185	1760	19	153	95	156	1131	2	0	4629	722	0
г	1592	2202	6	21	14	2171	560	0	6	1637	0	16	1137	0	943	23436	10	1360	24	7	850	2	2	0	28	19	0	0	0	0	0
д	3636	10526	20	1239	8	88	11760	1	31	5110	0	124	1583	118	3666	7007	72	1610	460	150	4409	0	22	1139	12	142	37	10	1507	2309	2
е	31024	147	4441	4093	9559	5338	2461	1742	1694	1152	4648	2217	11456	10474	13534	305	1372	9315	9406	10045	142	527	1513	613	2001	1729	689	0	0	0	10
ж	125	1336	54	5	53	1993	5951	192	0	3620	0	41	3	2	796	38	0	96	3	0	406	0	0	0	36	0	0	0	0	74	0
з	2946	8017	483	1696	298	1042	2129	23	294	503	0	86	970	312	2139	1030	6	1953	3	5	484	2	0	4	4	1	0	17	780	358	0
и	54722	747	2274	3383	602	3490	5692	413	5943	2982	2337	3900	11385	7413	5249	1364	621	1617	7052	8612	902	335	6552	1440	734	1179	835	0	0	0	11
й	11027	21	24	75	15	838	26	3	9	324	0	48	6	44	451	43	57	5	1172	949	17	2	6	69	11	207	0	0	0	0	4
к	7767	9958	10	231	3	4	533	168	1	3833	0	68	961	3	465	12417	15	2686	122	1101	1508	2	30	1	3	13	0	0	0	0	0
л	13105	9416	82	49	126	76	8458	551	11	15090	0	208	156	82	713	9219	59	5	1308	78	2267	14	47	4	152	13	6	0	1023	3471	3
м	14267	3212	29	150	46	43	7442	6	26	6284	0	180	2043	165	2904	6680	125	215	441	81	5327	26	7	20	40	4	42	0	1518	344	4
н	6901	18861	2	13	204	63	16390	2	16	17387	0	127	1	19	3764	11992	18	29	447	140	2306	8	21	375	156	2	190	0	5286	1641	0
о	31847	534	6165	14546	9846	14258	5744	3151	2752	5001	4498	3400	7348	7755	9747	216	1693	11061	15110	15055	173	412	629	211	1853	1543	309	0	0	0	45
п	50	2031	0	0	0	0	2527	0	0	1616	0	86	1444	0	126	19023	56	11905	86	347	1541	14	0	22	15	60	6	0	264	132	0
р	985	13855	186	466	265	1088	11533	374	211	8950	0	141	117	137	775	12562	186	105	922	1312	4233	26	178	71	50	404	26	0	3077	1038	1
с	4823	3894	95	5175	31	1024	9266	34	11	3525	0	6987	6410	972	1265	4756	6550	607	1122	18518	1906	56	353	109	279	84	0	55	3004	3276	0
т	15462	8206	40	7883	19	353	9352	2	8	6311	0	573	174	83	985	18664	325	3898	2446	201	1557	23	33	682	177	18	36	0	3978	9276	2
у	12547	61	1007	828	1379	5987	381	1799	438	559	245	1634	635	1524	286	63	1482	373	3702	2609	14	41	863	36	1085	1408	621	0	0	0	7
ф	316	1048	0	6	2	4	322	0	2	525	0	5	29	5	14	205	3	144	52	10	100	54	1	0	0	1	0	0	45	25	0
х	9681	1057	30	371	35	52	405	6	23	1075	0	56	382	42	161	2690	90	927	173	80	347	3	8	8	14	11	0	0	0	0	3
ц	685	4222	0	78	6	1	1546	0	0	70	0	2	2	3	1	311	11	6	4	4	334	4	0	1	0	0	0	0	460	0	1
ч	298	2166	0	0	0	0	4536	0	3	2419	0	69	33	16	753	97	3	216	0	4967	488	0	0	5	0	143	0	0	0	438	0
ш	191	1710	11	17	1	2	3593	0	0	3255	0	56	1080	4	437	212	9	0	8	10	407	3	5	40	0	7	0	0	0	1393	0
щ	17	846	0	42	0	0	2562	0	0	2008	0	0	0	0	88	3	0	13	0	1	299	0	0	0	0	0	0	0	0	49	0
ъ	0	0	0	0	0	0	67	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ы	11027	22	97	1325	36	188	2039	44	33	182	2279	312	2479	1485	3536	37	131	314	997	653	4	1	2160	5	248	1268	74	0	0	0	0
ь	17530	34	33	141	37	219	409	5	107	409	0	486	3	775	1194	62	87	11	1395	414	13	6	3	168	12	307	30	0	0	0	3
э	1	0	0	0	3	2	0	0	0	0	1	0	7	2	2	0	1	3	1	1601	0	0	3	3	0	1	0	0	0	0	0
ю	7023	31	484	51	60	687	18	43	54	292	0	39	9	19	189	47	59	5	163	1117	14	1	0	3	126	1	1062	0	0	0	5
я	18996	51	67	579	117	250	399	125	639	495	171	646	1223	1037	1515	89	129	128	454	3446	26	7	339	198	448	17	1328	0	0	0	7
ё	45	0	0	0	0	0	0	0	0	0	0	0	1	0	2	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

э	ю	я	ё
1494	203	3408	0
7	1780	1293	1
0	0	1514	0
0	0	1597	0
0	0	1	0
2	5	931	0
10	1116	727	0
0	0	0	0
0	16	595	0
11	779	3410	0
4	1	12	0
0	0	0	0
3	2223	3437	0
4	0	1126	0
0	240	3010	0
45	2518	1014	0
0	0	593	0
1	452	1653	0
0	146	6526	38
2	2	992	0
7	1343	96	0
0	0	19	0
3	1	12	0
1	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	163	0
0	8	6	0
3	635	878	0
0	0	0	0
5	52	11	0
7	254	132	0
0	0	0	0

Таблиці частотності отримали, тож видаляємо функцію для друку матриць, щоб на це не витрачався час при кожному запуску.

Тепер нам треба знайти за означенням H1 та H2, тобто ентропію для окремих літер і для біграм у тексті, але у формулі імовірності замінити відповідними частотами, що ми тільки що знайшли.

Ентропія тексту обчислюється як сума ентропій для кожної літери (H1) або біграми (H2) окремо. Тобто ми обчислюємо ентропію для кожного можливого символу або біграми в тексті і потім сумуємо ці значення. Це дає

загальну ентропію тексту і вказує на ступінь невизначеності або «непередбачуваності» тексту.

Загальні формули для ентропії:

$$H1(X) = -\sum_{i=1}^{n} P(x_i) \cdot \log_2(P(x_i))$$

$$H2(X) = -\sum_{i=1}^{n} P(bigram_i) \cdot \log_2(P(bigram_i))$$

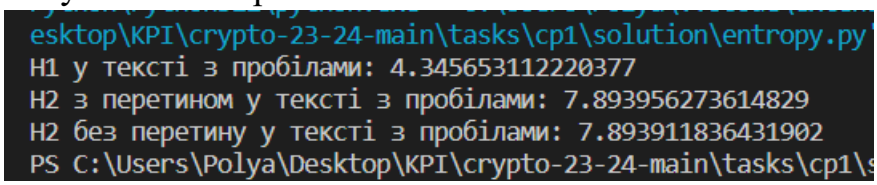
Ну а у нашому випадку, ми замість P(xi) і P(bigram) ставитимо обраховане значення частоти для цієї літери або біграми.

Пройшла ніч і я зрозуміла, щоб не порушувати логіку формули ентропії ми можем розрахувати ці ймовірності, просто поділивши частотність літер/біграм на загальну кількість літер/біграм. Бо просто замінивши значення P на частотність навряд вийде розрахувати ентропію за означенням, бо ми викинемо логічну частину формули. Тому мій розрахунок виглядає так:

```
# Підрахунок ентропії для літер (H1)
entropy_letter = 0.0
total_letters = len(cleaned_text)

for count in sorted_letter_counts.values():
    probability = count / total_letters
    entropy_letter -= probability * math.log2(probability)

print(«H1 у тексті з пробілами:», entropy_letter)

# Ентропія для біграм з перетином (H2)
entropy_bigram_with_overlap = 0.0
total_bigrams_with_overlap = len(cleaned_text) - 1

for count in sorted_bigram_with_overlap_counts.values():
    probability = count / total_bigrams_with_overlap
    entropy_bigram_with_overlap -= probability * math.log2(probability)

print(«H2 з перетином у тексті з пробілами:», entropy_bigram_with_overlap)

# Ентропія для біграм без перетину (H2)
entropy_bigram_without_overlap = 0.0
```

```
total_bigrams_without_overlap = len(cleaned_text) // 2


for count in sorted_bigram_without_overlap_counts.values():
    probability = count / total_bigrams_without_overlap
    entropy_bigram_without_overlap -= probability * math.log2(probability)


print(«H2 без перетину у тексті з пробілами:»,
entropy_bigram_without_overlap)
```

Запустимо і збережемо значення:

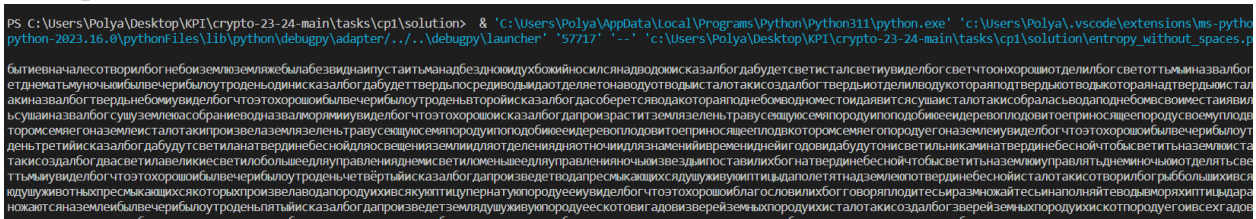


Тепер виконаємо ті ж самі розрахунки, але попередньо вилучивши всі пробіли з тексту. Для цього створю окремий файл.

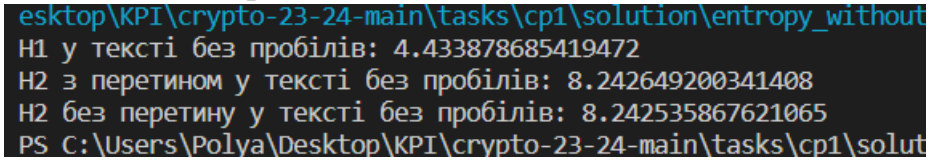
Код виглядатиме так само, просто з трошки зміненим кодом фільтрації тексту

```
with open(file_path, «r») as file:
    cleaned_text = ‘’
    for line in file:
        line = re.sub(r’[^a-яА-ЯёЁ]’, ‘’, line)  # Вилучення всіх символів окрім літер
        cleaned_text += line.lower()
print (cleaned_text)
```

Тепер наш текст виглядатиме отак

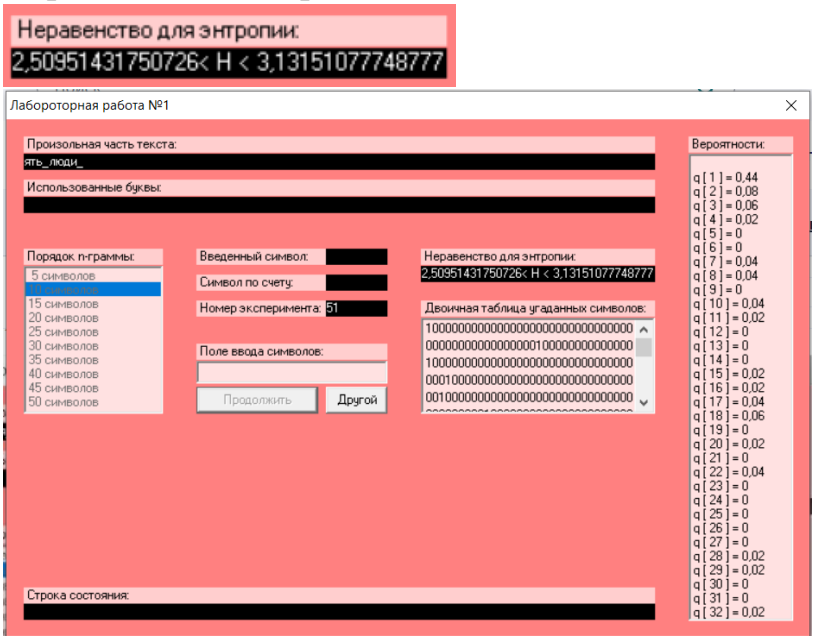


Та значення ентропії:



2. За допомогою програми CoolPinkProgram оцінити значення (10)H , (20)H, (30)H .
    - Оцінюємо значення для 10 символів:

Нерівність для ентропії:

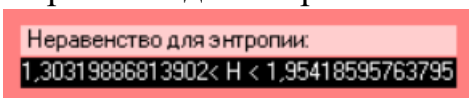


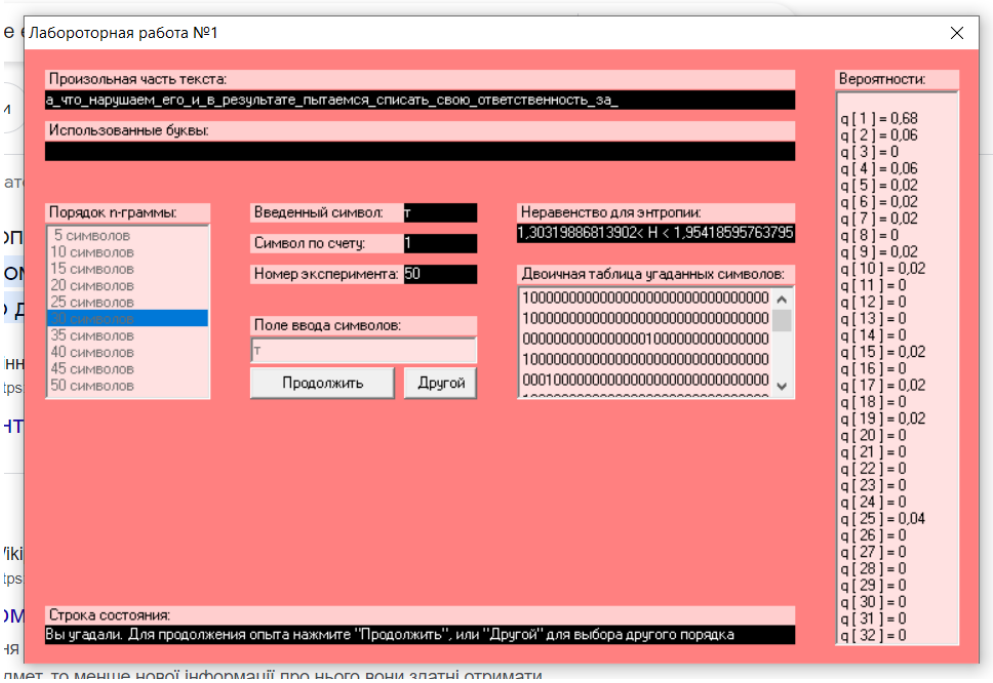
- Для 20 символів:

Нерівність для ентропії:

- Для 30 символів:

Нерівність для ентропії:





За моїми спостереженнями, чим більше символів ми бачимо – тим менше загальне значення ентропії = менша «хаотичність» і невизначеність тексту.

3. Використовуючи отримані значення ентропії, оцінити надлишковість російської мови в різних моделях джерела.

Надлишковість джерела відкритого тексту (мови) розраховується за формулою:

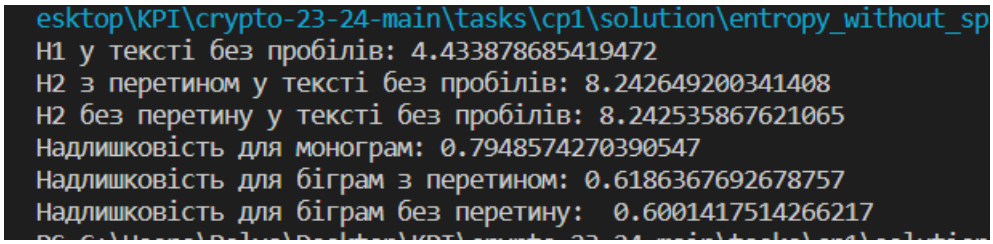
$R = 1 - \frac{H_\infty}{H_0}$, де ми прийматимемо за H(inf) наші розраховані значення

```
r_for_monograms = 1 - (entropy_letter/math.log2(total_letters))

r_for_bigram_with_overlap = 1 -
(entropy_bigram_with_overlap/math.log2(total_bigrams_with_overlap))

r_for_bigram_without_overlap = 1 -
(entropy_bigram_without_overlap/math.log2(total_bigrams_without_overlap))


print(«Надлишковість для монограм:»,r_for_monograms)

print(«Надлишковість для біграм з перетином:»,r_for_bigram_with_overlap)

print(«Надлишковість для біграм без перетину: «,r_for_bigram_without_overlap)
```

Результат:

```
H1 у тексті з пробілами: 4.345653112220377
H2 з перетином у тексті з пробілами: 7.893956273614829
H2 без перетину у тексті з пробілами: 7.893911836431902
Надлишковість для монограм: 0.8013513691606089
Надлишковість для біграм з перетином: 0.6391512180027582
Надлишковість для біграм без перетину:  0.6218680704424144
```

Тепер розрахуємо це значення для джерела без пробілів.



Висновки

Під час виконання цієї лабораторної роботи я не тільки покращила свої навички у програмуванні, а й зрозуміла, чому значення ентропії і надлишковості є важливими у кібербзпеці, а не нас просто вирішили задовбати вищою математикою.

Як ми визначили, ентропія - це міра невизначеності чи непередбачуваності. У контексті тексту ентропія вимірює, наскільки випадковим є розподіл символів чи біграм у тексті (у нашому випадку, можна так то рахувати триграми і т.д.) . Чим більше ентропія, тим більше невизначеність та непередбачуваність.

Різниця ентропій для тексту з пробілами та без пробілів може бути пояснена тим, що пробіли - це також символи, котрі вважаються частиною алфавіту, і вони додаються до загальної кількості символів у тексті. Якщо текст має багато пробілів, то розподіл символів буде менш випадковим, оскільки пробіли будуть домінувати серед інших символів. Таким чином, ентропія тексту з пробілами буде нижчою порівняно з текстом без пробілів.

Надлишковість - це міра того, наскільки розподіл символів чи біграм у тексті відрізняється від рівномірного розподілу. Висока надлишковість вказує на невизначеність мови та може бути використана для криптографічних цілей, так як важко передбачити наступний символ. Висока надлишковість джерела мови вказує на те, що мова має багато невизначеності та непередбачуваних залежностей між символами чи біграмами. Це означає, що для побудови криптосистеми буде складно передбачити, які символи чи біграми випадуть наступними, що робить шифрування більш надійним перед атакою.

В криптографії, ентропія та надлишковість важливі, оскільки вони впливають на стійкість криптосистем. Висока ентропія тексту робить його складнішим для аналізу і підірвання. Надлишковість джерела мови може використовуватися для побудови надійних шифрів, де важко передбачити, який символ або біграм вийде наступним.

Тож ми розібрались з базовими поняттями та принципами роботи криптографії та надійності шифрування, було весело 😊