Problem 1: Frequency Analysis

(a) (50 pts) Submit a soft copy of your program via Canvas. Please write your own program! Do not copy other people's programs.

50 pts

Notes

The code file below does frequency analysis of the text and cleanly outputs the frequency of each character and frequency of n-grams where $n = 2, 3$. These n-grams ignore words that are too small to fit them (ie. a 4-gram will ignore the word "the" because its 3 letters). I've used the argparse library just so its easier to interact with and the program can handle text files or directly pasted text.

Code for Part (a)

```
from argparse import ArgumentParser
from collections import Counter


def print_freq(title: str, data: dict[str, int], top: int = 30) ->
    None:
    all_items = sorted(data.items(), key=lambda kv: (-kv[1], kv[0]))
        # Sort from greatest to least, weird ahh lambda but it works
    if top is not None and top > 0:  # Quick way for us to ignore "
        top"
        shown_items = all_items[:top]
    else:
        shown_items = all_items
        top = len(shown_items)

    showing_counter = f" (showing {len(shown_items)}/{len(all_items)
        })"
    print(f"\n{title}{showing_counter}")
    print("-" * (len(title) + len(showing_counter)))

    cols = 3
    rows = (len(shown_items) + cols - 1) // cols
    items = []
    for k, v in shown_items:
        items.append(f"{k}:{v}")

    # Reorganize into columns and print (added this cause it was
        spamming my terminal)
```

```
    for r in range(rows):
        line = []
        for c in range(cols):
            idx = r + c * rows
            if idx < len(items):
                line.append(items[idx])
        print("\t".join(line))


def freq_counter(text: str) -> dict[str, int]:
    text = "".join([c for c in text if c.isalpha()])
    letter_frequency = Counter(text)

    return dict(letter_frequency)


def ngram_counter(text: str, n: int) -> dict[str, int]:
    ngrams = [text[i : i + n] for i in range(len(text) - n + 1)]
    ngrams = [ngram for ngram in ngrams if " " not in ngram] # Ignore ngrams with spaces
    ngram_frequency = Counter(ngrams)

    return dict(ngram_frequency)


def get_argparser() -> ArgumentParser:
    parser = ArgumentParser(description="Frequency Counter")
    group = parser.add_mutually_exclusive_group(required=True)
    group.add_argument("--text", type=str, help="Text to analyze directly")
    group.add_argument("--file", type=str, help="Path to a .txt file to analyze")
    parser.add_argument(
        "--ngram", type=int, help="Max N-gram size for analysis", default=3
    )
    return parser


def main():
    parser = get_argparser()
    args = parser.parse_args()

```

```
63      if args.file:
64          with open(args.file, "r", encoding="utf-8") as f:
65              text = f.read()
66      else:
67          text = args.text
68
69      text = text.lower()
70      text = text.replace("\n", " ").replace("\r", " ").replace("\t",
            " ")
71
72      result = freq_counter(text)
73      print_freq("Character Frequency", result, top=-1)
74
75      for i in range(2, args.ngram + 1):
76          ngram_result = ngram_counter(text, i)
77          print_freq(f"{i}-gram Frequency", ngram_result, top=30)
78
79
80  if __name__ == "__main__":
81      main()
```

(b) (25 pts) Output of your program run against at least 2 different texts of sufficient length. Make sure that the texts are sufficiently long enough for frequency analysis.

25 pts

Hamlet Act 3 Scene 1 Freq Analysis

Character Frequency (showing 26/26)

e:136	d:57	p:13
t:95	u:50	b:12
n:89	l:42	k:8
o:85	m:35	z:7
a:76	c:33	q:5
i:74	f:26	v:4
s:74	g:26	j:2
r:72	y:23	x:1
h:58	w:17	

2-gram Frequency (showing 30/238)

an:24	on:14	of:10
en:23	st:14	to:10
he:22	os:12	ee:9
th:19	de:11	nc:9
hi:18	im:11	te:9
er:16	ou:11	ar:8
in:15	ra:11	cr:8
nt:15	ro:11	di:8
se:15	es:10	ea:8
nd:14	is:10	ge:8

3-gram Frequency (showing 30/397)

and:11	ran:7	den:4
him:10	ros:7	ear:4
the:10	sen:7	ens:4
cra:8	ent:6	ern:4
ose:8	ing:6	ert:4
ant:7	ter:6	ger:4
enc:7	ius:5	gui:4
his:7	tru:5	hea:4
ncr:7	but:4	ild:4
ntz:7	con:4	ion:4

Hamlet Act 4 Scene 1 Freq Analysis

Character Frequency (showing 26/26)

e:206	d:80	p:21
t:137	l:74	y:19
s:121	u:71	k:16
n:120	w:45	v:12
a:114	m:38	q:4
o:112	g:34	z:4
i:106	c:29	x:3
h:105	f:22	j:2
r:93	b:21	

2-gram Frequency (showing 30/266)

th:41	ou:20	on:13
er:36	se:20	te:13
he:34	de:18	to:13
an:29	st:17	us:13
en:29	ra:16	es:12
in:24	wh:16	il:12
nd:24	ee:15	la:12
hi:22	at:14	ll:12
is:22	nt:14	rt:12
ha:20	it:13	ud:12

3-gram Frequency (showing 30/512)

the:22	rtr:7	ius:6
and:16	rud:7	ran:6
his:13	thi:7	ste:6
hat:9	tru:7	unt:6
ing:9	ude:7	wha:6
ter:9	you:7	all:5
een:8	den:6	aud:5
ert:8	enc:6	cla:5
ger:7	ens:6	diu:5
ose:7	her:6	ern:5

(c) (5 pts) Compare the 2 sets of frequencies you produced for part (b). Are they similar or different? Explain why they are similar or different.

5 pts

Comparison & Explanation

Act 4 Scene 1 is clearly longer as the total number of letters is higher. However they do share a lot of similarities, for one their top two most used characters are the same as well as their last two most unused characters. They also both use "and" & "the" as their most used words and have a lot of male pronouns (him, his, etc).

(d) (20 pts) Using your frequency analysis results, decrypt the ciphertext given below.

20 pts

Ciphertext

bt jpx rmlx pcuv amlx icvjp ibtwxvr ci m lmt'r pmtn, mtn yvcjx cdxv mwmbtrj jpx amtngxrjbah uqct jpx qgmrjxv ci jpx ymgg ci jpx hbtw'r qmgmax; mtn jpx hbtw rmy jpx qmvj ci jpx pmtn jpmj yvcjx. jpxt jpx hbtw'r acutjxtmtax ymr apmtwxn, mtn pbr jpcuwpjr jvcufgxn pbl, rc jpmj jpx scbtjr ci pbr gcbtr yxvx gccrxn, mtn pbr htxxr rlcjx ctx mwmbtrj mtcjpxv. jpx hbtw avbxn mgcun jc fvbtw bt jpx mrjvcgcwxvr, jpx apmgnxmtr, mtn jpx rccjprmexvr. mtn jpx hbtw rqmhx, mtn rmbn jc jpx ybrx lxt ci fmfegct, ypcrcxdxv rpmgg vxmn jpbr yvbjbtw, mtn rpcy lx jpx btjxvqvxjmjbct jpxvxci, rpmgg fx agcjpxn ybjp ramvgxj, mtn pmdx m apmbt ci wcgn mfcuj pbr txah, mtn rpmgg fx jpx jpbvn vugxv bt jpx hbtwncl. jpxt amlx bt mgg jpx hbtw'r ybrx lxt; fuj jpxe acugn tcj vxmn jpx yvbjbtw, tcv lmhx htcyt jc jpx hbtw jpx btjxvqvxjmjbct jpxvxci. jpxt ymr hbtw fxgrpmoomv wvxmjge jvcufgxn, mtn pbr acutjxtmtax ymr apmtwxn bt pbl, mtn pbr gcvnr yxvx mrjctbrpxn. tcy jpx kuxxt, fe vxmrct ci jpx ycvnr ci jpx hbtw mtn pbr gcvnr, amlx btjc jpx fmtkuxj pcurx; mtn jpx kuxxt rqmhx mtn rmbn, c hbtw, gbdx icvxdxv; gxj tcj jpe jpcuwpjr jvcufgx jpxx, tcv gxj jpe acutjxtmtax fx apmtwxn; jpxvx br m lmt bt jpe hbtwncl, bt ypcl br jpx rqbvbj ci jpx pcge wcnr; mtn bt jpx nmer ci jpe ybrncl ci jpx wcnr, ymr icutn bt pbl; ypcl jpx hbtw txfuapmntxoomv jpe imjpxv, jpx hbtw, b rme, jpe imjpxv, lmnx lmrjxv ci jpx lmwbabmtr, mrjvcgcwxvr, apmgnxmtr, mtn rccjprmexvr; icvmrluap mr mt xzaxggxtj rqbvbj, mtn htcygxnwx, mtn utnxvrjmtnbtw, btjxvqvxjbtw ci nvxmlr, mtn rpcybtw ci pmvn rxtjxtaxr, mtn nbrrcgdbtw ci ncufjr, yxvx icutn bt jpx rmlx nmtbxg, ypcl jpx hbtw tmlxn fxgjxrpmoomv; tcy gxj nmtbxg fx amggxn, mtn px ybgg rpcy jpx btjxvqvxjmjbct.

Decrypted Plaintext

in the same hour came forth fingers of a man's hand, and wrote over against the candlestick upon the plaster of the wall of the king's palace; and the king saw the part of the hand that wrote. then the king's countenance was changed, and his thoughts troubled him, so that the joints of his loins were loosed, and his knees smote one against another. the king cried aloud to bring in the astrologers, the chaldeans, and the soothsayers. and the king spake, and said to the wise men of babylon, whosoever shall read this writing, and show me the interpretation thereof, shall be clothed with scarlet, and have a chain of gold about his neck, and shall be the third ruler in the kingdom. then came in all the king's wise men; but they could not read the writing, nor make known to the king the interpretation thereof. then was king belshaxxar greatly troubled, and his countenance was changed in him, and his lords were astonished. now the queen, by reason of the words of the king and his lords, came into the banquet house; and the queen spake and said, o king, live forever; let not thy thoughts trouble thee, nor let thy countenance be changed; there is a man in thy kingdom, in whom is the spirit of the holy gods; and in the days of thy wisdom of the gods, was found in him; whom the king nebuchadnexxar thy father, the king, i say, thy father, made master of the magicians, astrologers, chaldeans, and soothsayers; forasmuch as an ezcellent spirit, and knowledge, and understanding, interpreting of dreams, and showing of hard sentences, and dissolving of doubts, were found in the same daniel, whom the king named belteshaxxar; now let daniel be called, and he will show the interpretation.

Method / Key / Mapping

First I checked the top two 3-grams because "the" & "and" are the two most common words and substituted them in, that gave a better idea of the rest of the words now that I had some of the most common letters revealed. Then I got the most common 2-gram (was actually the 3rd most common but the first two were just parts of "the") as "in" which revealed more. Then I got "of" as well basically following the same idea as "in" at which point it was pretty much guess and check from there on out. I just kept following the n-grams (i upped my frequency checker to do 4-grams as well) and the most common words in english which I just googled to get and eventually figured it out by guess and checking with a decrypter script I made.