实验一：破解维吉尼亚密码

谢子洋 202100460116
冯大玮 202100460126
程雨森 202100460090
刘志 202122460178

2023 年 03 月 03 日星期五

目录

1 组员分工

姓名	任务
谢子洋	代码实现, 实验过程
冯大玮	论文细节完善
程雨森	论文细节完善
刘志	论文主体书写与排版

2 前置知识介绍

2.1 重合指数

定义：设 $x_1x_2\dots x_n$ 是含有 n 个字母的串，则在 x 中随机选择两个元素且这两个元素相同的概率为

$$I_c(x)=\frac{\sum_{i=0}^{n}f_i(f_i-1)}{n(n-1)}$$

其中 fi 为 26 个字母中第 i 个字母在 x 中出现的次数

2.2 自然语言重合指数

利用英文字母频率表

$$I_c(x)\approx\sum_{i=0}^{25}p_i^2\approx 0.065$$

注意：单表代换不改变该值，即用相同密钥字加密的密文应服从该值

2.3 重合互指数

定义：设 $x=x_1,x_2,\dots x_n,y=y_1,y_2,\dots,y_n$ 分别为长度为 n 与 n' 的串，其重合指数为从 x 与 y 中分别随机选出一个元素且两个元素相同的概率

$$MI_c(x,y)=\frac{\sum_{i=0}^{25}f_if_i'}{nn'}$$

3 Vigenere 密码的原理

3.1 Vigenere 密码的数学原理

维吉尼亚密码是使用一系列凯撒密码组成密码字母表的加密算法，属于多表密码的一种简单形式。在凯撒密码中，每一个字母会有一定的偏移量变成另外一个字母，而维吉尼亚密码就是有多个偏移量不同的凯撒密码组成。Vigengere 密码使用一个字符串作为密钥，第一个密钥字母加密明文的第一个字母，第二个密钥字母加密明文的第二个字母，等所有密钥字母使用完后，密钥又再循环使用，以此得到密文。用数字 0-25 代替字母 A-Z，维吉尼亚密码加密算法为：

$$C_i \equiv M_i + K_i(mod26)$$

解密算法为：

$$M_i \equiv C_i - K_i(mod26)$$

例：P = vigenere K = key

P_i	v	i	g	e	n	e	r	e
K_i	k	e	y	k	e	y	k	e
C_i	f	m	e	o	r	c	a	i

3.2 Vigenere 密码的保密性分析

维吉尼亚密码的密钥空间大小为 26^m，所以即使 m 的值很小，使用穷尽密钥搜索方法也需要很长的时间。例如，当 m=5 时，密钥空间大小超过 $1.1 * 10^7$，这样的密钥量已经超出了使用手算进行穷尽搜索的能力范围 (当然使用计算机另当别论)。一般来说，这种多表代换密码比单表代换密码更为为安全一些。

4 Vigenere 密码的破解原理

4.1 确定密钥长度

4.1.1 Kasiski 测试法

原理：密文中出现两个相同字母序列，它们所对应的明文字母相同的可能性很大。这样的两个密文字母组之间的距离可能为密钥长度的整数倍。

4.1.2 重合指数法

原理：自然语言（英语）的重合指数约为 0.065，且单表代换不会改变该值.
猜测密钥长度：假设密钥长度为 d，提取相同密钥字加密的密文，测试其重合指数。如果猜测正确，则重合指数值接近 0.065；否则，字符串表现得更为随机，一般在 0.038（1/26）～0.065 之间。

4.2 确定密钥组

4.2.1 确定密钥字相对位移

考虑不同密钥字加密后密文串的重合互指数，我们有

$$MI_c(C_i, C_j) \approx \sum_{l=0}^{25} p_{l-k_i} p_{l-k_j}$$

上式等价于

$$MI_c(x, y) = \frac{\sum_{i=0}^{25} f_{i,t} f_{j,t-s}}{n_i n_j}$$

若 s 猜对，则该值应接近于 0.065，这意味着找到了不同密钥字加密的相同的明文字母，即找到了密钥字之间的相对位移

4.2.2 直接确定密钥组

特别的, 当计算重合互指数的两个字符串中有一个为自然语言, 则重合互指数

$$M_g = \sum_{i=0}^{25} \frac{p_i f_{i+g}}{n'}$$

当 g 为加密字符串中某一组的密钥时, 重合互指数 M_g 应接近 0.065.

4.3 穷举搜索密钥字

确定密钥字之间关系式基础上，穷举搜索 26 种可能。分别计算所得明文与自然语言的重合互指数, 最为接近 0.065 即为所求明文。

5 Vigenere 破解过程实例

5.1 确定密钥长度

5.1.1 Kasiiki 测试法

取子字符串长度为 4，找出所有长度为 4 的重合子字符串位置, 根据重合字符串之间距离计算其共同的最大公因子。结果展示如下。通过观察不难发现, 计算出的大部分最大公因子都是 7 的倍数, 因此密钥长度大概率为 7。

此处展示部分字符子串及其位置, 以及据此求出的最大公因子.

表 1: 重合子串列表

重合字符串	最大公因子	各字符串位置
nzuq	63	[18, 81]
zuqi	7	[19, 82, 103, 194]
uqig	7	[20, 83, 104, 195]
qigs	7	[21, 84, 105, 196]
igsc	7	[22, 85, 106, 197]
gscv	7	[23, 86, 107, 198]
scvx	175	[24, 199]
scvf	21	[87, 108]
cvfj	21	[88, 109]
vfjw	21	[89, 110]
fjwq	21	[90, 111]
honb	21	[129, 150]
onbm	21	[130, 151]
nbmb	21	[131, 152]
bmbv	21	[132, 153]
ejif	196	[234, 430]
esvz	28	[260, 288]
svzi	28	[261, 289]
vzif	28	[262, 290]
zifu	28	[263, 291]
futz	28	[265, 293]
utzf	28	[266, 294]

5.1.2 重合指数法

计算出密钥长度分别为 1-20 时所对应的重合指数, 结果如下表。观察下表不难发现密钥长度为 7 时重合指数最接近 0.065, 故密钥长度最可能为 7。两种方法相互印证。

表 2: 重合互指数表

密钥长度	重合互指数
1	0.044870830725922115
2	0.04340605818742676
3	0.04255186879826416
4	0.042699137540801575
5	0.04242518095769651
6	0.042226726851218144
7	0.07352056602641621
8	0.04093671729547781
9	0.039687319676896246
10	0.041885499168623955
11	0.0368481140935599
12	0.040375473573854126
13	0.038112832720675865
14	0.07563691076715055
15	0.03836266700156686
16	0.039139598235912135
17	0.0386876497987609
18	0.03905763751917598
19	0.03971014492753624
20	0.039611617116343006

5.2 计算密钥组与明文

5.2.1 拟重合指数法

根据密钥，将密文分成七个子字符串，对七个子字符串分别运用遍历法，找出密钥为 0-25 时每个子字符串与自然语言的重合互指数，即

$$M_g = \sum_{i=0}^{25} \frac{p_i f_{i+g}}{n'}$$

其中结果最接近 0.065 的数大概率为该子字符串的加密密钥，七个数字组成的向量即为密钥组，并根据此密钥进行解密，即可得到明文。基于此原理得到密钥组和明文如下：

[20, 8, 14, 25, 21, 17, 1]

Itwasthebestoftimesitwastheworstoftimesitwastheageofwisdomitwastheageoffoolishnessitwastheepochofbeliefitwastheepochofincredulityitwastheseasonoflightitwastheseasonofdarknessitwasthespringofhopeitwasthewinterofdespairwehadeverythingbeforeuswehadnothingbeforeuswewereallgoingdirecttoheavenwewereallgoingdirecttheotherwayinshorttheperiodwassofarlikethepresentperiodthatsomeofitsnoisiestauthoritiesinsistedonitsbeingreceivedforgoodorforevilinthesuperlativedegreeofcomparisononly

注：此方法得出结果唯一, 无需再次选择。

5.2.2 计算密钥相对位移法

分别对各密钥字之间的相对位移 s 从 0-25 进行遍历，分别计算其重合互指数，其中结果最接近 0.065 的 s 为两密钥字之间的相对位移。所得结果如下：

K0=K1+12; K0=K2+6 ; K0=K3+21;
K0=K4+25; K0=K5+3 ; K0=K6+19;

遍历 K0 的 26 种可能, 得到 26 个密钥组, 解密后得到 26 个明文,26 个密钥组及其对应的 26 种明文. 结果如上文所示。

5.3 检验明文是否为自然语言

5.3.1 人工检验

此处略

5.3.2 通过计算重合互指数检验

对 26 种情况分别计算明文与自然语言的重合互指数, 其中最接近 0.065 的情况即为所求.

表 3: 重合互指数

初始偏移量	重合互指数
1	0.03160792842105263
2	0.033967107368421054
3	0.03253810947368422
4	0.027217414736842107
5	0.036804
6	0.04609724210526316
7	0.0383367957894737
8	0.040161301052631576
9	0.04398917894736842
10	0.04648454315789472
11	0.03598247578947369
12	0.03504650526315788
13	0.03571088421052632
14	0.035916901052631595
15	0.03016953263157894
16	0.03011535157894737
17	0.04288002105263158
18	0.035032968421052635
19	0.027161018947368422
20	0.03709063157894737
21	0.06811093894736844
22	0.043774400000000005
23	0.03040333052631579
24	0.03464564210526317
25	0.0493041852631579
26	0.03367559157894737

根据表 3, 我们发现当 k0 等于 20 时, 所求明文特征与自然语言最接近, 此时根据 5.2.2 可知密钥组为: [20, 8, 14, 25, 21, 17, 1], 据此得明文如下：

Itwasthebestoftimesitwastheworstoftimesitwastheageofwisdomitwastheageoffoolishnessitwastheepochofbeliefitwastheepochofincredulityitwastheseasonoflightitwastheseasonofdarknessitwasthespringofhopeitwasthewinterofdespairwehadeverythingbeforeuswehadnothingbeforeuswewereallgoingdirecttoheavenwewereallgoingdirecttheotherwayinshorttheperiodwassofarlikethepresentperiodthatsomeofitsnoisiestauthoritiesinsistedonitsbeingreceivedforgoodorforevilinthesuperlativedegreeofcomparisononly

6 实验结果与分析

根据以上一系列分析可知密钥组为: [20, 8, 14, 25, 21, 17, 1]，结果符合一般规律

7 参考文献

[1]William Stallings. 密码编码学与网络安全——原理与实践（第七版）[M]. 背景：电子工业出版社，2017

8 附录

密文内容：

’cbkznkiyjsrofgnqadnzuqigscvxizgsjwucusrdkxuahgzrhywtvdjeiuwsrrt npszbvpzncngztbvsrnzuqigscvfjwqgjwcytwdazuqigscvfjwqgjwjhkfdylmcbmhonbmbvdnvbmw bnacjaphhonbmbvdnvbmwbnaublsbdnjjneoroyfmxfhixpzpcozzuqigscvxcvhdmfgxmgovz sqmvzyvwyzmsczoajsejifoakdcrehwhgdehvmtnmvvmesvzifutzfjzoalwqztunwvdvmfhesvzifut zfjzoalwqztunpsnoyfleoxdetbwfsoyfjmfhjuxuagnarsfqydoyfjzsrzeujmfhjuubihrjdfinwsnepcawd nkbobvnmzucmghijjmbscjejnapddehlmqddmfxncqbfpxwfejifpqzhikiyaiozimubwuzufazsdjwdiu dzmztivcmgp’

明文内容：

’Itwasthebestoftimesitwastheworstoftimesitwastheageofwisdomitwastheageoffoolish nessitwastheepochofbeliefitwastheepochofincredulityitwastheseasonoflightitwastheseasonofdarkness itwasthespringofhopeitwasthewinterofdespairwehadeverythingbeforeuswehadnothingbeforeuswewe reallgoingdirecttoheavenwewereallgoingdirecttheotherwayinshorttheperiodwassofarlikethepresent periodthatsomeofitsnoisiestauthoritiesinsistedonitsbeingreceivedforgoodorforevilinthesuperlative degreeofcomparisononly

实验代码：

```

#2023.02.28  202100460116@mail.sdu.edu.cn

import math
cipherText='cbkznkiyjsrofgnqadnzuqigscvxizgsjwucusrdkxuahgzrhywtvdjeiuwsrrtnpszbvpzncngztbvsr
nzuqigscvfjwqgjwcytwdazuqigscvfjwqgjwjhkfdylmcbmhonbmbvdnvbmwbnacjaphhonbmbvdnvbmwbnaublsbdnj
jneoroyfmxfhixpzpcozzuqigscvxcvhdmfgxmgovzsqmvzyvwyzmsczoajsejifoakdcrehwhgdehvmtnmvvmesvzifu
tzfjzoalwqztunwvdvmfhesvzifutzfjzoalwqztunpsnoyfleoxdetbwfsoyfjmfhjuxuagnarsfqydoyfjzsrzeujmf
hjuubihrjdfinwsnepcawdnkbobvnmzucmghijjmbscjejnapddehlmqddmfxncqbfpxwfejifpqzhikiyaiozimubwuz
ufazsdjwdiudzmztivcmgp'
NaturalLanguageFrequencyList = ...
    [0.08167,0.01492,0.02782,0.04253,0.12702,0.02228,0.02015,0.06094,0.06966,
     0.00153,0.00772,0.04025,0.02406,0.06749,0.07507,0.01929,0.00095,0.05987,
     0.06327,0.09056,0.02758,0.00978,0.02360,0.00150,0.001974,0.00074]

#统计字符串中每个字母出现的频率
def countAlpha(str):
    return [str.count(chr(i)) for i in range(97,123)]

#找到序列中与某个值value(默认0.065)最接近的元素的序号
def ordinalOfClosest(indexList,value=0.065):
    minmum=min([abs(indexList[i]-value) for i in range(len(indexList))])
    for i in range(len(indexList)):
        if abs(indexList[i]-value)==minmum:
            ordinal=i
            break
```

```python
    return ordinal

def gcd_many(list_):
    result=list_[0]
    for i in list_[1:]:
        result=math.gcd(result,i)
        if result==1:
            return result
    return result

#计算固定长度下存在的重复子串 与 重复子串对应的位置
def findFixedLengthStr(cipherText,length):
    set_=set()
    dict_=dict()
    for i in range(len(cipherText)-1):
        item=cipherText[i:i+length]
        if item not in set_:
            set_.add(item)
            dict_[item]=[i]
        else:
            dict_[item].append(i)
    #输出与某个子字符串重复字符串的位置(无重复的去除)
    result={key:dict_[key] for key in dict_ if len(dict_[key])!=1}
    return result

#计算每种重复子串所代表的key长度
def analyse(dict_):
    keyLengthList=dict()
    for subStr in dict_:
        posList=dict_[subStr]
        disList=[posList[i+1]-posList[0] for i in range(len(posList)-1)]
        keyLength=gcd_many(disList)
        keyLengthList[subStr]=keyLength
    return keyLengthList


#1. 计算key长度
#1.1 kasiiki法计算密钥长度
def calKeyLengthByKasiiki(cipherText,length):
    dict_=findFixedLengthStr(cipherText,length)
    distance=analyse(dict_)
    for key in dict_:
        print(f'{key}: {distance[key]:<4} {dict_[key]}')

#计算某字符串重合指数
def calCoinIndex(str_):
    if(len(str_)==1):return 1
    n=len(str_)
    countList=countAlpha(str_)
    return sum([countList[i]*(countList[i]-1) for i in range(26)])/(n*(n-1))
#1.2 通过重合指数计算密钥长度
def calKeyLengthByCoinIndex(cipherText,keyLength):
    result=[calCoinIndex(cipherText[i::keyLength]) for i in range(len(cipherText)//keyLength)]
    return result
```

```


#2. 已知密钥长度，计算密钥组

#计算字符串与自然语言的重合互指数
def calCoinMutIndexListWithNatural(subStr):
    countList=countAlpha(subStr)
    resultList=[]
    for g in range(26):
        resultList.append(   sum([NaturalLanguageFrequencyList[i]*countList[(i+g)%26] for i ...
            in range(26)])/len(subStr)   )
    return resultList
#2.1 计算密钥相对于自然语言的长度
def calOffsetWithPlain(cipherText,keyLength):
    keyList=[]
    for groupOrdinal in range(keyLength):
        subStr=cipherText[groupOrdinal::keyLength]
        indexList=calCoinMutIndexListWithNatural(subStr)
        #找到该组26个重合互指数中最接近0.065的值g=k_i,则i即为该组的密钥
        ordinal=ordinalOfClosest(indexList)
        keyList.append(ordinal)
    return keyList

# 计算重合互指数
def calCoinMutIndexList(subStr1,subStr2):
    countList1=countAlpha(subStr1)
    countList2=countAlpha(subStr2)
    resultList=[]
    for Δ in range(26):
        resultList.append(  sum([countList1[i]*countList2[(i+Δ)%26] for i in ...
            range(26)])/(len(subStr1)*len(subStr2))   )
    return resultList
#2.2 计算密钥之间的相对距离
def calOffsetWithKeys(cipherText,keyLength):
    offsetList=[[ ] for i in range(keyLength)]
    #  keyLength*keyLength 的下对角矩阵
    for a in range(keyLength):
        for b in range(a+1):
            indexList=calCoinMutIndexList(cipherText[a::keyLength],cipherText[b::keyLength])
            #找到该组26个重合互指数中最接近0.065的值g=k_i,则i即为该组的密钥
            ordinal=ordinalOfClosest(indexList)
            offsetList[a].append(ordinal)
    return offsetList


#3. 使用给定密钥组解密
def decrypt(cipherText,key):
    keyLength=len(key)
    plainText=""
    for i in range(len(cipherText)):
        plainText+=chr(   ( ord(cipherText[i])-97-key[i%keyLength]  )%26 +97)
    return plainText

#4. 验证解密结果
```

```python
#4.1人工验证
#4.2 计算字符串与自然语言的重合互指数(字符串是否接近于自然语言)
def calCoinMutIndexWithNatural(subStr):
    countList=countAlpha(subStr)
    resultList=[]
    g=0
    result=sum([NaturalLanguageFrequencyList[i]*countList[(i+g)%26] for i in ...
        range(26)])/len(subStr)
    return result


def main():
    global cipherText
    global NaturalLanguageFrequencyList
    meanList=[]

    #1.1 重合指数法求密钥长度
    print(f'各猜测密钥长度下字符串的平均重合指数')
    maxKeyLength=20
    for i in range(1,maxKeyLength):
        coinList=calKeyLengthByCoinIndex(cipherText,i)
        meanList.append((sum(coinList)/len(coinList)))
        print(f'keyLength={i:3} {meanList[-1]}')
    print(f'keyLength应为{ordinalOfClosest(meanList)+1}')

    #1.2 kasiiki法求密钥长度
    calKeyLengthByKasiiki(cipherText,4)
    keyLength=7


    #2.1直接求密钥
    print(f'求字符串与自然语言重合互指数得到密钥组与密文如下')
    keyList=calOffsetWithPlain(cipherText,keyLength)
    print(keyList)
    plainText=decrypt(cipherText,keyList)
    print(plainText)


    #2.2先求密钥间相对距离,遍历一个密钥26次,对每种可能进行判断
    print(f'\n求密钥间距离矩阵为')
    offsetList=calOffsetWithKeys(cipherText,keyLength)
    for i in range(keyLength):
        print(offsetList[i])
    plainIndexList=[]
    print(f'遍历key_0 26种可能,分别对应的重合互指数')
    for i in range(26):
        keyList2=[(i-offsetList[k][0])%26 for k in range(1,keyLength)]
        keyList2.insert(0,i)
        plainText=decrypt(cipherText,keyList2)

        #3.1检验 人工识别
        #print(f'{plainText}\n')

        #3.2检验方式：计算重合指数最接近0.065的明文
        index=calCoinMutIndexWithNatural(plainText)
        print(index)
        plainIndexList.append(index)
    key_0=ordinalOfClosest(plainIndexList)
    print(f'\n当key_0为{key_0}时,key=')
    keyList3=[(key_0-offsetList[k][0])%26 for k in range(1,keyLength)]
    keyList3.insert(0,key_0)
    print(keyList3)
    print(f'对应明文为')
    print(decrypt(cipherText,keyList3))
main()
```

实验代码输出结果：

```
各猜测密钥长度下字符串的平均重合指数
keyLength= 1 0.044870830725922115
keyLength= 2 0.04340605818742676
keyLength= 3 0.04255186879826416
keyLength= 4 0.042699137540801575
keyLength= 5 0.04242518095769651
keyLength= 6 0.042226726851218144
keyLength= 7 0.07352056602641621
keyLength= 8 0.04093671729547781
keyLength= 9 0.039687319676896246
keyLength= 10 0.041885499168623955
keyLength= 11 0.0368481140935599
keyLength= 12 0.040375473573854126
keyLength= 13 0.038112832720675865
keyLength= 14 0.07563691076715055
keyLength= 15 0.03836266700156686
keyLength= 16 0.039139598235912135
keyLength= 17 0.0386876497987609
keyLength= 18 0.03905763751917598
keyLength= 19 0.03971014492753624
keyLength 应为 7
nzuq: 63 [18, 81]
zuqi: 7 [19, 82, 103, 194]
uqig: 7 [20, 83, 104, 195]
qigs: 7 [21, 84, 105, 196]
igsc: 7 [22, 85, 106, 197]
gscv: 7 [23, 86, 107, 198]
scvx: 175 [24, 199]
scvf: 21 [87, 108]
cvfj: 21 [88, 109]
vfjw: 21 [89, 110]
fjwq: 21 [90, 111]
jwqg: 21 [91, 112]
wqgj: 21 [92, 113]
qgjw: 21 [93, 114]
honb: 21 [129, 150]
onbm: 21 [130, 151]
nbmb: 21 [131, 152]
```

```
bmbv: 21 [132, 153]
mbvd: 21 [133, 154]
bvdn: 21 [134, 155]
vdnv: 21 [135, 156]
dnvb: 21 [136, 157]
nvbm: 21 [137, 158]
vbmw: 21 [138, 159]
bmwb: 21 [139, 160]
mwbn: 21 [140, 161]
wbna: 21 [141, 162]
ejif: 196 [234, 430]
esvz: 28 [260, 288]
svzi: 28 [261, 289]
vzif: 28 [262, 290]
zifu: 28 [263, 291]
ifut: 28 [264, 292]
futz: 28 [265, 293]
utzf: 28 [266, 294]
tzfj: 28 [267, 295]
zfjz: 28 [268, 296]
fjzo: 28 [269, 297]
jzoa: 28 [270, 298]
zoal: 28 [271, 299]
oalw: 28 [272, 300]
alwq: 28 [273, 301]
lwqz: 28 [274, 302]
wqzt: 28 [275, 303]
qztu: 28 [276, 304]
ztun: 28 [277, 305]
oyfj: 21 [326, 347]
jmfh: 28 [329, 357]
mfhj: 28 [330, 358]
fhju: 28 [331, 359]
求字符串与自然语言重合互指数得到密钥组与密文如下
[20, 8, 14, 25, 21, 17, 1]
itwasthebestoftimesitwastheworstoftimesitwastheageofwisdomitwastheageoffoolishnes
sitwastheepochofbeliefitwastheepochofincredulityitwastheseasonoflightitwastheseason
ofdarknessitwasthespringofhopeitwasthewinterofdespairwehadeverythingbeforeusweha
dnothingbeforeuswewereallgoingdirecttoheavenwewereallgoingdirecttheotherwayinsho
rttheperiodwassofarlikethepresentperiodthatsomeofitsnoisiestauthoritiesinsistedonitsbe
ingreceivedforgoodorforevilinthesuperlativedegreeofcomparisononly
求密钥间距离矩阵为
[0]
[12, 0]
[6, 20, 0]
[21, 5, 19, 11]
[25, 9, 23, 20, 1]
[3, 17, 23, 4, 4, 4]
[19, 7, 17, 9, 8, 20, 11]
遍历 key_0 26 种可能,分别对应的重合互指数
0.03160792842105263
0.033967107368421054
0.03253810947368422
0.027217414736842107
```

```
0.036804
0.04609724210526316
0.0383367957894737
0.040161301052631576
0.04398917894736842
0.04648454315789472
0.03598247578947369
0.03504650526315788
0.03571088421052632
0.035916901052631595
0.03016953263157894
0.03011535157894737
0.04288002105263158
0.035032968421052635
0.027161018947368422
0.03709063157894737
0.06811093894736844
0.043774400000000005
0.03040333052631579
0.03464564210526317
0.0493041852631579
0.03367559157894737
当 key_0 为 20 时,key=
[20, 8, 14, 25, 21, 17, 1]
对应明文为
itwasthebestoftimesitwastheworstoftimesitwastheageofwisdomitwastheageoffoolishnes
sitwastheepochofbeliefitwastheepochofincredulityitwastheseasonoflightitwastheseason
ofdarknessitwasthespringofhopeitwasthewinterofdespairwehadeverythingbeforeusweha
dnothingbeforeuswewereallgoingdirecttoheavenwewereallgoingdirecttheotherwayinsho
rttheperiodwassofarlikethepresentperiodthatsomeofitsnoisiestauthoritiesinsistedonitsbe
ingreceivedforgoodorforevilinthesuperlativedegreeofcomparisononly
```