Assignment 02: Markov Models

HE Yongyi from USTC

1. Temporal subsampling of a Discrete Time Markov Process

Suppose $X_1, X_2, \ldots$ forms a Markov process (that is **not** necessarily homogeneous for the purpose of this problem). Then recall that as per our definition,

$$p\left(x_n \mid x_{n-1}, x_{n-2}, \ldots x_1\right)=p\left(x_n \mid x_{n-1}\right) \tag{1}$$

for all $n \geq 1$. It can be seen that the definition in (1) is also equivalent to the condition that

$$p\left(x_n, x_{n-1}, x_{n-2}, \ldots x_1\right)=p\left(x_1\right) p\left(x_2 \mid x_1\right) p\left(x_3 \mid x_2\right) \cdots p\left(x_n \mid x_{n-1}\right) \tag{2}$$

for all $n \geq 1$.

(a) Show that for any positive integer n and any $k<n$

$$p\left(x_n, x_{n-1}, \ldots x_{n-k}\right)=p\left(x_{n-k}\right) p\left(x_{n-k+1} \mid x_{n-k}\right) p\left(x_{n-k+2} \mid x_{n-k+1}\right) \cdots p\left(x_n \mid x_{n-1}\right). \tag{3}$$

This result is equivalent to the condition that for any positive integer n and any $k<n$

$$p\left(x_n \mid x_{n-1}, x_{n-2}, \ldots x_{n-k}\right)=p\left(x_n \mid x_{n-1}\right). \tag{4}$$

It is also immediately obvious that (3) and (4) imply (1) and (2), respectively. Thus the conditions in (1), (2), (3), and (4) are all equivalent and any of these an be used as the defining condition for a Markov process.

Solution:

$$\begin{aligned} p\left(x_n, x_{n-1}, \ldots x_{n-k}\right) &= \sum_{x_{n-k-1} \in A_{n-k-1}} p\left(x_n, x_{n-1}, \ldots x_{n-k}, x_{n-k-1}\right) \\ &= \sum_{x_1 \in A_1} \cdots \sum_{x_{n-k-1} \in A_{n-k-1}} p\left(x_n, x_{n-1}, \ldots x_{n-k}, x_{n-k-1}, \ldots, x_1\right) \\ &= \sum_{x_1 \in A_1} \cdots \sum_{x_{n-k-1} \in A_{n-k-1}} p\left(x_1\right) p\left(x_2 \mid x_1\right) p\left(x_3 \mid x_2\right) \cdots p\left(x_n \mid x_{n-1}\right) \\ &= \sum_{x_1 \in A_1} \cdots \sum_{x_{n-k-1} \in A_{n-k-1}} p\left(x_2, x_1\right) p\left(x_3 \mid x_2\right) \cdots p\left(x_n \mid x_{n-1}\right) \\ &= \sum_{x_2 \in A_2} \cdots \sum_{x_{n-k-1} \in A_{n-k-1}} p\left(x_2\right) p\left(x_3 \mid x_2\right) \cdots p\left(x_n \mid x_{n-1}\right) \\ &= \sum_{x_{n-k-1} \in A_{n-k-1}} p\left(x_{n-k}, x_{n-k-1}\right) p\left(x_{n-k+1} \mid x_{n-k}\right) \cdots p\left(x_n \mid x_{n-1}\right) \\ &= p\left(x_{n-k}\right) p\left(x_{n-k+1} \mid x_{n-k}\right) \cdots p\left(x_n \mid x_{n-1}\right) \end{aligned}$$

(b) For $n=4$, (2) becomes

$$p\left(x_1, x_2, x_3, x_4\right)=p\left(x_1\right) p\left(x_2 \mid x_1\right) p\left(x_3 \mid x_2\right) p\left(x_4 \mid x_3\right) \tag{5}$$

Formally show that (1) also implies that

$$p\left(x_1, x_3, x_4\right)=p\left(x_1\right) p\left(x_3 \mid x_1\right) p\left(x_4 \mid x_3\right) \tag{6}$$

Solution:

$$\begin{aligned} p\left(x_1, x_3, x_4\right) &= \sum_{x_2 \in A_2} p\left(x_1, x_2, x_3, x_4\right) \\ &= \sum_{x_2 \in A_2} p(x_4 \mid x_1, x_2, x_3) p(x_1, x_2, x_3) \\ &= \sum_{x_2 \in A_2} p(x_4 \mid x_3) p(x_1, x_2, x_3) \\ &= p(x_4 \mid x_3) p(x_1, x_3) \\ &= p(x_4 \mid x_3) p(x_3 \mid x_1) p(x_1) \end{aligned}$$

(c) From the result of the preceding part, conclude that

$$p\left(x_4 \mid x_3, x_1\right)=p\left(x_4 \mid x_3\right) \tag{7}$$

Solution:

$$p\left(x_1, x_3, x_4\right)=p(x_4 \mid x_3) p(x_3 \mid x_1) p(x_1)$$

Also, we can get:

$$\begin{aligned} p\left(x_1, x_3, x_4\right) &= p\left(x_4 \mid x_3, x_1\right) p(x_3, x_1) \\ &= p\left(x_4 \mid x_3, x_1\right) p(x_3 \mid x_1) p(x_1) \end{aligned}$$

So, we can conclude the equation

$$
\begin{gathered}
p\left(x_4 \mid x_3, x_1\right) p(x_3 \mid x_1) p(x_1)=p(x_4 \mid x_3) p(x_3 \mid x_1) p(x_1) \\
p\left(x_4 \mid x_3, x_1\right)=p\left(x_4 \mid x_3\right)
\end{gathered}
$$

(d) Using the results from the preceding parts, formally show that

$$
p\left(x_1, x_2, x_4\right)=p\left(x_1\right) p\left(x_2 \mid x_1\right) p\left(x_4 \mid x_2\right) \tag{8}
$$

Solution:

From part (a), for any positive integer n and any $k<n$:

$$
p\left(x_n \mid x_{n-1}, x_{n-2}, \ldots x_1\right)=p\left(x_n \mid x_{n-1}\right)=p\left(x_n \mid x_{n-1}, x_{n-2}, \ldots x_{n-k}\right)
$$

For $n=4$ and $k=2$, the equation becomes

$$
p(x_4 \mid x_3, x_2, x_1)=p(x_4 \mid x_3, x_2)
$$

$$
\begin{aligned}
p\left(x_1, x_2, x_4\right) & =\sum_{x_3 \in A_3} p\left(x_1, x_2, x_3, x_4\right) \\
& =\sum_{x_3 \in A_3} p(x_4 \mid x_1, x_2, x_3) p(x_3 \mid x_1, x_2) p(x_1, x_2) \\
& =\sum_{x_3 \in A_3} p(x_4 \mid x_2, x_3) p(x_3 \mid x_2) p(x_1, x_2) \\
& =\sum_{x_3 \in A_3} p(x_4, x_3 \mid x_2) p(x_1, x_2) \\
& =p(x_4 \mid x_2) p(x_1, x_2) \\
& =p(x_4 \mid x_2) p(x_2 \mid x_1) p(x_1)
\end{aligned}
$$

(e) From the result of the preceding part, conclude that

$$
p\left(x_4 \mid x_2, x_1\right)=p\left(x_4 \mid x_2\right) . \tag{9}
$$

Solution:

$$
p\left(x_1, x_2, x_4\right)=p\left(x_1\right) p\left(x_2 \mid x_1\right) p\left(x_4 \mid x_2\right)
$$

Also, we can get:

$$
\begin{aligned}
p\left(x_1, x_2, x_4\right) & =p\left(x_4 \mid x_2, x_1\right) p(x_2, x_1) \\
& =p\left(x_4 \mid x_2, x_1\right) p(x_2 \mid x_1) p(x_1)
\end{aligned}
$$

So, we can conclude the equation

$$
\begin{gathered}
p\left(x_4 \mid x_2, x_1\right) p(x_2 \mid x_1) p(x_1)=p(x_4 \mid x_2) p(x_2 \mid x_1) p(x_1) \\
p\left(x_4 \mid x_2, x_1\right)=p\left(x_4 \mid x_2\right)
\end{gathered}
$$

(f) By continuing this line of reasoning, we can argue that if k is some positive integer and $n_1, n_2, \ldots n_k$ is any strictly increasing sequence of positive integers, then

$$p\left(x_{n_1}, x_{n_2}, \ldots x_{n_k}\right)=p\left(x_{n_1}\right) p\left(x_{n_2} \mid x_{n_1}\right) p\left(x_{n_3} \mid x_{n_2}\right) \cdots p\left(x_{n_k} \mid x_{n_k-1}\right) \quad (10)$$

State the above relation in words.

Solution:

$$\begin{aligned}
& p\left(x_{n_k}, x_{n_{k-1}}, \ldots, x_{n_1}\right) \\
& =p\left(x_{n_k} \mid x_{n_{k-1}}, \ldots, x_{n_1}\right) p\left(x_{n_{k-1}}, \ldots, x_{n_1}\right) \\
& =p\left(x_{n_k} \mid x_{n_{k-1}}, \ldots, x_{n_1}\right) p\left(x_{n_{k-1}} \mid x_{n_{k-2}} \ldots, x_{n_1}\right) p\left(x_{n_{k-2}} \ldots, x_{n_1}\right) \\
& =p\left(x_{n_k} \mid x_{n_{k-1}}\right) p\left(x_{n_{k-1}} \mid x_{n_{k-2}}\right) p\left(x_{n_{k-2}} \ldots, x_{n_1}\right) \\
& =\cdots \\
& =p\left(x_{n_k} \mid x_{n_{k-1}}\right) p\left(x_{n_{k-1}} \mid x_{n_{k-2}}\right) \cdots p\left(x_{n_2} \mid x_{n_1}\right) p\left(x_{n_1}\right)
\end{aligned}$$

We can state this relationship in such a way that for an ordered sequence of states, each state is only related to its nearest state.

---

2. Markov Models for Text: Seuss and Saki

The files "spamiam.txt" and "saki_story.txt" available on the website have poetry and prose of specific genres. For this problem, use the text in these files to empirically estimate probabilities and transition probabilities as indicated. Ignore any characters in these files other than the 26 alphabets 'a'-'z' (use white space and carriage returns as indicated in specific parts). Also ignore any case distinctions among alphabets (example 'C' and 'c' are equivalent).

For each of the sub-parts indicated below, print the 100 words that you generate in the form of a 10×10 array and circle any valid English words that you recognize.

(a) Assuming that the 26 letters of the alphabet are equiprobable. Generate one hundred random 4 letter words by selecting the 4 individual letters of each word independently.

Solution:

```python
import random

def word_print(str):
    for i in range(100):
        print(str[i],end=' ')
        if (i+1)%10 == 0:
            print('\n')

alphabet = "abcdefghijklmnopqrstuvwxyz"
for i in range(100):
    word = ''.join(random.choices(alphabet,k=4))
    solution_a[i] = word
word_print(solution_a)
```

```
draw deyh kozz knxe xzzl dogb jscg ufqc bvsg hbak

lqgl bcov ohww eyfs ebnp jugg ogzh gqck jxhx hche

xvsf gyul hpow xjqg mecl wpjp uyjt vxjl zqga vxpt

xagx dadm tlzd npet ahao nwfy ysae jpkg yrkf uxuv

bslu mizq ekmm kaqs kbud omlu tlnq kclw seti kupy

poyr thjn zgff kujv xcyk dika zfsm xayh eder qqbr

half hnkt tpvc wmok vipq yxso tskm zidx gugq ewvd

tsmg mudu inwe zcfj rnkq vzza gjjp wxra mgwm ptak

dksu mmjj lqvu ureq myqq nnms kguy zuqs lmil qrhd

aigz ntst srvj fulr enwg wsbj zbvy ivan sjna cxst
```

Score: 2/100

(b) Estimate the probabilities of individual letters using "spamiam.txt". Generate one hundred random 4 letter words by selecting the 4 individual letters of each word independently according to the estimated probability distribution.

Solution:

```python
file = open('spamiam.txt', 'r')
text = file.read().lower()
text = re.split(r'[\',-.?!;\n\t 1234567890]+',text)

temp = ''.join(text)

solution_b=[0]*100
for i in range(100):
    word = ''.join(random.choices(temp,k=4))
    solution_b[i] = word

word_print(solution_b)
```

```
dafk obnl itro etkt snpd eupn tket pvia audy atru

ohey kakn oeoe etil tneh piia timo oais iaay yeai

imyw lnat ilel alut paae kawn eyeo oltu ntln onoa

cdey tuel ldtp rtwu taul otid trod tlbu ztno olri

ikek tnur nttm ydao tmlw llsu dtam rkbe aiel tisr

eiyw oely eydy fait eeic miry ltya eroe otpe isem

cwey inid teth niva mfed oefm mowr eiut idro aldt

eimi yaei odhl ycia rdoi hsid etot etal keys aiel

ltdi taik itul neuo utro leod uiti peue ekee htnl

fvee wheo nous leoh amue urii dlkr utfs ahui onyf
```

Score: 4/100

(c) Again using the file "spamiam.txt", estimate the transition probabilities, $P(x_{n+1}|x_n)$, for all 26 possible values of x_n - the n^{th} letter in a word and x_{n+1} - the $(n+1)^{th}$ letter in a word (assume that these probabilities are independent of n). Also for this part and the next, for your estimation of transition probabilities, use only the letters inside a word for the computation and do not incorporate letters from adjacent words (with a blank in between). Generate one hundred random 4 letter words by first generating a letter at random according to the probability mass function (pmf) in 2b and then generating remaining letters according to appropriate transition probabilities. *Note: You may default to the model of 2b if you end up with a situation where your estimate of* $P(x_{n+1}|x_n)$ *is zero for all values of* x_{n+1}.

Solution:

```python
file = open('spamiam.txt', 'r')
text = file.read().lower()
text = re.split(r'[\',-.?!;\n\t 1234567890]+',text)

```

```python
# train
cfd=nltk.ConditionalFreqDist()
for k in range(len(walden)):
    if len(walden[k])-1 < 0:
        continue
    for i in range(len(walden[k])-1):
        cfd[walden[k][i]][walden[k][i+1]] += 1

# test
temp = ''.join(text)
solution_c=[0]*100
for n in range(100):
    letter=random.choice(temp)
    word=letter
    for i in range(3):
        arr = []
        if len(cfd[letter]) == 0:
            letter = random.choice(temp)
        else:
            for j in cfd[letter]:
                for k in range(cfd[letter][j]):
                    arr.append(j)
            letter = random.choice(arr)
        word += letter
    solution_c[n]=word

word_print(solution_c)
```

```
tspy eewi heee thee kera nyou oudo scke meno ikea
oule heer arer pamy ldou dldo itou hike youl illl
ithe deer idou spat spit thet reem them tere thew
nony here lldo nome chen ulde ikev tspa mist urot
mamy ther rere noth noul illi noth lild nouy dono
tama dour hete ulik dere oure othe ldou kero newo
ldou noun fldo visp ikee omet math hero oudo whee
youl mean eano ulde aill ouro mere toth youl hath
noth ould omam ewie trea ther onke like myor noth
unce ldou ithe mema thar eare othe emik ithe adou
```

Score: 15/100

(d) Once again use the file "spamiam.txt", to estimate the transition probabilities $P(x_{n+1}|x_n, x_{n-1})$, for all possible values of the successive letters. Generate one hundred random four letter words using these estimated probabilities. Make reasonable assumptions that generalize what was indicated in 2c.

Solution:

```
file = open('spamiam.txt', 'r')
text = file.read().lower()
text = re.split(r'["\',-.?!;\n\t 1234567890]+',text)

# train
cfd2=nltk.ConditionalFreqDist()
for k in range(len(text)):
    if len(text[k])-1 < 1:
        continue
    for i in range(len(text[k])-2):
        cfd2[text[k][i]+text[k][i+1]][text[k][i+2]] += 1

# test
temp = ''.join(text)

solution_d=[0]*100

for n in range(100):
    letter=random.choice(temp)
#     letter=''.join(random.choices(temp,k=2))
    for j in cfd[letter]:
        for k in range(cfd[letter][j]):
            arr.append(j)
    letter += random.choice(arr)
    word=letter
    for i in range(2):
        arr = []
        if len(cfd2[letter]) == 0:
            letter = letter[-1] + random.choice(temp)
        else:
            for j in cfd2[letter]:
                for k in range(cfd2[letter][j]):
                    arr.append(j)
            letter = letter[-1] + random.choice(arr)
        word += letter[-1]
    solution_d[n]=word

```

```python
word_print(solution_d)
```

```
your ithe iiio that otnl deck kedd your enec llyi
ldet vcau ldai iken zieo eeet some spam ywhe dead
like urth ould myeu like getc spam amid maym rees
meth ther itha woul mlrn ldec coul eree then otly
nche otit amid tant tdsm amid tsun yram athi otdl
heel noto eree inkv ithe eytl like ueme rome gail
itst notr ikeo seel heyi enee eekr real youl ethe
uetc houl tldo ldni eead keud uree amid heme orom
pami trea reet from fink like emet like tche notl
ldms here neck noti lewd trea youl elet ldle test
```

Score: 28/100

(e) Repeat parts 2b-2d using the file "saki_story.txt".

Solution:

Repeat part 2b:

```
isas uoeo hhmu olwo rini hose vaot rdar witm iiip
bted gead ooyt ithe idoa iesn etse idto baav uoai
tbee eoem mntt desw dsee iulu erya hlaf amaa heti
nooh fsmd sipe ihne obyi nieh wenm sdtp oiap haae
olbs snht zdgh oitt isot drga hihe ftec siup eene
ihtu acwt enuc rnpe rhad sewi oian sdhe arvo lsoh
hers tsha gbor hifn nhhe tteh aora gteo aeno sini
sdoe halr ecfa feuh otea hohd eeci pbem meir omtb
ietk fati nhah eoin eere tens odhm rshe anhw floe
hsea heeb oftn ogne nbnw rept tfrf teen ocxt guea
```

Score: 6/100

Repeat part 2c:

```
eyen kelo trem mast dfou faly hads enge oule erea
erev lost snye erys nlyo tolu aver ngro kefr idft
rorr nonc expl veld deas dyse frst icti enki llon
yesi rpth east toss thes tory ldfu eerr lida stba
ince olol tinv megs cyet arer yith sale emon eare
esin aing amem athe sthe wist wild both sase obem
test tier asth nsak ilde esim geno rogr mlde isth
asst nden dofu hiou thed dien rofr fofo tran ingr
celo nger avie rent spor neme amer yard teen erst
thel ratf enge kest hent hase hero ldyo oren voke
```

Score: 20/100

Repeat part 2d:

```
stly edus fami aste itti mend hatz ntiv ther orti
dsat howl ands rnot ones ooki heye ithe self agiv
nest heen hour ndsi arpe ndfa acto ofic tory tice
thad rstl omem drat egen siti expe inge ityh past
esse rast bled chat ther ntle ndow isni bern esse
heir orib anyw ging llyk eyes oman lden omen clag
edge hade ncer hunt rand kger denc heat scal king
gger octf ulde anin rone ally ouch arou ince oman
ines ewsp cemp tong cone essa sori mere utbu orie
ardi thei ayed resp iene anyw arou gove said erne
```

Score: 30/100

(f) Comment on your results.

In 2c we first use the probability mass function in 2b to generate the first letter, and then use the first-order Markov model. We can see, the results of training with the first-order Markov model in 2c are significantly better than the results of training with just the probability mass function in 2b.

In 2d, we first use the probability mass function in 2b to generate the first letter, then use the first-order Markov model in 2c to generate the second letter, and finally use the second-order Markov model. We can see, the results of training with the second-order Markov model in 2d are significantly better than the results of training with first-order Markov model in 2c. And the generated words are highly correlated with the training text, in other words, the generated words have the style of the training text.

Next, we do a side-by-side comparison of the two texts. For the same model, the number of valid words generated by training "saki_story.txt" is always more than the number of valid words generated by training "spamiam.txt". "spamiam.txt" contains 771 words (2434 letters), and "saki_story.txt" contains 1727 words (7292 letters). So we can infer that, within a certain range, the larger the training set is, the better the results obtained.

(g) **Extra Credit**: Estimate the entropy rate for each of the Markov models you developed and compare these both across models for a single data file and across the two data files. You may need to make suitable assumptions in order to determine your answers (which may be hard/impossible to validate)

Solution:

The entropy rate of the stochastic process X_i is defined when the following limit exists:

$$H(\mathcal{X})=\lim_{n \rightarrow \infty} \frac{1}{n} H\left(X_1, X_2, \cdots, X_n\right) \quad \text{or} \quad H^{\prime}(\mathcal{X})=\lim_{n \rightarrow \infty} H\left(X_n \mid X_{n-1}, X_{n-2}, \cdots, X_1\right).$$

The above two equations reflect two different aspects of the concept of entropy rate. The first refers to the entropy of each character of the n random variables. The second refers to the conditional entropy of the last random variable in the case where the previous $n-1$ random variables are known. For a stationary process, both of these limits exist and are equal, i.e, $H(\mathcal{X})=H^{\prime}(\mathcal{X})$.

For a stationary Markov chain, the entropy rate is

$$\begin{aligned} H(\mathcal{X})=H^{\prime}(\mathcal{X}) & =\lim H\left(X_n \mid X_{n-1}, \cdots, X_1\right)=\lim H\left(X_n \mid X_{n-1}\right) \\ & =H\left(X_2 \mid X_1\right) \end{aligned}$$

where the conditional entropy can be calculated from the stationary distribution.

The entropy rate convergence theorem for Markov chains is described formally below. Let X_i be a stationary Markov chain with a stationary distribution μ and a transition matrix P. Then the entropy rate is

$$H(\mathcal{X})=\sum_i \mu_i\left(\sum_j-P_{i j} \log P_{i j}\right)=-\sum_{i j} \mu_i P_{i j} \log P_{i j}.$$

So for this part, we assume that this Markov process is a stationary process and the distribution of each letter of the training text (data file) is stationary distribution.

Model 2c (Model I):

- The file "spamiam.txt"(File I):

  Count and calculate the distribution of each letter through this file, and generate the transition matrix by training. Then estimate the entropy rate $H(\mathcal{X}) \approx 1.9503$.

- The file "saki_story.txt"(File II):

The method and process are the same as above. Then estimate the entropy rate $H(\mathcal{X}) \approx 1.0314$.

For Model I, the entropy rate in File II is lower than in File I, may because File II's training set is larger. By training in File II, uncertainty may be effectively reduced.

Model 2d (Model II):

- The file "spamiam.txt"(File I):

  Count and calculate the distribution of each two letters through this file, and use only the letters inside a word for the computation and do not incorporate letters from adjacent words. For example, in the string 'want to', we only count 'wa', 'an', 'nt' and 'to'. Generate the transition matrix, and we mark the transition step for $P(x_{n+1}|x_n, x_{n-1})$ as $(x_{n-1}, x_n) \rightarrow (x_n, x_{n+1})$. Then estimate the entropy rate $H(\mathcal{X}) \approx 0.8900$.

- The file "saki_story.txt"(File II):

  The method and process are the same as above. Then estimate the entropy rate $H(\mathcal{X}) \approx 6.1801$.

For Model II, the entropy rate in File I is lower than in File II. In this model, we use two states to predict one state. So the larger training set is, the more kinds of state model may have. For this model, large training set may lead to overfitting problem.

For File I, the entropy rate in Model II is lower than in Model I. File I's training set is not large, so not many states may be generated with Model II and the uncertainty may be effectively reduced.

For File II, the entropy rate in Model I is lower than in Model II. By training the Model II, too many states may be generated, uncertainty may be significantly increased.