# DEPARTMENTOFINFORMATIONTECHNOLOGY

**COURSECODE: DJ19ITL504 DATE:6/12/22**

# COURSENAME: Artificial Intelligence Laboratory CLASS:TY-IT

**EXPERIMENTNO.10**

**CO/LO:** Apply NLP techniques on domain specific problems.

**AIM / OBJECTIVE:** To implement Hidden Markov Model for tagging Parts of Speech.

# DESCRIPTIONOFEXPERIMENT:

# A Hidden Markov Model (HMM) is a statistical Markov model in which the system being modeled is assumed to be a Markov process with unobserved (hidden) states. In a regular Markov model (Markov Model (Ref: http://en.wikipedia.org/wiki/Markov\_model)), the state is directly visible to the observer, and therefore the state transition probabilities are the only parameters. In a hidden Markov model, the state is not directly visible, but output, dependent on the state, is visible.

# Hidden Markov Model has two important components-

# 1)Transition Probabilities: The one-step transition probability is the probability of transitioning from one state to another in a single step.

# 2)Emission Probabilities: The output probabilities for an observation from state. Emission probabilities B = { bi,k = bi(ok) = P(ok | qi) }, where ok is an Observation. Informally, B is the probability that the output is ok given that the current state is qi

# For POS tagging, it is assumed that POS are generated as random process, and each process randomly generates a word. Hence, transition matrix denotes the transition probability from one POS to another and emission matrix denotes the probability that a given word can have a particular POS.

**Objective:**

1. Take a corpus

2. Calculate the emission and transmission probabilities.

3. Analyze the results for a valid and invalid sentence structure.

4. Analyze the results for a valid and invalid POS tagging.

# Explanation/Solutions (Design):

import itertools as itr

s1="Mary Jane can see Will"   #dataset

s2="Spot will see Mary"

s3="Will Jane spot Mary"

s4="Mary will pat Spot"

s1=s1.lower()      #preprocessing

s2=s2.lower()

s3=s3.lower()

s4=s4.lower()

corpus=[s1,s2,s3,s4]      #corpus

s=set()

for i in s1.split():

    s.add(i)

for i in s2.split():

    s.add(i)

for i in s3.split():

    s.add(i)

for i in s4.split():

    s.add(i)

print("corpus : \n",corpus)

print("\n")

s1tags={"mary":"N","jane":"N","can":"M","see":"V","will":"N"}

s2tags={"spot":"N","will":"M","see":"V","mary":"N"}

s3tags={"will":"M","jane":"N","spot":"V","mary":"N"}

s4tags={"mary":"N","will":"M","pat":"V","spot":"N"}       #manual tagging

tp={}                                     #transmission probablity

for i in s:

    tp[i]=[0,0,0]

for i in s1tags.keys():

    if s1tags[i]=="N":

        tp[i][0]=tp[i][0]+1

    elif s1tags[i]=="M":

        tp[i][1]=tp[i][1]+1

    elif s1tags[i]=="V":

        tp[i][2]=tp[i][2]+1

for i in s2tags.keys():

    if s2tags[i]=="N":

        tp[i][0]=tp[i][0]+1

    elif s2tags[i]=="M":

        tp[i][1]=tp[i][1]+1

    elif s2tags[i]=="V":

        tp[i][2]=tp[i][2]+1

for i in s3tags.keys():

    if s3tags[i]=="N":

        tp[i][0]=tp[i][0]+1

    elif s3tags[i]=="M":

        tp[i][1]=tp[i][1]+1

    elif s3tags[i]=="V":

        tp[i][2]=tp[i][2]+1

for i in s4tags.keys():

    if s4tags[i]=="N":

        tp[i][0]=tp[i][0]+1

    elif s4tags[i]=="M":

        tp[i][1]=tp[i][1]+1

    elif s4tags[i]=="V":

        tp[i][2]=tp[i][2]+1

a=0

b=0

c=0

for i in tp.keys():

    a=tp[i][0]+a

    b=tp[i][1]+b

    c=tp[i][2]+c

#print(a,b,c)

for i in tp.keys():

    tp[i][0]=tp[i][0]/a

    tp[i][1]=tp[i][1]/b

    tp[i][2]=tp[i][2]/c

print("transmission probabilties : \n",tp)

print("\n")

s1="<s> " +s1 +" <e>"  #preprocessing

s2="<s> " +s2 +" <e>"

s3="<s> " +s3 +" <e>"

s4="<s> " +s4 +" <e>"

s1indxed=[]

s2indxed=[]

s3indxed=[]

s4indxed=[]

for i in s1.split():

    if i=="<s>" or i=="<e>":

        s1indxed.append(i)

    else:

        s1indxed.append(s1tags[i])

for i in s2.split():

    if i=="<s>" or i=="<e>":

        s2indxed.append(i)

    else:

        s2indxed.append(s2tags[i])

for i in s3.split():

    if i=="<s>" or i=="<e>":

        s3indxed.append(i)

    else:

        s3indxed.append(s3tags[i])

for i in s4.split():

    if i=="<s>" or i=="<e>":

        s4indxed.append(i)

    else:

        s4indxed.append(s4tags[i])

ep={}                                  #emission probablities

for i in range(1,len(s1indxed)):

    try:

        ep[s1indxed[i-1]+s1indxed[i]]=ep[s1indxed[i-1]+s1indxed[i]]+1

    except:

        ep[s1indxed[i-1]+s1indxed[i]]=1

for i in range(1,len(s2indxed)):

    try:

        ep[s2indxed[i-1]+s2indxed[i]]=ep[s2indxed[i-1]+s2indxed[i]]+1

    except:

        ep[s2indxed[i-1]+s2indxed[i]]=1

for i in range(1,len(s3indxed)):

    try:

        ep[s3indxed[i-1]+s3indxed[i]]=ep[s3indxed[i-1]+s3indxed[i]]+1

    except:

        ep[s3indxed[i-1]+s3indxed[i]]=1

for i in range(1,len(s4indxed)):

    try:

        ep[s4indxed[i-1]+s4indxed[i]]=ep[s4indxed[i-1]+s4indxed[i]]+1

    except:

        ep[s4indxed[i-1]+s4indxed[i]]=1

#print(ep)

s=0

n=0

m=0

v=0

for i in ep.keys():

    if i[0]=="N":

        n=n+ep[i]

    elif i[0]=="M":

        m=m+ep[i]

    elif i[0]=="V":

        v=v+ep[i]

    elif i[0]=="<":

        s=s+ep[i]

#print(n,m,v,s)

for i in ep.keys():

    if i[0]=="N":

        ep[i]=ep[i]/n

    elif i[0]=="M":

        ep[i]=ep[i]/m

    elif i[0]=="V":

        ep[i]=ep[i]/v

    elif i[0]=="<":

        ep[i]=ep[i]/s

print("emission probabilities : \n",ep)

print("\n")

sentence="Will can spot Mary"                     #testing

sentence=sentence.lower()                         #preprocessing

words=sentence.split()

possible=["N","M","V"]

ap=list(itr.product(possible,repeat=len(words)))  #all possible ways to tag the sentence

#print(ap)

sol=[]

for i in ap:

    wtags=dict(zip(words,i))

    #w="<s> "+sentence +" <e>"

    blocks=list(wtags.values())

    blocks.append("<e>")

    blocks.insert(0,"<s>")

    #print(wtags)

    #calculate for assumed sequence :

    prob=1

    for j in words:

        if wtags[j]=="N":

            col=0

        elif wtags[j]=="M":

            col=1

        elif wtags[j]=="V":

            col=2

        prob=prob\*tp[j][col]     #multiply transmission probablities

    for z in range(1,len(blocks)):

        curr=blocks[z-1]+blocks[z]

        try:

            prob=prob\*ep[curr]      #multiply emission probablities

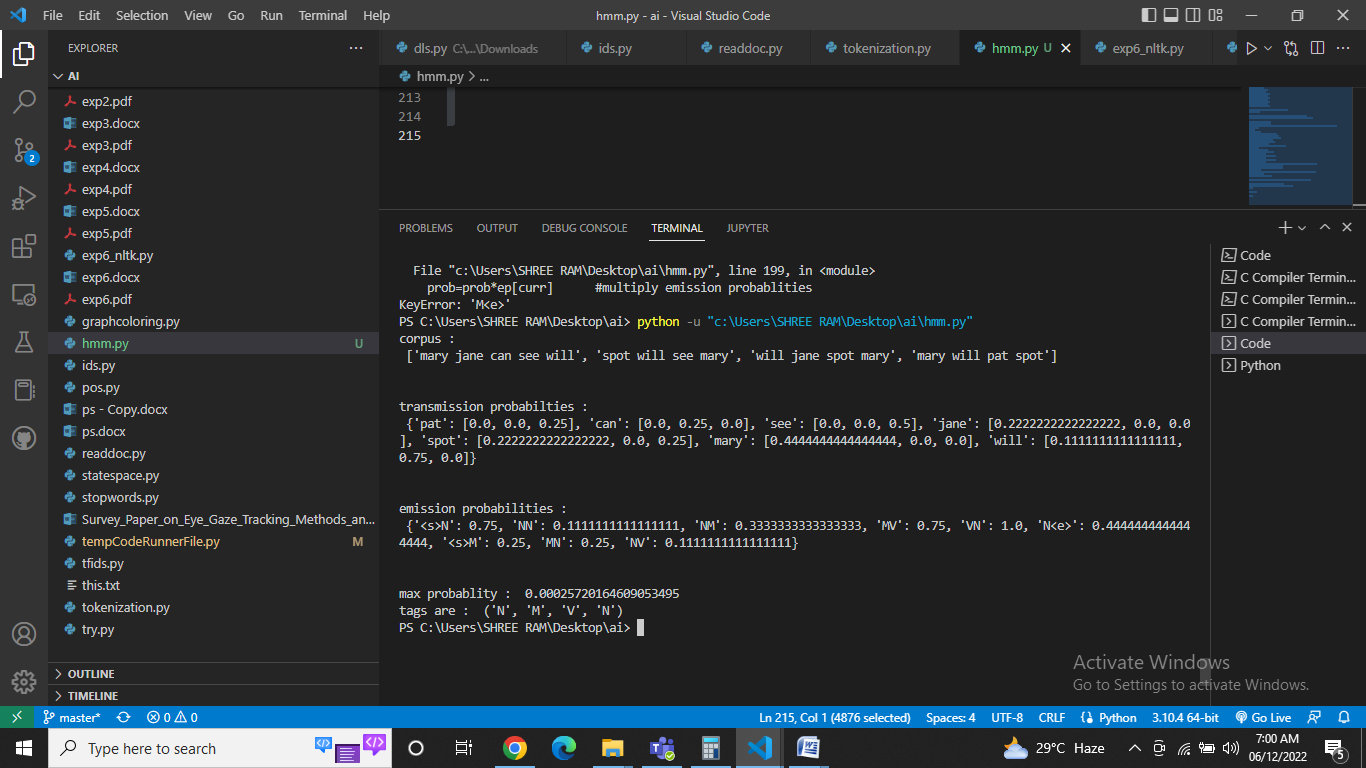
        except:

            prob=prob\*0

    sol.append(prob)           #keep track of all probablities

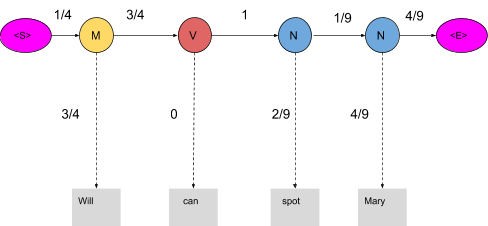
print("max probablity : ",max(sol))

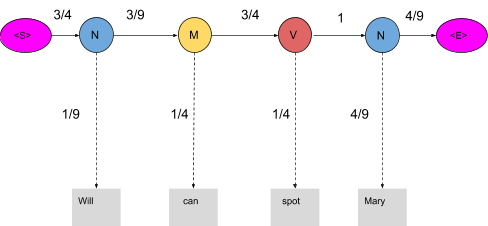
print("tags are : ",ap[sol.index(max(sol))])



Output:

any arbitrary tags probability: **1/4\*3/4\*3/4\*0\*1\*2/9\*1/9\*4/9\*4/9=0**





max tags probability: **3/4\*1/9\*3/9\*1/4\*3/4\*1/4\*1\*4/9\*4/9=0.00025720164**

# POS taggingHMM model:

# POS taggingFinal Tags:

# CONCLUSION:

# Thus we have implemented parts of speech tagging on given corpus using hidden markov model

# REFERENCES:

# [[1]: https://www.mygreatlearning.com/blog/pos-tagging/#:~:text=HMM%20(Hidden%20Markov%20Model)%20is,%2C%20partial%20discharges%2C%20and%20bioinformatics.](%5b1%5d:%20https:/www.mygreatlearning.com/blog/pos-tagging/%23:~:text=HMM%20(Hidden%20Markov%20Model)%20is,%2C%20partial%20discharges%2C%20and%20bioinformatics.)