CS 533: Natural Language Processing **(Due: 03/31/20)**

Assignment 4

Name: Madhu Sivaraj, *NetID:* ms2407

Problem 1: HMM

(1 + 6 + 6 = 13 points)

1. Emission probabilities:
   $o(the \mid D) = 1$
   $o(the \mid N) = \frac{1}{6}$
   $o(man \mid N) = \frac{1}{3}$
   $o(saw \mid V) = \frac{1}{2}$
   $o(saw \mid N) = \frac{1}{3}$

   Transitional probabilities:
   $t(D \mid *) = \frac{2}{3}$
   $t(D \mid V) = 1$
   $t(N \mid D) = 1$
   $t(N \mid N) = \frac{1}{6}$
   $t(N \mid *) = \frac{1}{3}$
   $t(V \mid N) = \frac{1}{3}$
   $t(STOP \mid N) = \frac{1}{2}$

   The nonzero emission probabilities for the word 'cut' are:
   $o(cut \mid N) = \frac{1}{6} = 0.166667$
   $o(cut \mid V) = \frac{1}{2} = 0.5$

2. The probability under the HMM that the third word is tagged with V conditioning on $x^{(2)}$ is 0.00626, rounded to 5 decimal places. This probability was calculated by running the forward and backward algorithms as a function of forward and backward probabilities.

```
def forward(words):
    forward_prob=[{}]
    for tag in tags:
        forward_prob[0][tag]=transition_prob['*'][tag]*emission_prob[tag][words[0]]
    for i, obs in enumerate(words[1:],1):
        forward_prob.append({})
        for tag in tags:
            prob=0
            for prev in tags:
                prob+=forward_prob[i-1][prev]*transition_prob[prev][tag]*emission_prob[
                    ↪ tag][obs]
            forward_prob[i][tag]=prob
    return forward_prob

def backward(words):
    backward_prob=[{} for i in range(len(words))]
    for tag in tags:
        backward_prob[-1][tag]=transition_prob[tag]['STOP']
    for i, obs in reversed(list(enumerate(words[:-1]))):
        for tag in tags:
            prob=0
```

```
            for prev in tags:
                prob+=forward_prob[i+1][prev]*transition_prob[tag][prev]*emission_prob[
                    ↪ prev][words[i + 1]]
            backward_prob[i][tag]=prob
    return backward_prob
```

3. The probability that the fifth word is tagged with N conditioning on $x^{(1)}$ is 0.00313, rounded to 5 decimal places. This probability was calculated by running the forward and backward algorithms as a function of forward and backward probabilities.

Problem 2: PCFG (1 + 6 + 6 = 13 points)

1. The MLE parameter values of (u, b) estimated from this corpus.

   Unary Productions:
   $u(D \rightarrow the \mid D) = \frac{4}{6}$
   $u(D \rightarrow a \mid D) = \frac{2}{6}$
   $u(N \rightarrow boy \mid N) = \frac{2}{6}$
   $u(N \rightarrow man \mid N) = \frac{2}{6}$
   $u(N \rightarrow telescope \mid N) = \frac{1}{3}$
   $u(P \rightarrow with \mid P) = 1$
   $u(V \rightarrow saw \mid V) = 1$

   Binary Productions:
   $b(S \rightarrow NPVP \mid S) = 1$
   $b(NP \rightarrow DN \mid NP) = \frac{6}{7}$
   $b(NP \rightarrow NPPP \mid NP) = \frac{1}{7}$
   $b(VP \rightarrow VNP \mid VP) = \frac{2}{3}$
   $b(VP \rightarrow VPPP \mid VP) = \frac{1}{3}$
   $b(PP \rightarrow PNP \mid PP) = 1$

   I calculated this by counting up all the times a production occurs across both trees, and then divided it by the number of times it occurred at all.

2. The probability under the PCFG that NP spans (4, 8) (i.e., "the man with a telescope") conditioning on x is 0.000329, rounded to 6 decimal places. I calculated this probability by running the inside and outside algorithms.

```
def inside(words,unary_rules,binary_rules):
    inside_prob=defaultdict(float)
    for i,word in enumerate(words,start=1):
        for urule in unary_rules.keys():
            if word==urule[1]:
                inside_prob[urule[0],i,i]=unary_rules[urule]
    for i in range(len(words)):
        for j in range(len(words)):
            for brule in binary_rules:
                rule0,rule1=brule[1].split()
                for k in range(j,i+j):
                    if inside_prob[rule0,j,k] and inside_prob[rule1,k+1,i+j]:
                        inside_prob[brule[0],j,i+j] += binary_rules[brule]*inside_prob[
                            ↪ rule0,j,k]*inside_prob[rule1,k+1,i+j]
    return inside_prob
```

```
def outside(words,binary_rules,inside_prob):
    outside_prob=defaultdict(float)
    outside_prob['S',1,len(words)]=1
    for i in reversed(range(len(words))):
        for j in range(len(words) - i+1):
            for brule in binary_rules:
                rule0,rule1=brule[1].split()
                for k in range(i+j,len(words)+1):
                    outside_prob[rule0,j,i+j] += binary_rules[brule]*outside_prob[brule
                        ↪ [0],j,k]*inside_prob[rule1,i+j+1,k]
            for rule in binary_rules:
                rule0,rule1=rule[1].split()
                for k in range(1,i+j):
                    outside_prob[rule1,j,i+j] += binary_rules[rule]*outside_prob[rule[0],
                        ↪ k,i+j]*inside_prob[rule0,k,j-1]
    return outside_prob
```

3. The probability that VP spans (3, 5) (i.e., "saw the man") conditioning on x is 0.000768 (rounded to 6 decimal places). I calculated this probability by running the inside and outside algorithms.

Problem 3: Programming (CRF) (22 points)

See crf.py for my implementation of the compute_normalizers and decode functions.