CSE 250A Hwk 1 —

1.1) a) $P(X|Y,E)$ x

$P(X,Y|E) =$

b) $P(X, Y | E)$

2-3:

$$P(x, y \mid E)$$

$$P(y \mid x, E)$$

Problem 1.3

a) X: Person has C

c) X: John

Y: Nan

Z: A

X, Y are not c

John & Nan

1.4) (B

P(D = 1)

P(D = 0)

1.5) $H(X) =$

$D_H = \frac{d(}{\delta}$

$$= - \sum_{x_1} \sum_{x_2} \sum_{x_3} \sum_{x_4}$$

$$= - \sum_{(x)} p(x_i)$$

Intuition

Problem 1.6

$$K_L = \sum_i p_i \log($$

a) let's first

b) KL(p,q

given the

$\sum p_i \log$

given $\sum p_i = \sum q_i =$

we can include $\sum$

$KL(p,q) \geq \sum p$

1.7)a - Using resu

$$I(X;Y) = \sum \sum$$

$$\times$$

$$= D_{KL}(p, q)$$

b) Indep. = $P(x,$

if $I(x,y) = 0$.

while $P(x,y) = 0$.

1.8

a) Yes, BN, implies

Problem 1.9 - Part a

```
import operator
import numpy as np
with open("hw1_word_counts_05.txt", 'r') as f:
  contents = f.readlines()

dicti = {}
for item in contents:
    word, num = item.split()
    dicti[word] = int(num)

sorted_dicti = dict(sorted(dicti.items(), key=operator.itemgetter(1),reverse=True))
# print(dicti)
keysList = list(sorted_dicti.keys())
valsList = list(sorted_dicti.values())
#dictionary holding prob of each word
count = len(keysList)
total = sum(valsList)
prob = {}
for i in range(count):
  prob[keysList[i]] = ( valsList[i]/total)
print(prob)
print("15 Most common words")
print(keysList[:15])
print("14 Least common words")
print(keysList[-14:])
```

```
{'THREE': 0.03562714868653127, 'SEVEN': 0.023332724928853858, 'EIGHT': 0.021626496097709
15 Most common words
['THREE', 'SEVEN', 'EIGHT', 'WOULD', 'ABOUT', 'THEIR', 'WHICH', 'AFTER', 'FIRST', 'FIFT
14 Least common words
['CCAIR', 'CLEFT', 'FABRI', 'FOAMY', 'NIAID', 'PAXON', 'SERNA', 'TOCOR', 'YALOM', 'BOSA
```

Yes, results do make sense.

Part b

Removing words that are not compatible with our current setting

```
import string
```

```
def remove_corr(corr_guess):
  temp = {}
  output = []
  alphabet = list(string.ascii_uppercase)
  for elemi in corr_guess:
      output.append(elemi[1])
      if elemi[0] in alphabet:
        alphabet.remove(elemi[0])
  # print(output)
  # print(alphabet)
  for elem in dicti:
    f = 0
    for i in range(5):
      if(i in output):
        for element in corr_guess:
          if element[1] == i:
            x = element[0]
        if(elem[i] != x):
          f = 1
          break
      else:
        if(elem[i] not in alphabet):
          f = 1
          break
    if(f == 0):
      temp[elem] = dicti[elem]
  return temp
```

Removing words with incorrect letters in them

```
  def removeincorr (incorr_guess, temp):
    final = {}
    for elem in temp:
      f = 0
      for i in range(len(incorr_guess)):
        if(incorr_guess[i] in elem):
          f = 1
          break
      if(f == 0):
        final[elem] = temp[elem]
    return final


# temp = remove_corr([["D",0],["I",3]])
# removeincorr(["A"] , temp)
```

Combining these two sub functions into one:

```
def validwords(dicti , corr_guess, incorr_guess):
  if corr_guess == [[]]:
    temp = dicti
  else:
    temp = remove_corr(corr_guess)
  if(incorr_guess == []):
    return temp
  else:
    final = removeincorr( incorr_guess , temp)
  return final


# validwords(dicti,[["U",1]],["A","E","I","O","S"])


import string
def rec_lett(dicti , corr_guess, incorr_guess):
  space = validwords(dicti , corr_guess, incorr_guess)
  rec_dict = {}
  total_count = sum(space.values())
  for elem in space:
    seti = set()
    for i in range(5):
      if(elem[i] not in rec_dict and elem[i] not in seti):
        rec_dict[elem[i]] = space[elem]
        f = 1
      elif(elem[i] in rec_dict and elem[i] not in seti):
        rec_dict[elem[i]] += space[elem]
        f = 1
      seti.add(elem[i])
  if(corr_guess == [[]]):
    answer = dict(sorted(rec_dict.items(), key=operator.itemgetter(1),reverse=True))
    # print("Answer dictionary:", answer)
    # print("total count:", total_count)
    res = list(answer.keys())[0]
    value = answer[str(res)]
    probability = value/total_count
    return (res,probability)
  else:
    final_output = {}
    for element in rec_dict:
      flag = 0
      for i in range(len(corr_guess)):
        if corr_guess[i][0] == element:
          flag = 1
          break
      if (flag == 0):
```

```
            final_output[element] = rec_dict[element]
        answer = dict(sorted(final_output.items(), key=operator.itemgetter(1),reverse=True))
        # print("total count:", total_count)
        res = list(answer.keys())[0]
        value = answer[str(res)]
        probability = value/total_count
        return(res,probability)
```

```
print(rec_lett(dicti, [[]] ,[]))
print(rec_lett(dicti, [[]] ,["E", "A"]))
print(rec_lett(dicti, [["A",0] , ["S" , 4]] ,[]))
print(rec_lett(dicti, [["A",0] , ["S" , 4]] ,["I"]))
print(rec_lett(dicti, [["O",2]] ,["A","E","M","N","T"]))
print(rec_lett(dicti, [[]] ,["E","O"]))
print(rec_lett(dicti, [["D",0] , ["I" , 3]] ,[]))
print(rec_lett(dicti, [["D",0] , ["I" , 3]]  ,["A"]))
print(rec_lett(dicti, [["U",1]] ,["A","E","I","O","S"]))
```

```
     ('E', 0.5394172389647974)
     ('O', 0.5340315651557658)
     ('E', 0.7715371621621622)
     ('E', 0.7127008416220352)
     ('R', 0.7453866259829712)
     ('I', 0.6365554141009611)
     ('A', 0.8206845238095238)
     ('E', 0.7520746887966805)
     ('Y', 0.626965110163053)
```

Colab paid products - Cancel contracts here

✓ 0s completed at 11:12 PM