Caitlyn Jones

Prof. Yangyang Wang

MATH 40A: Intro to Applied Math

9 October 2024

Homework 2

1. Information theory worksheets 1 and 2

2. We'll repeat the guessing game, but now using max daily temperatures from Boston over the last 82 years as our random numbers (`https://www.ncdc.noaa.gov/cdo-web/`). As before, you may only ask yes or no questions to determine the temperature. Draw samples by running `random_day_temperature.py`. You will need the files `BostonTemps.txt` and `BostonDates.txt`.

   Guess the temperature with and without knowing the month.

| Trial | Without month |
|---|---|
| 1 | 6 qs |
| 2 | 7 qs |
| 3 | 7 qs |

| Trial | With month |
|---|---|
| 1 | 6 qs |
| 2 | 7 qs |
| 3 | 7 qs |

   (a) Max temperatures in Boston range from 6 to 103 °F. How many questions would you expect to have to ask? around 15 questions

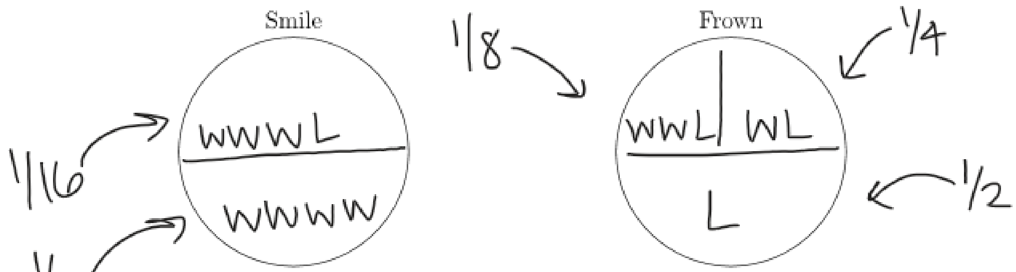
   (b) Are you doing better or worse than that? Why? We did way better than expected

2. We will continue the scenario from question 1. Let's say you run into me the next morning. A smile means we played four games, a frown means we played fewer than four games. Let's calculate how much information is in that smile.

   We'll use the following expression for the uncertainty (or entropy) $H$ of a discrete random variable $X$:

   $$H = -\sum_i p_i \log_2 p_i.$$

   (a) Draw pie diagrams for both scenarios and add probabilities (as fractions).




   Tot. prob. of smile: _____ Tot. prob. of smile: _____

   (b) What is the uncertainty in the outcome of the game given that you observed a smile?

   $$H = -\sum_i p(x_i) \log_2 p(x_i)$$

   $$H = -\left(\frac{1}{16}\log_2\left(\frac{1}{16}\right) + \frac{1}{16}\log_2\left(\frac{1}{16}\right)\right) = \frac{1}{2}$$

   (c) Using the probabilities from the "frown" pie chart, what is the uncertainty in the outcome given a frown?

   $$H = -\left(\frac{1}{2}\log_2\left(\frac{1}{2}\right) + \frac{1}{4}\log_2\left(\frac{1}{4}\right) + \frac{1}{8}\log_2\left(\frac{1}{8}\right)\right) = 1.375$$

   (d) What is the average uncertainty in the outcome of the game, given that you saw me the next morning? (Hint: this is a weighted average of your answers to (b) and (c)).

   smile prob $= \frac{1}{16} + \frac{1}{16} = \frac{1}{8}$

   frown prob $= \frac{1}{2} + \frac{1}{4} + \frac{1}{8} = \frac{7}{8}$

   average $= \frac{1}{8} \cdot \frac{1}{2} + \frac{7}{8} \cdot 1.375$

   $= 1.265625$

2.

a. To calculate the Shannon entropy of each digit, I first created a function `entropy(N)` that will compute the entropy of a given N.

```
def entropy(N):
    return np.log2(N)
```

Then, I needed to determine what N will be for equation. For the first digit, N is 2 because the only two possible first digits are 0 and 1.

```
first_entropy = entropy(2)
```

```
The entropy of the first digit is 1.0
```

For the second digit, N is 5, as the second digit ranges from [0,4].

```
second_entropy = entropy(5)
```

```
The entropy of the second digit is 2.321928094887362
```

N is 10 for the third digit because N ranges from [0,10].

```
third_entropy = entropy(10)
```

```
The entropy of the third digit is 3.321928094887362
```

b. To calculate the sum of all three entropies, I created a variable `entropy_sum` that will compute the sum.

```
entropy_sum = first_entropy + second_entropy + third_entropy
```

```
The sum of all three entropies is 6.643856189774724
```

Then I calculated `total_entropy`.

```
total_entropy = entropy(150)
```

```
The total entropy is 7.22881869049588
```

The sum of entropies is different from the total entropy because there are restrictions placed on the second two digits if the first digit is a 0 versus if the first digit is a 1. However, `total_entropy` represents the randomness that takes place in selecting a number from the 150 different possibilities, which is why this number is greater than `entropy_sum`. These differences occur because `entropy_sum` has restrictions in place while `total_entropy` does not.

3.

a. To inspect the 100 most common word pairs in the English language, I first had to read in the four given text files which contain the words, the indices of the common pairs, and the frequencies of the common pairs.

```
words = open("words.txt", "r").read().split()
frequencies = open("frequencies.txt", "r").read().split()
index1 = open("word1_index.txt", "r").read().split()
index2 = open("word2_index.txt", "r").read().split()
```

Then, I created variable `frequency` that will track the index of the frequency of the word pairs. Next, I created variable `word_pairs` that will hold the word pairs based off their corresponding word and index from the various text files.

```
frequency = [int(f) for f in frequencies]
word_pairs = [(words[int(index1[i])], words[int(index2[i])]) for i in range(len(index1))]
```

From there, I was able to print the 100 most common word pairs.

```
for i in range(100):
    print(f"{word_pairs[i][0]} -> {word_pairs[i][1]}: {frequency[i]}")
```

For the sake of space, I will not put all 100 pairs, but here are the first 5.

```
of -> the: 2586813
in -> the: 2043262
to -> the: 1055301
on -> the: 920079
and -> the: 737714
```

It makes sense that these words pairs are the 100 most common; however, the contractions are interesting because technically “n’t” is not a word, but the prefix with “n’t” is still among the 100 most common word pairs.

b. To write the 20 most common word pairs, I created a variable `twenty_common` that will hold the top 20 values from `word_pairs`.

```
twenty_common = [(word_pairs[i][0], word_pairs[i][1], frequency[i]) for i in range(20)]
```

From there, I was able to print all 20 word pairs.

```
for word_one, word_two, freq in twenty_common:
    print(f"{word_one} -> {word_two}: {freq}")
```

```
of -> the: 2586813
in -> the: 2043262
to -> the: 1055301
on -> the: 920079
and -> the: 737714
to -> be: 657504
at -> the: 617976
for -> the: 616400
in -> a: 544137
do -> n't: 537718
with -> the: 477541
from -> the: 472490
it -> was: 466117
of -> a: 444998
that -> the: 399074
as -> a: 365847
is -> a: 364083
going -> to: 363524
by -> the: 361826
and -> i: 345078
```

I used the built-in `DiGraph()` function to create the directed graph diagram that will have arrows from every first to in the pair to the second word in the pair.

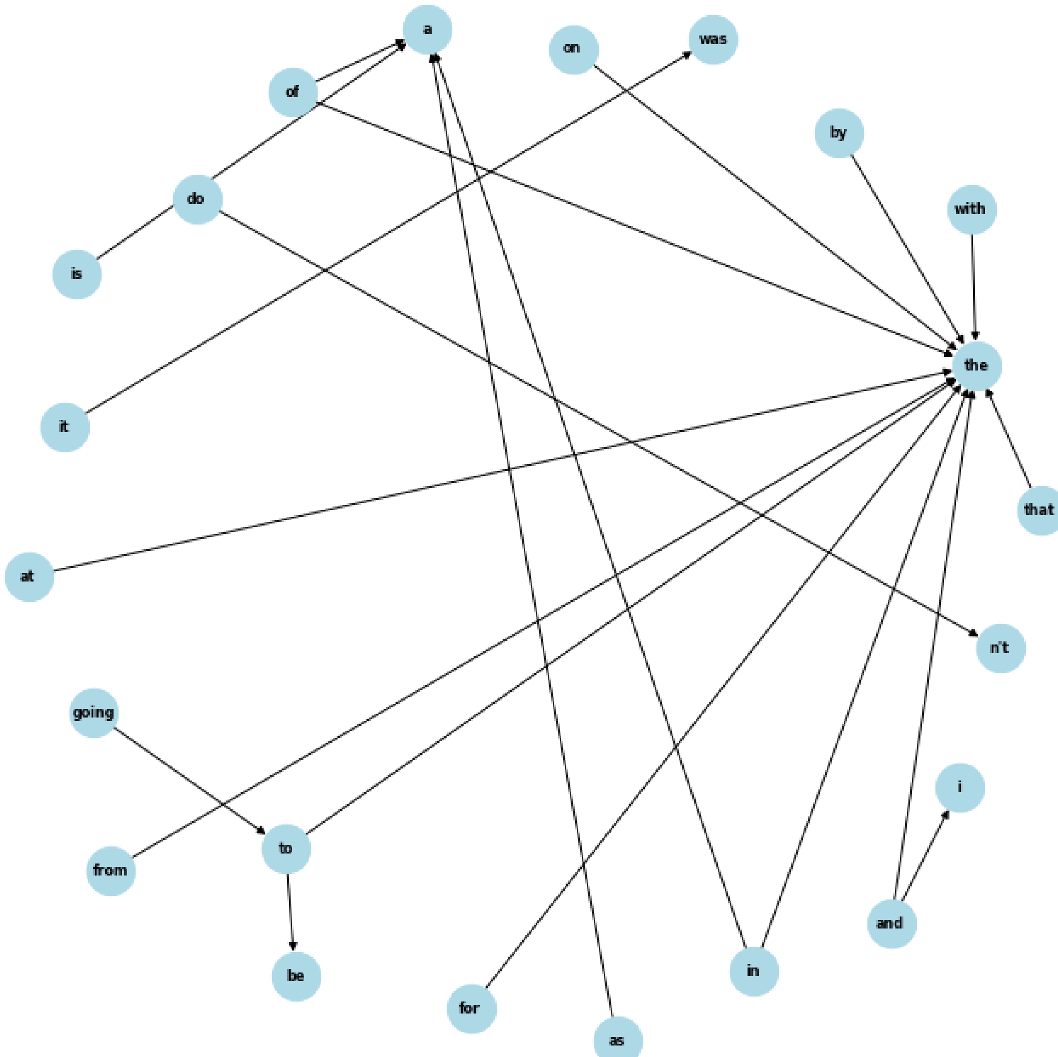
```
G = nx.DiGraph()

for word_one, word_two, freq in twenty_common:
    G.add_edge(word_one, word_two)

plt.figure(figsize = (9,9))
pos = nx.spring_layout(G, k=2.0)

nx.draw(G, pos, with_labels=True, node_size=900, node_color="lightblue", font_size=8, font_weight='bold', arrows=True)

plt.show()
```




I was able to generate the longest word sequence by calling `nx.dag.longest_word_path` on this graph.

```
print(nx.dag_longest_path(G))
```

```
['going', 'to', 'the']
```

c. No, I cannot travel to every word through the arrows, as some of the pairs are incredibly common, but have no connection to the other common pairs. For example, "do → n't" is in the top 20 common pairs, yet I can only access this pair through "do".

4. We can derive $\mathrm{H(X, Y) = H(X|Y) + H(Y)}$ given that

$$\mathrm{H(X,Y)} = -\sum_{x,y} \mathrm{p(x,y)logp(x,y)}$$

By the definition of conditional probability, $\mathrm{p(x,y) = p(x|y)p(y)}$.

$$\mathrm{H(X,Y)} = -\sum_{x,y} \mathrm{p(x,y)\log\big(p(x|y)p(y)\big)}$$

By logarithmic properties, we can rewrite this equation.

$$\mathrm{H(X,Y)} = -\sum_{x,y} \mathrm{p(x,y)\big(log(p(x|y) + logp(y)\big)}$$

Using distributive properties, we can expand this equation.

$$\mathrm{H(X,Y)} = -\sum_{x,y} \mathrm{p(x,y)logp(x|y)} + -\sum_{x,y} \mathrm{p(x,y)logp(y)}$$

We can recognize the first equation as $\mathrm{H(X|Y)}$.

$$\mathrm{H(X,Y) = H(X|Y)} + -\sum_{x,y} \mathrm{p(x,y)logp(y)}$$

Since $\mathrm{logp(y)}$ does not depend on x, we can factor it out.

$$\mathrm{H(X,Y) = H(X|Y)} + -\sum_{y} \mathrm{p(y)logp(y)}$$

Now, we can recognize the second equation as $\mathrm{H(Y)}$.

$$\mathrm{H(X,Y) = H(X|Y) + H(Y)} \qquad \blacksquare$$

5. Given $p_0 = p_1 = \frac{1}{2}$ and the equation

   $$H(X) = -\sum_i p(x_i)\log_2 p(x_i)$$

   We can assume, that for two symbols,

   $$H(X) = -(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2})$$

   $$H(X) = -(\frac{1}{2}(-1) + \frac{1}{2}(-1)$$

   $$H(X) = -(-1)$$

   $$H(X) = 1$$

   From here, we can compute

   $$I(X;Y) = H(X) - H(Y|X)$$

   $$I(X;Y) = 1 - 1$$

   $$I(X;Y) = 0$$

   Therefore, the rate of transmission is 0.