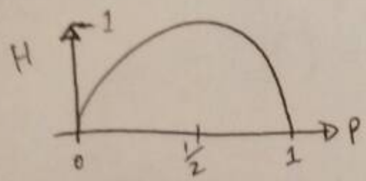
HW#2:

1) a) $P(W = w_i) = \frac{1}{N}$

$$H(w) = -\sum_{w \in V} P(w) \log P(w)$$



$$H(w) = -\sum_{w \in V} \frac{1}{N} \log \frac{1}{N} = \sum_{w \in V} \frac{1}{N} \log N$$

$$= \cancel{N} \left(\frac{\cancel{1}}{\cancel{N}} \log N \right) = \log N$$

If the number of unique words $N = 1$ (i.e all words are identical)

then $H(w) = \log 1 = 0$

$\therefore$ minimum $H(w) = 0$; Maximum $H(w) = \log N$

b) Sample Minimum $H(w)$ article $= \{ w_1, w_1, w_1, w_1, w_1 \}$
(e.g. maximally homogeneous)

Sample maximum $H(W)$ article $= \{ w_2, w_1, w_3, w_4, w_5, w_6, w_3, w_5 \}$
(eg. maximally heterogeneous)

c) Two article which has $H(w) = 0$ probably mens that the documents themselves Contain only one unique word each.

For example, $D_1 = \{ w_1, w_1, w_1, \ldots, w_1 \}$ and $D_2 = \{ w_2, w_2 \ldots w_2 \}$

in that case, when combining the documents, the most distinct set that you would get is $\{ w_1, w_2 \}$ so the maximum entropy for A_3 is $\log 2 = 0.69$

2) a) $H(X|X) = -\sum_{x\in\Omega_x}\sum_{y\in\Omega_y} p(x)\, \cancel{p(x|x)}^{\,1} \underbrace{\log \cancel{p(x|x)}^{\,1}}_{=0} = 0$

$p(x|x)$: probability of x given x is 1

$$H(X|X) = 0$$

when the two random variable under comparison is the same, then there is no extra information required for communicating Y given X

b) $I(X;Y)$ = mutual information. = how much reduction in uncertainty given info about Y

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

If X & Y are independent then $H(X|Y) = H(X)$

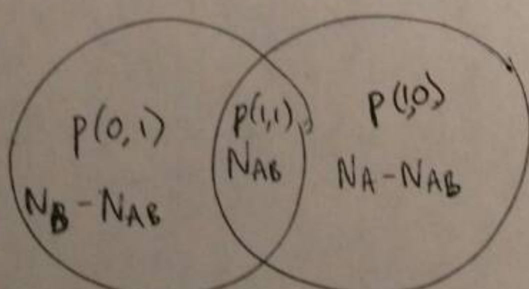
∵ having the additional info ~~about~~ ~~the~~ given Y will not provide you with any additional info

$$I(X;Y) = H(x) - H(x) = 0 \quad \text{iff } X, Y \text{ independent}$$

---

3) a) $p(X_A = 0, X_B = 1) = \dfrac{N_B - N_{AB}}{N}$

$$p(X_A = 0, X_B = 0) = \frac{N - (N_A + N_B - N_{AB})}{N} = \frac{N - N_A - N_B + N_{AB}}{N}$$



HW2

September 4, 2016

```
In [1]: %pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

```
In [2]: df = open("cacm.trec.filtered.txt")
        data = df.readlines()
```

```
In [3]: corpus = [_.split() for _ in data]
```

```
In [4]: wordlist = unique(list(flatten([_.split() for _ in data])))
```

total of 88700 words and 1845 unique words

```
In [6]: import itertools
        all_word_combos = list(itertools.combinations(wordlist,2))
```

```
In [7]: from tqdm import tqdm
        dict_counts ={}
        for wordpair in tqdm(all_word_combos):
            for doc in corpus:
                if (wordpair[0]  in doc and wordpair[1] in doc ):
                    if wordpair in dict_counts:
                        dict_counts[wordpair]+=1
                    else:
                        dict_counts[wordpair]=1
```

```
In [8]: top10keys = sorted(dict_counts, key=dict_counts.get, reverse=True)[:10]
```

```
In [18]: for i in top10keys:
             print str(i) +":"+ str(dict_counts[i])
```

```
('january', 'paper'):181
('language', 'programming'):153
('january', 'time'):150
('january', 'program'):149
('january', 'systems'):149
('data', 'january'):142
('january', 'presented'):141
('january', 'programming'):139
('program', 'programs'):133
('january', 'method'):125
```

1 3c) Mutual information

```
In [9]: X  = wordpair[0]
        Y  = wordpair[1]
```

```
In [10]: all_words = list(flatten([_.split() for _ in data]))
```

$$I(X;Y) = \sum_{x,y} p(x,y) log \frac{p(x,y)}{p(x)p(y)}$$

```
In [11]: def p(x):
             return (all_words.count(X)+0.5)/(len(all_words)+1.)
```

```
In [12]: p(X)*p(Y)
```

```
Out[12]: 4.3499716321668095e-08
```

```
In [13]: wordpair
```

```
Out[13]: ('yields', 'zeros')
```

```
In [14]: MI_dict = {}
```

```
In [15]: def pAB(wp):
             return dict_counts[wp]+0.25/(len(all_words)+1)
```

```
In [16]: for wp in tqdm(dict_counts):
             MI_dict[wp] = pAB(wp)*log(pAB(wp)/p(X)*p(Y))
```

```
In [19]: top10MI = sorted(MI_dict, key=MI_dict.get, reverse=True)[:10]
```

```
In [21]: for i in top10MI:
             print str(i) +":"+ str(MI_dict[i])
```

```
('january', 'paper'):940.927980129
('language', 'programming'):769.65701897
('january', 'time'):751.595311055
('january', 'program'):745.588016508
('january', 'systems'):745.588016508
('data', 'january'):703.727458966
('january', 'presented'):697.77516131
('january', 'programming'):685.891893431
('program', 'programs'):650.416450655
('january', 'method'):603.53923359
```

('january', 'paper'):181 ('language', 'programming'):153 ('january', 'time'):150 ('january', 'program'):149 ('january', 'systems'):149 ('data', 'january'):142 ('january', 'presented'):141 ('january', 'programming'):139 ('program', 'programs'):133 ('january', 'method'):125

i) The top 10 word pairs with the highest mutual information is the same as the top 10 pairs based on the co-occurence counts.

ii) The top 5 words which have the highest mutual information with the word "programming" are:

```
In [23]: topMIs = sorted(MI_dict, key=MI_dict.get, reverse=True)[:30]
         for i in topMIs:
             if i[0] =='programming' or i[1] =='programming':
                 print str(i) +":"+ str(MI_dict[i])
```

```
('language', 'programming'):769.65701897
('january', 'programming'):685.891893431
('program', 'programming'):586.090583816
('paper', 'programming'):562.940801951
('languages', 'programming'):545.667210954
```

The results are somewhat reasonable, programming language(s), program, programming all have word meaning that makes sense. We saw previously that january is a common word that appears with many other words, (possibly because this is a subset of articles that have January listed in its publication date), so it is not surprising that january also occur frequently with programming. ('paper', 'programming') is a bit surprising, but maybe the word 'paper' occurs frequently in the same phrase common throughout the collection (e.g. 'Paper published on July 1, 2016', 'Paper accepted to CACM journal' ..etc), so any papers containing the word 'programming' will be counted as co-occurence.