Assignment 2

Hincu Alice

Problem 1

Compute variable byte codes and γ codes for the postings list < 777, 17743, 294068, 31251336 >.
Use gaps instead of docIDs where possible. Write binary codes in 8-bit blocks.
You can use Google, or any other resource, to convert numbers to binary.

17743-777=16966 ; 294068-17743=276325 ; 31251336-294068=30957268 ;
For the postings list <777, 17743, 294068, 31251336>, using gaps instead of docIDs, the gap-encoded postings list is <777, 16966, 276325, 30957268>

Variable byte codes

Dedicate 1 bit (high bit) to be a continuation bit c. If the gap G fits within 7 bits, binary-encode it in the 7 available bits and set c = 1. Else: encode lower-order 7 bits and then use one or more additional bytes to encode the higher order bits using the same algorithm. At the end set the continuation bit of the last byte to 1 (c = 1) and of the other bytes to 0 (c = 0).

After we convert each number to its binary form (and add the '|' separator for each 7 bits), we create the variable byte for each number:

- 777 = 0110|0001001:
    - Get the last byte and add 1 as the head bit (0001001 -> 10001001)
    - Get the remaining byte and add 0 as the head bit (110 -> 00000110)
    - 00000110 10001001 = 6 137
- 16966 = 01|0000100|1000110
    - Get the last byte and add 1 as the head bit (1000110 -> 11000110)
    - Get the next byte and add 0 as the head bit (0000100 -> 00000100)
    - Get the remaining byte and add 0 as the head bit (01 -> 00000001)
    - 00000001 00000100 11000110 = 1 4 198
- 276325 = 010000|1101110|1100101
    - Get the last byte and add 1 as the head bit (1100101 -> 11100101)
    - Get the next byte and add 0 as the head bit (1101110 -> 01101110)
    - Get the remaining byte and add 0 as the head bit (010000 -> 00010000)
    - 00010000 01101110 11100101 = 16 110 229
- 30957268 = 01110|1100001|0111101|1010100
    - Get the last byte and add 1 as the head bit (1010100 -> 11010100)
    - Get the next byte and add 0 as the head bit (0111101 -> 00111101)
    - Get the next byte and add 0 as the head bit (1100001 -> 01100001)
    - Get the remaining byte and add 0 as the head bit (01110 -> 00001110)
    - 00001110 01100001 00111101 11010100 = 14 97 61 212

docIDs	777	17743	294068	31251336
gaps		16966	276325	30957268
VB code	10001001 00000110	00000001 00000100 11000110	00010000 01101110 11100101	00001110 01100001 00111101 11010100

<u>Gamma codes</u>

Represent a gap G as a pair of length and offset. Offset is the gap in binary, with the leading bit chopped off. Length is the length of offset. Encode length in unary code. Gamma code is the concatenation of length and offset:

- 777 = 01100001001
    - 100001001 after chopping
    - length = 9 => 1111111110 (unary representation)
    - 1111111110100001001
- 16966 = 0100001001000110
    - 00001001000110 after chopping
    - length = 14 => 111111111111110 (unary representation)
    - 11111111111111000001001000110
- 276325 = 01000011011101100101
    - 000011011101100101 after chopping
    - length = 18 => 1111111111111111110 (unary representation)
    - 1111111111111111110000011011101100101
- 30957268 = 01110110000101111011010100
    - 110110000101111011010100 after chopping
    - length = 24 => 1111111111111111111111110 (unary representation)
    - 1111111111111111111111110110110000101111011010100

number	unary code	length	offset	γ code
777	1111111110	9	100001001	1111111110, 100001001
16966	1111111111 11110	14	0000100100 0110	111111111111110,00001001000 110
276325	1111111111 111111110	18	0000110111 01100101	1111111111111111110,0000110 11101100101
30957268	1111111111 1111111111 11110	24	1101100001 0111101101 0100	1111111111111111111111110,11 0110000101111011010100

Problem 2

1. We define the idf weight of term t as follows:

$idf_t = log_{10}(N/df_t)$, N = number of documents, dft = document frequency.

If a term occurs in every document, then dft=N. Therefore, the IDF becomes:

$idf_t = log_{10}(N/N) = log_{10}(1) = 0$ => if a term is present in every document, it does not provide any special information that can be used to differentiate one document from another. So, by putting this term on the stop list, it has the same effect as idf weighting: the word is ignored.

2.

In the formula $idf_t = log_{10}(N/df_t)$, the base "10" is used. We change it to a base b, where b>0. For that, we use the Logarithm Change of Base Formula:

$(log_a b) = (log_c b) / (log_c a)$ => $(log_{10}(N/df_t)) = log_b(N/df_t) / (log_b 10)$ (1)

Now instead of the log_{10} formula for idf_t, we replace it with log_b(because this is what it is asked of us), and continue with the (1) equation. So, for any base b>0:

$idf_t = log_b(N/df_t) = (log_b 10) * (log_{10}(N/df_t)) = c * log_{10}(N/df_t)$ (2)

where c = $(log_b 10)$, c is a constant.

When the base of the logarithm changes, the tf-idf score becomes (we use (2)):

$$tfidf_{t,d,b} = tf_{t,d} * idf_t = tf_{t,d} * c * (log_{10}(N/df_t))$$

But we know from the initial formula that $idf_t = log_{10}(N/df_t)$, so the eq becomes:

$$tfidf_{t,d,b} = tf_{t,d} * c * idf_t = c * tfidf_{t,d,b}$$

The score of the document d for the query q with a different base b is:

Score(q,d,b) = $\sum tfidf_{t,d,b}$ = c*$\sum tfidf_{t,d,b}$

So changing the base changes the score by a factor c = $log_b 10$

The relative scoring of the documents remains unaffected by changing the base since the scaling is uniform across all terms and documents.

Problem 3

1. The harmonic mean tends to be closer to the smaller of the two values (precision and recall). This means that if either precision or recall is low, the F1 score will also be low, reflecting a need for improvement in that area. The arithmetic mean gives an average that is more influenced by the higher value among precision and recall. This can mask the effect of a very low value in one of the metrics, leading to a deceptively high overall score. In practical scenarios, particularly in information retrieval or classification tasks, having extremely high recall (e.g., by returning all documents) can lead to very low precision (most returned documents are irrelevant). The harmonic mean (F1 score) in such cases would be much lower than the arithmetic mean, providing a more realistic assessment of the system's performance.

2. Precision = (nr relevant items retrieved)/(nr retrieved items) = 8/(10+8)=8/18=0.44

Recall = (nr relevant items retrieved)/(nr relevant items) = 8/20 = 0.4

3.

$$MAP(Q) = \frac{1}{|Q|}\sum_{j=1}^{|Q|}\frac{1}{m_j}\sum_{k=1}^{m_j} Precision(R_{jk})$$

System 1: RNRNN NNNRR

- Precision at 1st relevant document: 1/1
- Precision at 2nd relevant document: 2/3
- Precision at 3rd relevant document: 3/9
- Precision at 4th relevant document: 4/10

MAP = ¼ * (1/1 + 2/3 + 3/9 + 4/10) = ¼ * (1 + (60+30+36)/90) = ¼ * (1+126/90) = 0.6

System 2: NRNNR RRNNN

- Precision at 1st relevant document: 1/2
- Precision at 2nd relevant document: 2/5
- Precision at 3rd relevant document: 3/6
- Precision at 4th relevant document: 4/7

MAP = ¼ * (½ + ⅖ + 3/6 + 4/7) = ¼ * (1 + (14+20)/35) = ¼ * (1 + 34/35) = 0.493

<u>Problem 4</u>

Y = voter is young
P = voter voted for Peach

- 30% younger voters => 70% older voters
  - **P(Y) = 0.3 => P(Y') = 1 - 0.3 = 0.7**
- 60% of young voters chose Princess Peach => 40% of young voters chose Bowser
  - **P(P|Y) = 0.6 => P(P'|Y) = 1 - 0.6 = 0.4**
- 35% of voters are old AND chose Princess Peach
  - **P(P ∩ Y') = 0.35**

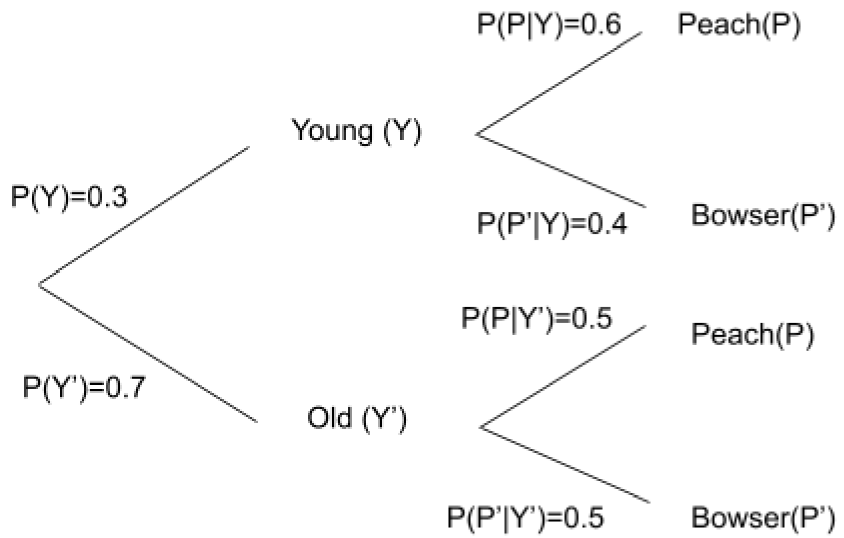
1. P(Y) = 0.30 (30% younger voters)
P(P|Y) = 0.60 (60% of young voters chose Princess Peach)
P(P∩Y') = 0.35 (35% of voters are old and chose Princess Peach)

2. We first start with the Y event: a branch for people under 30 (Y), and a branch for people over 30 (Y'). From each of these branches, there will be another 2 branches for the event P.

First branch calculations (all calculated above):
P(Y) = 0.3
P(P|Y) = 0.6
P(P'|Y) = 1 - P(P|Y) = 0.4

Second branch calculations (all calculated above):
P(Y') = 0.7
P(P|Y') = P(P ∩ Y') / P(Y') = 0.35/0.7 = 0.5
P(P'|Y') = 1 - P(P|Y') = 0.5

Tree:



3. $P(Y \cap P) = P(Y) * P(P|Y) = 0.3 * 0.6 = 0.18$

4. $P(Y|P) =$

We use Bayes Rule to reverse conditional probabilities.

$P(Y|P) = P(Y \cap P) / P(P) = 0.18 / ?$

$P(P) = P(Y) * P(P|Y) + P(Y') * P(P|Y') = 0.3 * 0.6 + 0.7*0.5 = 0.18 + 0.35 = 0.53$

$\Rightarrow P(Y|P) = 0.18 / 0.53 = 0.033962$

<u>Problem 5</u>

$$P(q|d) \propto \prod_{1 \leq k \leq |q|} (\lambda P(t_k|M_d) + (1-\lambda)P(t_k|M_c))$$

Doc1: the best **movie** should be a meaningful **movie**
Doc2: summer of soul is the best **movie** of the year

1. P(movie|Doc2)? λ = 0.5.

Doc 1 -> 8 words, “movie” appears 2 times.
Doc 2 -> 10 words, “movie” appears 1 time.

P(movie|Doc2) = (1/10 + 3/18)/2 = (0.1 + 0.1(6)) / 2 = 0.1(3) (aprox 0.134)

2. P(movie|best, Doc2) = ? $\lambda 1 = \lambda 2 = \lambda 3 = ⅓$

P(movie | best,Doc2) = $\lambda 1 \times$P(movie|best, Doc2) + $\lambda 2 \times$P(movie|best, Collection) + $\lambda 3 \times$P(movie|Collection)

- P(movie|best, Doc2) = $\frac{Count(movie, best)\ in\ Doc2}{Count(best)\ in\ Doc2}$ = $\frac{frequency\ of\ \text{"}best\ movie\text{"}\ in\ Doc2}{frequency\ of\ \text{"}best\text{"}\ in\ Doc2}$ = $\frac{1}{1}$ = 1
- P(movie|best, Collection) = $\frac{Count(movie, best)\ in\ Collection}{Count(best)\ in\ Collection}$ = $\frac{frequency\ of\ \text{"}best\ movie\text{"}\ in\ Collection}{frequency\ of\ \text{"}best\text{"}\ in\ Collection}$ = $\frac{2}{2}$ = 1
- P(movie|Collection) = $\frac{frequency\ of\ \text{"}movie\text{"}\ in\ Collection}{count\ of\ terms\ in\ Collection}$ = $\frac{3}{18}$

P(movie|best, Doc2) = ⅓ * (1 + 1 + 3/18) = 0.7(2) (aprox 0.73)

Problem 6

Doc1: great action movie
Doc2: bad action movie bad action movie
Doc3: great bad bad bad movie

Doc1 is labeled Positive. Doc2 and Doc3 are both labeled Negative.

1. Generate the feature matrix X and label vector y for this dataset. The features are individual words, with their values being their frequency in the corresponding document. Use y = 1 for positive reviews, and y = 0 for negative ones.

The features are: great, action, movie, bad
Frequencies:

- **Doc1** (Positive): great (1), action (1), movie (1), bad (0)
- **Doc2** (Negative): great (0), action (2), movie (2), bad (2)
- **Doc3** (Negative): great (1), action (0), movie (1), bad (3)

Feature matrix X (One example per row, one feature per column) :

great	action	movie	bad
1	1	1	0
0	2	2	2
1	0	1	3

Label vector y:

Label
1
0
0

2. Given the training dataset you constructed above, compute the parameters of a Perceptron model, w and b, after one training epoch. Both w and b are initialized with 0s before training. Trace each iteration of the learning algorithm.

w = 0
b = 0
while not converged -> iterate over dataset:

- For Doc1:
    - Feature vector x1 = [1, 1, 1, 0]
    - True label y1 = 1
    - Decision d = w*x1 + b = (0*1 + 0*1 + 0*1 + 0*0) + 0 = 0 => return NO
    - Since d≠y1 (We made a mistake on the positive label), we update w and b:
        - b = b+1 = 1
        - w = w + x1 = [1,1,1,0]
- For Doc2:
    - Feature vector x2 = [0, 2, 2, 2]
    - True label y2 = 0
    - Decision d = w*x2 + b = (1*0 + 1*2 + 1*2 + 0*2) + 1 = 5 => return YES
    - Since d≠y2 (We made a mistake on the negative label), we update w and b:
        - b = b-1 = 0
        - w = w - x2 = [1-0,1-2,1-2,0-2] = [1, -1, -1, -2]
- For Doc3:
    - Feature vector x3 = [1, 0, 1, 3]
    - True label y3 = 0
    - Decision d = w*x3 + b = (1*1 + (-1)*0 + (-1)*1 + (-2)*3) + 0 = -6 => return NO
    - Since d=y3, we do not update anything

== epoch finished ==

- Weights w=[1, -1, -1, -2]
- Bias b=0