Elliot Linsey 210764484

ECS736P_210764484

Question 1:

a) Boolean models rely on whether a term is present in a document or not using a binary system of 0 and 1. A query is defined by the Boolean expressions of and, or, not. It requires precise semantics and formal entry. Due to the binary nature of the term weights, there is no possibility to rank documents as they either match the query, or do not. An extension of this to allow partial matching is the extended Boolean model, which takes some features from the vector space model of term weighting and algebraic distance to obtain rankings for Boolean queries.
The vector space model utilises algebraic measures such as cosine similarity (the angle between vectors) to calculate the similarity between a document vector and query vector. It utilises non-binary weights such as tf-idf to allow for partial matching. An extension of this is generalised vector space model which introduces patterns of co-occurrence of terms within our document corpus.
Probability language models utilise the likelihood of seeing a query given a document language model. This can be in a unigram model that assumes word independence and is known as the 'bag of words' model. From this, we can calculate the probability of seeing a query term within each document, then generate a score and rank all documents with the highest being the document with the highest probability of generating this query. To avoid zero probability issues, smoothing methods such as Jelinek-Mercer and Dirichlet add parameters that can alter the retrieval function.

b)

	Yoda	Hope	Force	Jedi
D1	0	0	0	1
D2	1	0	0	2
D3	1	0	0	1
D4	0	0	1	1
D5	0	2	0	1
D6	0	1	0	1

c) We use cosine similarity as this takes the angle between vectors, with an angle of 0 displaying maximal similarity. This is better than using a measure such as Euclidean distance as these distances can be quite large. We can also perform length normalisation with cosine which makes long and short documents have comparable weights.

Query = ['Yoda','Hope','Force','Jedi] or [1,1,1,1]

D1 = (0x1) + (0x1) + (0x1) + (1x1) = 1
D2 = (1x1) + (0x1) + (0x1) + (2x1) = 3
D3 = (1x1) + (0x1) + (0x1) + (1x1) = 2
D4 = (0x1) + (0x1) + (1x1) + (1x1) = 2
D5 = (0x1) + (2x1) + (0x1) + (1x1) = 3
D6 = (0x1) + (1x1) + (0x1) + (1x1) = 2

Ranking order = D2,D5,D3,D4,D6,D1

Question 2:

Elliot Linsey 210764484

a) The zero probability problem occurs when a term in a query does not appear in a document. The model will assign a zero probability to this event and to any set of events including the unseen event. To deal with this, the technique of smoothing is applied, an example is Jelinek-Mercer smoothing. An alpha value is added and the probability of seeing the term in the collection model is also added. This means that while you may not see the term in the document model, if it appears in the language collection model it will be assigned a probability of occurring and thus it would not have a probability of 0.

bi) The most common types of language models are unigram and bigram models. Only unigram models assume word independence, for any n-gram model with n greater than 1 (such as bigram where n = 2) they do not assume word independence.

Bii) query = [t3,t1,t2,t1]
M1 = 1/12 x 4/12 x 7/12 x 4/12 = 0.0054
M2 = 3/6 x 1/6 x 2/6 x 1/6 = 0.00463
M3 = 6/15 x 5/15 x 4/15 x 5/15 = 0.0119
M4 = 4/15 x 7/15 x 4/15 x 7/15 = 0.0155
ranking order = M4,M3,M1,M2

c)

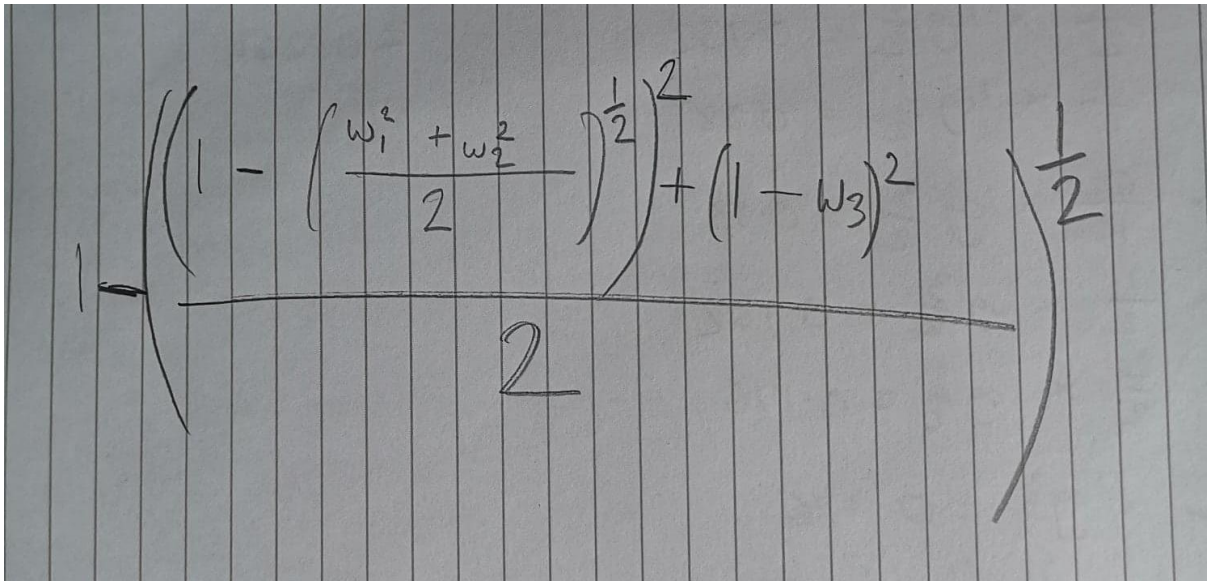
	Relevant documents containing ti	Number of documents containing ti
T1	4	15
T2	8	16
T3	4	14

N = 44
R = 20
r1 = 4
r2 = 8
r3 = 4
n1 = 15
n2 = 16
n3 = 14
a1 = 4/20 = 0.2
a2 = 8/20 = 0.4
a3 = 4/20 = 0.2
b1 = 15-4/44-20 = 0.458
b2 = 16-8/44-20 = 0.333
b3 = 14-4/44-20 = 0.417
c1 = 0.2x(1-0.458)/0.458x(1-0.2) = 0.189
c2 = 0.4x(1-0.333)/0.333x(1-0.4) = 0.481
c3 = 0.2x(1-0.417)/0.417x(1-0.2) = 0.224
R(D) = 0.189*t1 + 0.481*t2 + 0.224*t3
R(d1) = 0.189*1 + 0.481*1 = 0.670
R(d2) = 0.189*1 + 0.481*1 + 0.224*1 = 0.894
R(d3) = 0.189*1 = 0.189
Ranking order = d2,d1,d3

Elliot Linsey 210764484

Question 3:

a)



topleft = (1-(0.5)**0.5)**2
topright = (1-1)**2
combinedtop = (topleft+topright)/2
full = 1-combinedtop**0.5
R(d|q) = 0.793

bi) it assumes that the index terms are linearly independent but no longer pairwise orthogonal
bii) minterm vectors
biii) (1,0,...0), (0,1...,0), (0,0...1)

ci) m1 = (1,1,0,0), m2 = (1,1,1,1), m3 = (0,0,1,1), m4 = (0,1,1,1)
cii) m1v = (1,0,0,0), m2v = (0,1,0,0), m3v = (0,0,1,0), m4v = (0,0,0,1)
c2,1 = w2,1 + w2,2, + w2,8 = 1 + 5 + 1 = 8
c2,2 = 3 + 1 = 4
c2,3 = 0

c2,4 = 4 + 1 = 5
C = (8**2 + 4**2 + 5**2)**0.5 = 10.247
t2v = [8,4,0,5]/10.247
t2v = [0.781, 0.390, 0, 0.488]

Question 4:

a) 15*0.33 = 5.
5/0.25 = 20
number of relevant docs = 20
b) Text based information retrieval relies on human annotation of the image where CBIR relies on computer vision techniques. The human annotation can be down to subjectivity of the annotator, different people may have different opinions on what is being conveyed in the image. Similarly, relying on computer vision for CBIR may result in labelling errors due to misidentification of an object or theme. The language and culture of annotators may have an effect, the use of synonyms such as 'film' and 'movie' may lead to different annotation and search results. CBIR can detect low level features such as colours, textures, edges, but there is a semantic gap between this low-level metadata and the high-level data that contains the semantic meaning. For example, an image representation of an apple to a computer could be displayed as a vector of pixel values, from this it may learn edges, shapes, textures and colours, however the ability to identify this as an apple (an abstract concept) can be difficult and this leads to the semantic gap. For text based retrieval, with a human annotator the most similar concept would be different opinions on what an image contains, if an image contains somebody screaming one annotator may label it as 'fear' and the other may be 'joy', these are both vastly different concepts.

ci) D,C
cii) D, E
ciii) C,D
civ) C,D
v) A,E,B