INFORMATION RETRIEVAL

CSE 571

MID TERM EXAMINATION

SHWETA SOOD

2012164

Q1 We have two different information retrieval systems. The first system has a Mean Average Precision score of 0.5 and the second system has a Mean Average Precision Score of 0.55.

Average precision seen at relevant documents is averaged over all the queries and reported as a single score called the Mean Precision Score. A high mean average precision helps in retrieving relevant documents fast.

1. When we look at the two given information retrieval systems, we are not sure for **how many queries they have been tested**, or even if they have been tested for the same query or not while finding MAP. This is crucial as we cannot compare systems based on results of a few queries that may or may not be same for the two systems. (assuming corpus is same for the two systems)
2. Mean Average Precision score uses **binary relevance value (0,1)** for the documents. It cannot consider the ranking of documents across relevant levels in a range. So, there is a possibility that system 1 has retrieved less number of relevant documents compared to system 2 but these documents may be better in terms of quality of content relevant for the user. But MAP doesn't take this into account. Also, since there is binary relevance, the documents having less relevance may be marked as 0 by system 1 and 1 by system 2. This would lead to variation in their MAP values. If instead, the scoring was done using relevance in a range, a comparison could be drawn.

This means that we cannot conclude that system 2 is better than system 1 as MAP as a metric for comparison is not reliable or sufficient.

Q2 Given a query the IR System A returns following document ids= {d11, d12, d13, d14, d15, d16, d17}. For the same query 'q', the IR system B returns the following document ids = {d41, d32, d13, d16, d25}. We ask a user to assign scores to relevant documents: 1 as excellent match, 2 very good match, 3 as good match, 4 as fair match, and 0 as non-relevant.

For finding average precision and Precision at 5, I am taking 1,2,3,4 as 1 and 0 as 0 (as the system only takes binary levels)

1. Finding Average Precision

$$\text{AveP} = \frac{\sum_{k=1}^{n}(P(k) \times \text{rel}(k))}{\text{number of relevant documents}}$$

where:

AveP=average Precision

k=rank in the sequence of retrieved documents

n= number of retrieved documents

P(k)= precision at rank k

Average Precision for System A

Relevant docs retrieved	d11	d13	d16
Precision for relevant docs retrieved	1	2/3	3/6

Average Precision= (1+ 2/3+ 2/6) = **0.722**

Average Precision for System B

Relevant docs retrieved	d41	d13	d16
Precision for relevant docs retrieved	1	2/3	3/4

Average Precision= (1+ 2/3+ 3/4) = **0.805**

2. Finding Precision at 5

Precision at 5 for System A

Till 5th document retrieved, 2 are relevant (d11, d13). Thus, precision at 5= 2/5= **0.4**

Precision at 5 for System B

Till 5th document retrieved, 3 are relevant (d41, d13, d16). Thus, precision at 5= 3/5= **0.6**

3. Finding NDCG

$$\mathrm{DCG_p} = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{\log_2(i+1)}$$

where:

rel_i is the graded relevance of the result at position i

p=rank position upto which we consider documents

$$\mathrm{nDCG_p} = \frac{DCG_p}{IDCG_p}$$

where IDCG is value on sorting documents of a result list by relevance, producing the maximum possible DCG till position p

NDCG for System A

For NDCG values of relevance should be increasing from 0 (least relevant) to 4(most relevant). Hence, I am subtracting relevance value 1 to 4 from 5 (as relevance is decreasing from 1 to 4)

Doc	Rank	Relevance	$\log_2$(rank+1)	($2^{\text{relevance of doc}}$-1)/($\log_2$(rank+1))
d11	1	4	$\log_2 2$	15.5
d12	2	0	$\log_2 3$	0
d13	3	1	$\log_2 4$	0.5
d14	4	0	$\log_2 5$	0
d15	5	0	$\log_2 6$	0
d16	6	3	$\log_2 7$	2.493
d17	7	0	$\log_2 8$	0

DCG_7=17.99

Now for IDCG, sorting documents on best possible rank

Doc	Rank	Relevance	$\log_2$(rank+1)	($2^{\text{relevance of doc}}$-1)/($\log_2$(rank+1))
d11	1	4	$\log_2 2$	15
d16	2	3	$\log_2 3$	4.416
d13	3	1	$\log_2 4$	0.5
d12	4	0	$\log_2 5$	0
d14	5	0	$\log_2 6$	0
d15	6	0	$\log_2 7$	0
d17	7	0	$\log_2 8$	0

$IDCG_7$=19.916

$NDCG_7$=0.9032 ~ 0.90

NDCG for System B

For NDCG values of relevance should be increasing from 0 (least relevant) to 4(most relevant). Hence, I am subtracting relevance value 1 to 4 from 5 (as relevance is decreasing from 1 to 4)

Doc	**Rank**	**Relevance**	**$\log_2$(rank+1)**	**$(2^{\text{relevance of doc}}-1)/(\log_2(\text{rank}+1))$**
d41	1	3	$\log_2 2$	7
d32	2	0	$\log_2 3$	0
d13	3	1	$\log_2 4$	0.5
d16	4	3	$\log_2 5$	3.0146
d25	5	0	$\log_2 6$	0

DCG_7=10.5146

Now for IDCG, sorting documents on best possible rank

Doc	**Rank**	**Relevance**	**$\log_2$(rank+1)**	**$(2^{\text{relevance of doc}}-1)/(\log_2(\text{rank}+1))$**
d41	1	3	$\log_2 2$	7
d16	2	3	$\log_2 3$	4.416
d13	3	1	$\log_2 4$	0.5
d32	4	0	$\log_2 5$	0
d25	5	0	$\log_2 6$	0

$IDCG_7$=11.916

$NDCG_7$=0.88

Q3 Given three documents

d1 : “A B C”

d2 : “A B D”

d3 : “E F C”

and query, q : “A A C”

Term	d1 term frequency	d2 term frequency	d3 term frequency	Document Frequency $df_{t,d}$	$idf_{t,d} = \log_2(N/df_{t,d})$
A	1	1	0	2	0.584
B	1	1	0	2	0.584
C	1	0	1	2	0.584
D	0	1	0	1	1.584
E	0	0	1	1	1.584
F	0	0	1	1	1.584

Term	d1 tf-idf $tf_{t,d} * idf_{t,d}$	d2 tf-idf $tf_{t,d} * idf_{t,d}$	d3 tf-idf $tf_{t,d} * idf_{t,d}$
A	0.584	0.584	0
B	0.584	0.584	0
C	0.584	0	0.584
D	0	1.584	0
E	0	0	1.584
F	0	0	1.584

Term	Query term frequency	$idf_{t,d}$	query $tf_{t,d}$ * $idf_{t,d}$
A	2	0.584	1.168
B	0	0.584	0
C	1	0.584	0.584
D	0	1.584	0
E	0	1.584	0
F	0	1.584	0

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

where : A is a vector representation of 1 document and B vector representation of another document
Finding cos-sim(d1,q) = (0.584*1.168 + 0.584 * 0 + 0.584*0.584 + 0 + 0 + 0)/(0.584^2 + 0.584^2 + 0.584^2) * (1.168^2 + 0.584^2) = 1.023/1.32=0.7746~**0.775**
Finding cos-sim(d2,q) = (0.584*1.168 + 0.584 * 0 + 0*0.584 + 1.584*0 + 0 + 0)/(0.584^2 + 0.584^2 + 1.584^2) * (1.168^2 + 0.584^2) = 0.682/2.332=0.2924~**0.292**
Finding cos-sim(d3,q) = (0*1.168 + 0*0 + 0.584*0.584 + 0 +1.584* 0 +1.584* 0)/(0.584^2 + 1.584^2 + 1.584^2) * (1.168^2 + 0.584^2) = 0.3410/3.023=0.1128~**0.113**

Thus cosine similarity is highest for doc 1, followed by doc 2 and finally doc 3.

Q4 Query q = "A B A C D A B"
SE returns two documents
D1 = "B E A B"
D2 = "A F C"

Term	Query q_0 term frequency	D1 term frequency	D2 term frequency
A	3	1	1
B	2	2	0
C	1	0	1
D	1	0	0
E	0	1	0
F	0	0	1

$$\vec{q}_m = \alpha\vec{q}_0 + \beta\frac{1}{|D_r|}\sum_{\vec{d}_j \in D_r}\vec{d}_j - \gamma\frac{1}{|D_{nr}|}\sum_{\vec{d}_j \in D_{nr}}\vec{d}_j$$

where :
q_0 is the original query vector,
D_r and D_{nr} are the set of known relevant and non-relevant documents respectively
α, β, and γ are weights attached to each term

Taking $\alpha=1$, $\beta=0.75$, and $\gamma=0.25$
Using the above formula and the table providing document vectors, we can solve for the following parts:
Note – Negative weights are ignored and made 0

a. D1 relevant, D2 non-relevant

$q_m = [3,2,1,1,0,0]^T + 0.75 * [1,2,0,0,1,0]^T - 0.15 *[1,0,1,0,0,1]^T =$ **[3.6, 3.5, 0.85,1,0.75,0]**

b. D2 relevant, D1 non-relevant

$q_m = [3,2,1,1,0,0]^T + 0.75 * [1,0,1,0,0,1]^T - 0.15 *[1,2,0,0,1,0]^T =$ **[3.6, 1.7, 1.75, 1, 0, 0.75]**

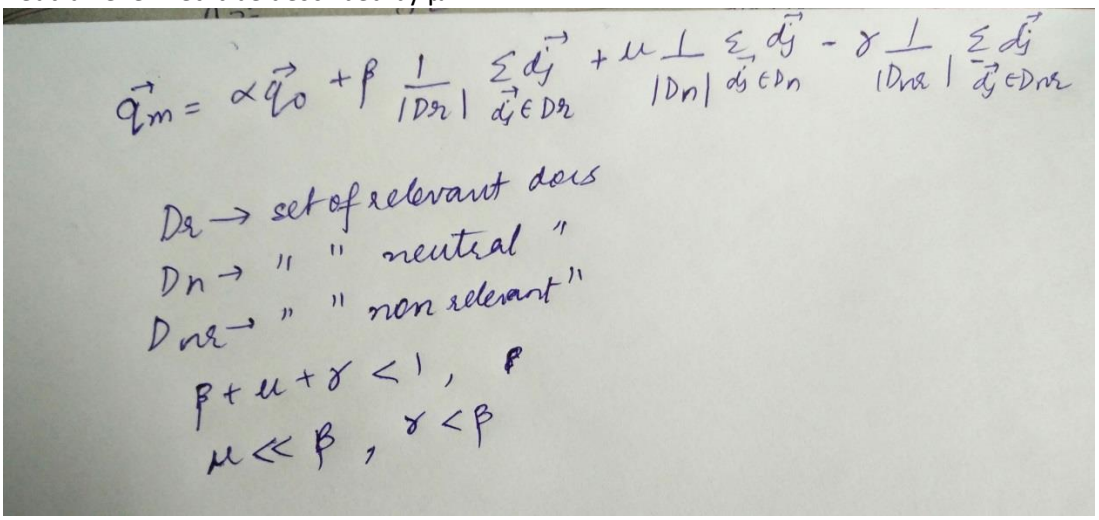
c. both relevant

$q_m = [3,2,1,1,0,0]^T + 0.75/2 *\{ [1,2,0,0,1,0]^T + [1,0,1,0,0,1]^T\} =$ **[3.75, 2.75, 1.375, 1, 0.375, 0.375]**

d. both non-relevant

$q_m = [3,2,1,1,0,0]^T - 0.15/2* \{[1,2,0,0,1,0]^T + [1,0,1,0,0,1]^T\} =$ **[2.85, 1.85, 0.925, 1, 0, 0]**

Traditionally, the feedback in relevance feedback is given as “Relevant” and “non-Relevant” , i.e., the feedback is binary. To extend the system to account for three levels of feedback “Relevant”, “Neutral” & “Non-Relevant”, we need to introduce an extra constant that takes care of neutral level. Let it be described by μ.

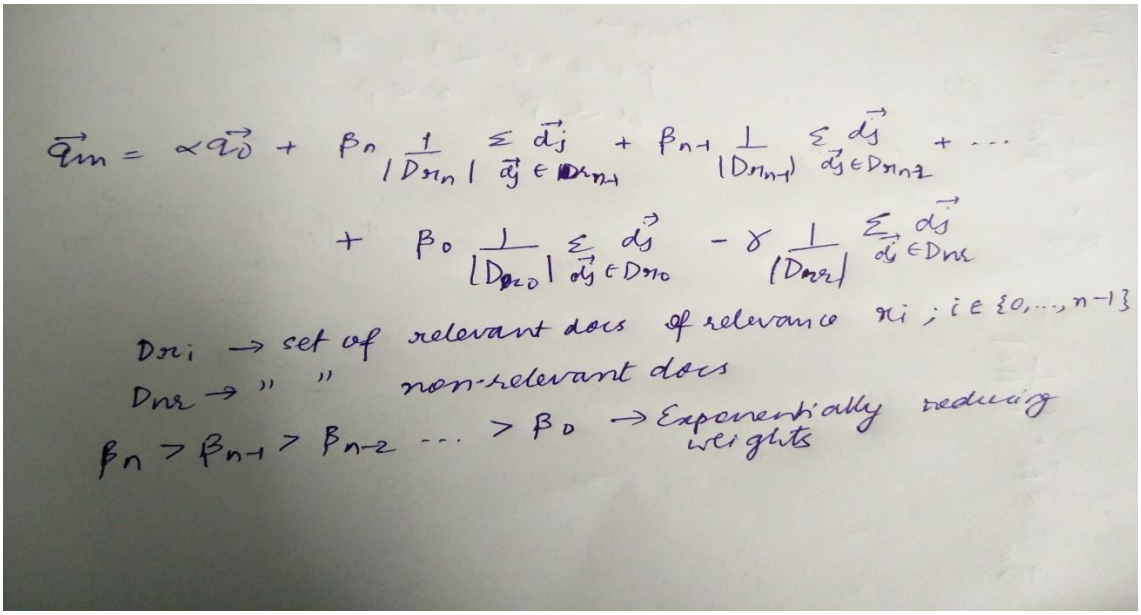


Since the neutral documents are neither relevant nor irrelevant, we need to add a separate weight. Irrelevant documents have lesser weight in the original Rocchio algorithm. This is because the system should have more relevant documents and that should get more weight. For neutral, its relevance is less than that of a relevant document but more than that of an irrelevant document. Thus, its weight should be lesser than β and it should be added to the equation (unlike irrelevant, which is subtracted). It weight should be closer to 0 as they should not heavily impact the query vector.

To extend the system to account for N levels of feedback:

We should get a relevance value and that should help us rank the results so that good results get higher weight and worse ones get lesser weight in the modified equation. We can modify the equation as follows:

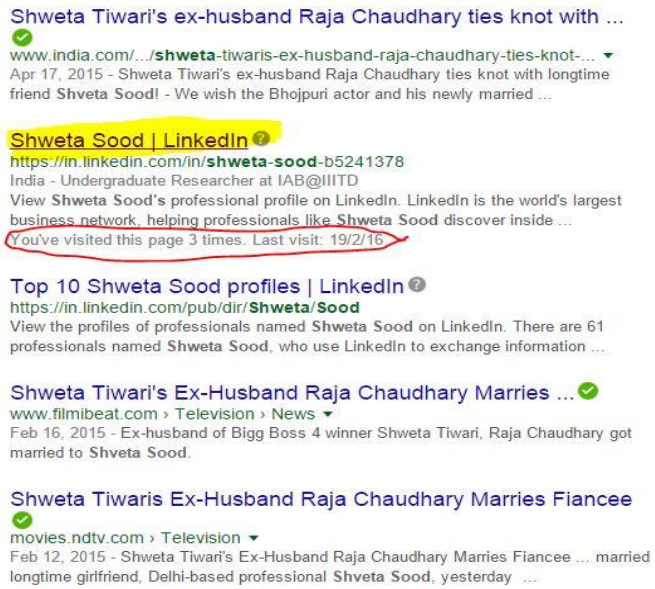
If the relevance value is in the range $[0,r_{n-1}]$ for n levels, we give them weights β_{n-1} (highest) to β_0, such that these weights are exponentially decreasing and the sum of betas and gamma is still less than one (as required).



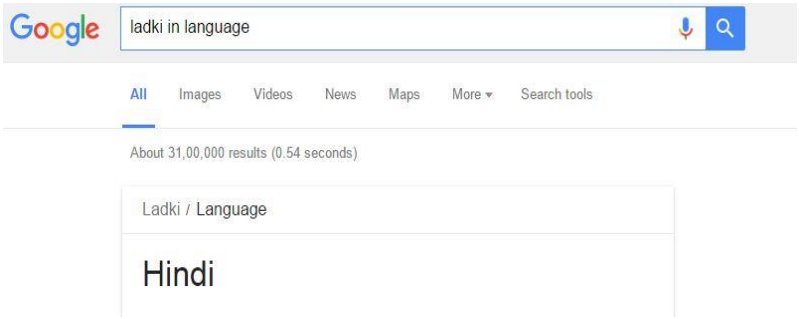
This will ensure that all relevant documents have a weight but lesser ones have a lesser weight. Also, non-relevant documents are still subtracted as before.

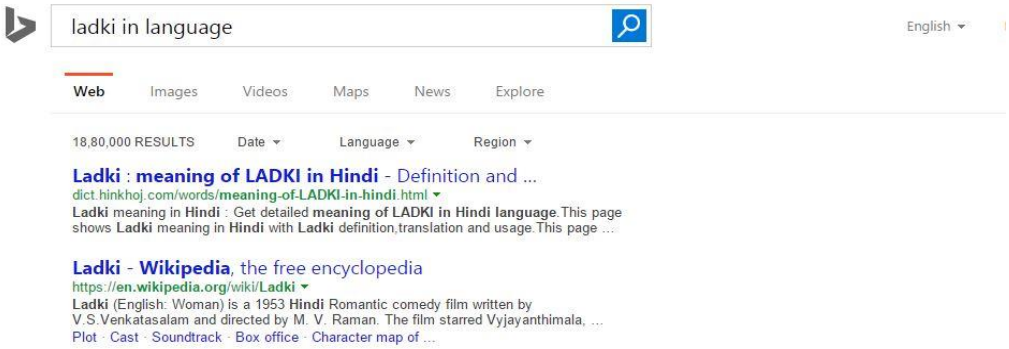
Q5 Surveyed Google and Bing to understand any features or improvements required. Three features that can be improved or added to the platform:

1. Personalized search in Google vs Bing- When I searched for my name “Shweta Sood” on the two search engines there was a difference in results. Google showed “Shveta Sood” on top. First 2-3 results were of spelling “Shveta” and not “Shweta” pertaining to a celebrity. However, Bing showed my linkedin profile of “Shweta Sood” on top. This means that Google is not very reliable for searching personal information as it ranks the results on popularity and not on relevance for user. However, Bing does it on relevance.
   Also, the personalized search for a user is supposed to rank the pages that the user visits often or is more likely to be interested in (based on user’s previous history searches) as higher. But Google ranked it much lower for me. I had visited my LinkedIn profile thrice before, yet it showed my page much below other results. This means it is not ranking on my preferences as shown in the figure.



2. Language Identification : I searched for a query “ladki language” to know the language the word “ladki” belongs to. Google automatically showed the results even before I pressed search. However, Bing didn’t show up the language. It had search results that were links to pages.





However, even Google didn’t show appropriate translated results when I queried “Ladki in English” though it was able to detect Hindi word for the query “Girl in Hindi” and showed translated word as “Ladki”. So, both search engines need improvement in Language identification. Google, performs fine for English but performs poorly for other languages.

3. Summarizing an article– Search engines display the relevant search results for queries. But very often, we don’t have the time to scroll through all the articles and rather require just a summary of the event/ query. Eg – Nowadays, Google does show meaning of words on typing for them, or description of Institutes in a small paragraph. Likewise, we can summarize an event or story in a small paragraph saving us the effort to read the entire article.

My outline of the proposed feature “Summarizing an article” is as follows :

- Input is a long document with many paragraphs of text. The text is preprocessed to remove non-alphabetic characters.
- For each sentence in the text, we can find its similarity score with other sentences in the text by finding the number of common tokens. We store this intersection value for every pair of sentences.
- Essentially, I converted my text to a fully connected weighted graph, where each sentence is a node in the graph and the edges indicate how similar are two sentences.
- Next, the score for each sentence is stored in a key value dictionary where, the sentence itself is the key and value is the score, obtained by summing up all the intersections will all other sentences in the text. So, we have summed up all edges connected to the node in the graph.
- We take the input document, split it into paragraphs. Next, to build the summary, for each paragraph in our pre-processed text, we choose the best sentence from each paragraph

according to the score of that sentence in the dictionary. Thus, every paragraph of the text gives the best sentence. This helps in building the summary of the paragraph.

Paragraph being the logical atomic unit of the text, it can be used to find the best sentence. Also, sentence having best intersection with other sentences should hold information from each of them and that becomes the key sentence in the paragraph.

The code has been provided as well.