Task 1: Evaluation Metrics and α-NDCG

1.1 Name two aspects that are not considered when using the standard information retrieval evaluation metrics precision and recall to measure the relevance of a ranked document by a search engine.

1.2 Given the equations for α-NDCG@k and for the gain (for explanations see appendix), answer the following questions for each letter in "α-NDCG":

- What aspect does the letter stand for?
- What is that aspect's role in α-NDCG?
- In which part of the equations is that aspect covered?

$$\alpha - NDCG(q,k) = Z_{kq} \sum_{m=1}^{k} \frac{G[m]}{log_2(1+m)}, \quad G[m] = \sum_{i=1}^{|S|} J(d_m, n_i)(1-\alpha)^{r_{i,m-1}}$$

Letter	Name	Description	Part of the Equation
G			
α			
D			
C			
N			

Task 2: Computing the Gain Vector for α-NDCG

Given is the following ranking of documents and the information nuggets they contain.

Document			Information Nuggets						Gain
ID	**Rank**	**Title**	n_1	n_2	n_3	n_4	n_5	n_6	
a	1	Carnival Re-Enters Norway Bidding		X		X			
b	2	NORWEGIAN CRUISE LINE SAYS OUTLOOK IS GOOD		X					
c	3	Carnival, Star Increase NCL Stake		X					
d	4	Carnival, Star Solidify Control							
e	5	HOUSTON CRUISE INDUSTRY GETS BOOST WITH...	X					X	
f	6	TRAVELERS WIN IN CRUISE TUG-OF-WAR	X						
g	7	ARMCHAIR QUARTERBACKS NEED... THIS CRUISE			X				
h	8	EUROPE, CHRISTMAS ON SALE	X						
i	9	TRAVEL DEALS AND DISCOUNTS							
j	10	HAVE IT YOUR WAY ON THIS SHIP							

2.1 Compute the gain for each document.

Task 3: Computing α-NDCG

Compute the α-NDCG for the top-10 results ($k = 10$) for a single query q that returns the document ranking from Task 2.

3.1 Create the perfect ranking for that document ranking and compute the corresponding gains and its DCG for $\alpha = 0.5$.

Document	Information Nuggets						Gain
ID	n_1	n_2	n_3	n_4	n_5	n_6	

3.2 Compute the α-NDCG of the ranking in Task 2.

4. Diversification algorithm

Given is the following list L[l] of top-k queries ranked by relevance:

Rank	ID	P(Q|K)	Query Interpretation
1	Q_1	0.95	{“consideration”: movie.title, “christopher guest“:director.name}
2	Q_2	0.87	{“christopher guest“:director.name}
3	Q_3	0.78	{“guest”: movie.title, “consideration”: movie.title}
4	Q_4	0.23	{“consideration”: movie.title}

Use the diversification algorithm to return the list R[r] of the relevant and diverse query interpretations.

4.1 Use the Jaccard similarity to compute the similarity between each pair of query interpretations.

	Q_1	**Q_2**	**Q_3**	**Q_4**
Q_1				
Q_2				
Q_3				
Q_4				

4.2 Given these queries, interpretation probabilities and similarity values, perform the diversification algorithm to generate a diversified ranking for $r = 3$ and $a = 0.5$.

Appendix

α-NDCG@k for a single query q

$$\alpha - NDCG(q,k) = Z_{kq} \sum_{m=1}^{k} \frac{G[m]}{log_2(1+m)}$$

k – the number of results in the ranking that are considered

Z_{kq} – a normalization factor such that the $\alpha - NDCG$ of the perfect ranking for q at *k* is 1.

$G[m] = \sum_{i=1}^{|S|} J(d_m, n_i)(1-\alpha)^{r_{i,m-1}}$ – gain of a result

$J(d_m, n_i)$ – an assessor judged that document d_m contains nugget n_i.

α – a factor that balances relevance and novelty

$r_{i,m-1}$ – the number of documents ranked up to position $m-1$ that have been judged to contain nugget n_i.

$S = \{n_1, ..., n_s\}$ – the set of information nuggets

Jaccard Similarity between two queries Q_1, Q_2 and their interpretations (I_1, I_2)

$$Sim(Q_1, Q_2) = \frac{I_1 \cap I_2}{I_1 \cup I_2}$$

Query Score combining Relevance and Similarity

$$Score(Q) = \lambda \cdot P(Q|K) - (1-\lambda) \cdot \Sigma_{q \in QI} \frac{Sim(Q,q)}{|QI|}$$

QI – a set of previously selected query interpretations in the ranked list

λ – a factor to balance relevance and novelty in the diversification procedure

Diversification Algorithm

Input: list $L[i]$ of top-k query interpretations ranked by relevance
Output: list $R[r]$ of the relevant and diverse query interpretations

```
Proc Select Diverse Query Interpretations
copy the first element of L in R;
while (less than r elements selected) {
      //select the best candidate for R[i]
      initialize best_score=0;
      while (more candidates for R[i] in L) {
            if (Score(L[j]) > best_score) set new best_score;
      }
      add the candidate from L with best_score to R
}
End Proc;
```