Task 1: Vector Space Retrieval (theoretical)

a) *Consider first a query with two terms and define a similarity threshold α. For both measures, identify the sub-space of documents that have a similarity score beyond α. Describe the space in geometrical terms.*

The inner vector product divides the space with a hyper-plane into two areas. In the two-dimensional space, the line $\boldsymbol{q}^\top \boldsymbol{x} = \alpha$ separates the areas. The relevant documents lie beyond the line looking from the null-point. The query vector $\boldsymbol{q}$ is the normal vector of the line (hyper-plane if more than two query terms are given). The cosine measure, partitions the hyper-space with a hyper-cone along the query vector $\boldsymbol{q}$ and the angle $\mathrm{acos}(\alpha)$. The space inside the hyper-cone holds the relevant documents.

b) Based on the geometrical semantics from a), identify the documents that are preferred by the measures. Construct an example document that "wins" the search (has highest scores). Generalize to queries with more than two terms.

*The inner vector product favors documents that contain the search terms **most frequently** (not necessarily all terms). In practice, smaller documents that have all the query terms may not appear at the top of the ranking, while long documents with only parts of the query term (but more frequent) are at the top. This similarity function is not robust against spamming and attackers may easily gain top rankings for interesting key words.*

*The cosine measure is not sensitive to the length of documents due to the usage of angles to query vectors. On the other side, it prefers documents that have **exactly the same ratio of term occurrences** as the query. If the query is "cat dog", documents that have an equal number of "cat" and "dog" occurrences are at the top (identical direction as the query vector). This does not change with the usage of an idf-weighting but can reduce the impact if frequent and infrequent terms are used in the same query. Example: for the query "car jaguar" and not using an idf-weighting, we would obtain documents that have the same number of occurrences for "car" and "jaguar" . With and idf-weighting, however, "jaguar" being the less frequent term obtains a much higher weighting and hence dominates the direction. Documents that contain only "jaguar" (regardless how often) are already pointing to the right direction, while documents with only "car" are pointing in the wrong direction. Spamming is not an option for an attacker, but adding lots of infrequent terms may still work to appear at the top of rankings.*

Task 1: Vector Space Retrieval (theoretical)

c) In web search, queries are often very short. What happens if you only select one query term? Are the measures working in this extreme case?

There are good reasons why web search engines do not use a pure vector space retrieval model. Web search often consists of only one or two query terms, very seldom more than four terms. The inner vector product, as we discussed previously, is not robust against key word spamming and would return long documents with frequent query term occurrences at the top of the ranking. Even without spamming, ordering millions of potential hits with the inner vector product will have the longest documents at the top, but not necessarily the most relevant ones. With only one query term, the document with highest term occurrence wins the search.

The cosine measure is not much better suited. It works quite well with longer queries that use some infrequent terms. But with short queries it looses a lot of its power. We already discussed the issue of query term ratios. With the extreme case of only one query term, the deficit becomes obvious. In a one-dimensional space, there is only one direction (no negative values). All documents containing the query term are equally relevant. How do we find "Microsoft" with this? Every web page with the term "Microsoft" is an equal candidate to be in the top ranking. Clearly not what a user is expecting.

d) Similar to a), describe the sub space of documents that have at most a distance of β to the query Q. What documents rank highest with this distance measure? Does this work in our scenario (finding similar pages) and why?

Using a Euclidean distance measure, the query defines a hyper-sphere (circle in two dimension) around the query point and with a radius of β. The relevant documents are within the sphere, the non-relevant outside the sphere. The measure prefers documents that have all the query terms and equally frequent as in the query document. Obviously, the method only finds identical or slightly adjusted copies, but is not able to find query texts that are embedded in longer documents. The method is suitable to find copies of the same pages like Javadoc instances, and to dismiss most of them to not flood the result list with a lot of (almost) identical documents.

Task 2: Probabilistic Retrieval (theoretical)

a) Given the relevance assessments, compute the new c_j-values given the feedback and compute the ordering.

The table below shows the c_j-values. We have: $l = 12$ (number of relevant documents) and $k = 20$ (number of presented documents).

	l_j	k_j	r_j	n_j	c_j
x_1	8	11	2/3	3/8	1.20
x_2	7	11	7/12	1/2	0.34

We can use the c_j values to order the binary representation as follows: $(1,1) > (1,0) > (0,1) > (0,0)$. Hence, the feedback was not changing the order and we obtain the same ranking. This is not uncommon for such small queries (with only very few terms). In subtask c), we will do better than this.

b) The BIR model makes three assumptions. We now test whether these assumptions hold true.

We first compute the probability $P(R|\boldsymbol{x})$ using the BIR assumptions:

$$sim(Q, D_i) = \frac{P(R|\boldsymbol{x})}{1 - P(R|\boldsymbol{x})} = \frac{P(R)}{P(NR)} \cdot \prod_{\forall j: x_j = 1, q_j = 1} \frac{r_j}{n_j} \cdot \prod_{\forall j: x_j = 0, q_j = 1} \frac{1 - r_j}{1 - n_j}$$

$$P(R|\boldsymbol{x}) = \frac{\frac{P(R)}{P(NR)} \cdot \prod_{\forall j: x_j = 1, q_j = 1} \frac{r_j}{n_j} \cdot \prod_{\forall j: x_j = 0, q_j = 1} \frac{1 - r_j}{1 - n_j}}{1 + \frac{P(R)}{P(NR)} \cdot \prod_{\forall j: x_j = 1, q_j = 1} \frac{r_j}{n_j} \cdot \prod_{\forall j: x_j = 0, q_j = 1} \frac{1 - r_j}{1 - n_j}}$$

Now for the counting part: we have $P(R) = 12/20$ (12 out of 20 documents are relevant) and $P(NR) = 1 - P(R) = 8/20$. The same for $P(R|\boldsymbol{x})$: for example, we have $P(R|(0,0)) = 1/3$ (1 out of 3 documents with representation (0,0) is relevant). In summary we obtain:

$P(R\|\boldsymbol{x})$	(0,0)	(0,1)	(1,0)	(1,1)
counted	0.33	0.50	0.67	0.80
computed	0.40	0.48	0.69	0.76

The results differ because of the independence assumption of terms. In practice, this does not hold true and hence the probabilities differ.

Task 2: Probabilistic Retrieval (theoretical)

c) *Consider the documents c1-c5, m1-m4 and the query "human computer interaction". Conduct two iterations with the BIR model.*

The following table shows the c_j-values with the BIR model. The first step uses the initial estimates for r_j and n_j, the second steps adjusts them with the feedback and $l = 5$ and $k = 9$.

	first step				second step				
term t_j	$df(t_j)$	r_j	n_j	c_j	l_j	k_j	r_j	n_j	c_j
human	2	0.5	0.22	1.25	2	2	0.42	0.1	1.86
computer	2	0.5	0.22	1.25	2	2	0.42	0.1	1.86
interaction	0	0.5	0	0	0	0	0.08	0.1	-0.2

We can use the c_j to order the document as per table below (columns 1 & 2). However, we already see the same problem as in subtask a) that the feedback is not really improving the ordering.

	1st step	2nd step	with new terms
D_i	$sim(Q, D_i)$	$sim(Q, D_i)$	$sim(Q, D_i)$
c1	2.50	3.72	5.58
c2	1.25	1.86	10.65
c3	0	0	6.93
c4	1.25	1.86	4.14
c5	0	0	6.26
m1	0	0	-3.25
m2	0	0	-6.49
m3	0	0	-8.89
m4	0	0	-5.89

One way to improve the query with the feedback is to add additional query terms with high (absolute) c_j-values. Lets try and add every term (right column in the table above). Now, we get all relevant documents and even c3 and c5 are found and make it to the top 3.

Term t_j	l_j	k_j	r_j	n_j	c_j
human	2	2	0.42	0.1	1.86
interface	2	2	0.42	0.1	1.86
computer	2	2	0.42	0.1	1.86
user	3	3	0.58	0.1	2.53
system	3	3	0.58	0.1	2.53
response	2	2	0.42	0.1	1.86
time	2	2	0.42	0.1	1.86
eps	2	2	0.42	0.1	1.86
survey	1	2	0.25	0.3	-0.25
trees	0	3	0.08	0.7	-3.25
graph	0	3	0.08	0.7	-3.25
minors	0	2	0.08	0.5	-2.40