1 (a)

The following documents have been processed by an IR system where stemming is not applied:

DocID	Text
Doc1	france is world champion 1998 france won
Doc2	croatia and france played each other in the semifinal
Doc3	croatia was in the semifinal 1998
Doc4	croatia won the other semifinal in russia 2018

(i) Assume that the following terms are stopwords: and, in, is, the, was. Construct an inverted file for these documents, showing clearly the dictionary and posting list components. Your inverted file needs to store sufficient information for computing a simple tf*idf term weight, where $w_{ij} = tf_{ij}*\log_2(N/df_i)$

[5]

(ii) Compute the term weights of the two terms "champion" and "1998" in Doc1. Show your working.

[2]

(iii) Assuming the use of a best match ranking algorithm, rank all documents using their relevance scores for the following query:

1998 croatia

Show your working. Note that $\log_2(0.75)= -0.4150$ and $\log_2(1.3333)= 0.4150$.

[3]

(b)

(i) In Web search, explain why the use of raw term frequency (TF) counts in scoring documents can hurt the effectiveness of the search engine.

[2]

(ii) Suggest a solution to alleviate the problem, and show through examples how it might work. Explain through examples how modern term weighting models in IR control the raw term frequency counts.

[3]

(c) Assume that you have decided to modify the approach you use to rank the documents of your collection. You have developed a new Web ranking approach that makes use of recent advances in neural networks. All other components of the system remain the same. Explain in detail the steps you need to undertake to determine whether your new Web ranking approach produces a better retrieval performance than the original ranking approach.

[5]

2.

(a) Consider a corpus of documents C written in English, where the frequency distribution of words approximately follows Zipf's law $r * p(w_r|C) = 0.1$, where $r = 1,2, \ldots, n$ is the rank of a word by decreasing order of frequency. w_r is the word at rank r, and $p(w_r|C)$ is the probability of occurrence of word w_r in the corpus C.

Compute the probability of occurrence of the most frequent word in C. Compute the probability of occurrence of the 2nd most frequent word in C. *Justify your answers.*

[4]

(b) Consider the query "michael jackson music" and the following term frequencies for the three documents D1, D2 and D3, where the search engine is using raw term frequency (TF) but no IDF:

	Indiana	Jackson	life	michael	music	pop	really
D1	0	4	1	3	0	6	1
D2	4	0	3	4	1	0	2
D3	0	4	0	5	4	4	0

Assume that the system has returned the following ranking: D2, D3, D1. The user judges D3 to be relevant and both D1 and D2 to be non-relevant.

(i) Show the original query vector, clearly stating the dimensions of the vector.

[2]

(ii) Use Rocchio's relevance feedback algorithm (with $\alpha=\beta=\gamma=1$) to provide a revised query vector for "michael jackson music". Terms in the revised query that have negative weights can be dropped, i.e. their weights can be changed back to 0. *Show all your calculations.*

[4]

(c) Suppose we have a corpus of documents with a dictionary of 6 words w_1, ..., w_6. Consider the table below, which provides the estimated language model p(w|C) using the entire corpus of documents C (second column) as well as the word counts for doc_1 (third column) and doc_2 (fourth column), where ct(w, doc_i) is the count of word w (i.e. its *term frequency*) in document doc_i. Let the query q be the following:

$$q = w_1\ w_2$$

Word	p(w|C)	ct(w, doc_1)	ct(w, doc_2)
w_1	0.8	2	7
w_2	0.1	3	1
w_3	0.025	2	1
w_4	0.025	2	1
w_5	0.025	1	0
w_6	0.025	0	0
SUM	1.0	10	10
Word	p(w|C)	ct(w, doc_1)	ct(w, doc_2)

(i) Assume that we do not apply any smoothing technique to the language model for doc_1 and doc_2. Calculate the query likelihood for both doc_1 and doc_2, i.e. p(q|doc_1) and p(q|doc_2) (Do not compute the log-likelihood; i.e. do not apply any log scaling). *Show your calculations*. Provide the resulting ranking of documents and state the document that would be ranked the highest.

[3]

(ii) Suppose we now smooth the language model for doc_1 and doc_2 using Jelinek-Mercer Smoothing with $\lambda = 0.1$. Recalculate the likelihood of the query for both doc_1 and doc_2, i.e., p(q|doc_1) and p(q|doc_2) (Do not compute the log-likelihood; i.e. do not apply any log scaling). *Show your calculations*. Provide the resulting ranking of documents and state the document that would be ranked the highest.

[4]

(iii) Explain which document you think should be reasonably ranked higher (doc_1 or doc_2) and why?

[3]

3.

(a) How would the IDF score of a word *w* change (i.e., increase, decrease or stay the same) in each of the following cases: (1) adding the word *w* to a document; (2) making each document twice as long as its original length by concatenating the document with itself; (3) Adding some documents to the collection. *You must suitably justify your answers.*

[4]

(b) Explain in detail why positive feedback is likely to be more useful than negative feedback to an information retrieval system. Illustrate your answer using an example from a **suitable** search scenario.

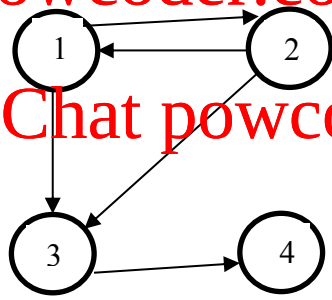
[4]

(c) Neural retrieval models often use a re-ranking strategy over BM25 to reduce computational overhead.

Explain the key limitation of this strategy. Describe in sufficient details an approach that you might use to overcome this problem.

[5]

(d) Consider a query q, which returns all webpages shown in the hyperlink structure below.



(i) Write the adjacency matrix A for the above graph.

[1]

(ii) Using the iterative HITS algorithm, provide the hub and authority scores for all the webpages of the above graph after a complete *single* iteration of the algorithm. *Show your workings.*

[3]

(iii) Describe in sufficient details an **alternative** approach to compute the hub and authority scores for the above graph. You need to show all required steps to generate the scores, but you do not need to actually compute the final scores.

[3]