SOLUTION SKETCH, INF3800 V12

QUESTION 1: INVERTED INDEX

a) By document ID, by static quality [Section 7.1.4], by impact [Section 7.1.5]. The latter precludes concurrent traversal of the posting lists (thus making document-at-a-time scoring a non-option), but necessitates query-term-at-a-time scoring.
b) See [Figure 6.2] versus [Figure 6.3]. Impacts size of dictionary versus complexity of posting lists. Weighted zone scoring.
c) See [Figure 4.6].

QUESTION 2: HEAPS OF FUN

a) See [Section 5.1.1].
b) See [Section 5.1.2].
c) See [Section 5.3.2]. Usually only when space is at an extreme premium, otherwise VB is usually preferred since it compresses "well enough" and is a lot more efficient to decode.
d) See [Figure 3.5]. Could be extended to also take into account *m[i–2, j–2]* in the *min* expression.

QUESTION 3: LEARNING TO RANK

a) See [Section 15.4.1]. Could, e.g., sort by distance to plane.
b) See [Section 15.4.2]. Feature vector of differences, pairs of documents form training instances.
c) See [Section 8.4]. A description of what it does (i.e., explain N, D, C and G) should suffice, I don't expect students to get the exact formula right. Fooled by duplicates.

QUESTION 4: NAÏVE BAYES

a) See [Section 13.2]. Bag-of-words, independence. Wrong probability estimation doesn't necessarily imply bad classification (calibration versus discrimination).
b) See [Section 13.2]. To deal with sparseness, eliminate zeros.
c) See [Example 13.1].
d) See [Example 13.1].

QUESTION 5: APPLICATIONS

a) Let's define an index called *purchases*. The *purchases* index would have one document per customer, listing the items that the customer has purchased. E.g., the *purchases* index might contain data like this:

   document1 = {"*userId*":"12345", "*itemIds*":["4", "19", "189"]}
   document2 = {"*userId*":"67890", "*itemIds*":["12", "34"]}
   ...

I.e., user 12345 has purchased items 4, 19 and 189, and so on. The users and items are here for simplicity and without loss of generality represented using integers, but any value that can be used as a lookup key for further data is OK. (For example, we here assume that an item identifier of 19 is a sufficient lookup to know that item's name, price, description etc in some other system, or possibly in some other index in the same search engine.) There is one document per user, i.e., there are no two documents that share the same value for the *userId* field. The *purchases* index is configured so that all fields are searchable and we can compute a facet over the *itemIds* field. Thus we can do searches like "retrieve documents that have *userId*=234" and "retrieve documents that have 19 as a member of *itemIds*", and for each such query we in the result set can produce a histogram of the distribution of values from *itemIds*. E.g., for a query like "retrieve documents that have 19 as a member of *itemIds*" we would be able to get back a facet (let's call it *copurchasedItems* in the following) having a value like, e.g., {"19":5675, "797":2892, "69": 1347, "3465":1201, ...}. We simply ignore the first element of *copurchasedItems* to read out the answer: Hence, if item 19 is the reference item for which recommendations should be triggered, we'd recommend items 797, 69, 3465, etc (in that order).

b) Extend above with *catId* or similar, and use that.