Home exam for an algorithms engineer candidate in Taboola

General Guidelines

Please prepare your answers at home before your next interview. During the interview, you will be asked to present your ideas and we will discuss them together. There is no need to submit a written solution.

In the interview, we will discuss Question 1 and three out of the four questions Q2-Q5, at your choice.

While solving this exam, you may use any tool or media that you find useful (including books, Google, etc.).

For any questions, please contact Gil Chamiel on +972 505 415 102 or by email gil.c@taboola.com

Good luck!

Questions

Question 1 (compulsory)

Taboola tracks users who browse the internet in websites where it is integrated in (which is most of the internet). We also keep information about previous clicks on Taboola recommendations and previous recommendations which weren't clicked.

Given this user history, how would you use it in order to predict the probability of a specific recommendation to be clicked by a specific user in a specific context?

You may assume any data which, in your opinion, seems reasonable for Taboola to have about the user history and the context (for instance, the article which the user views now, its geographic location and so on).

Please select three out of the four questions below

Question 2

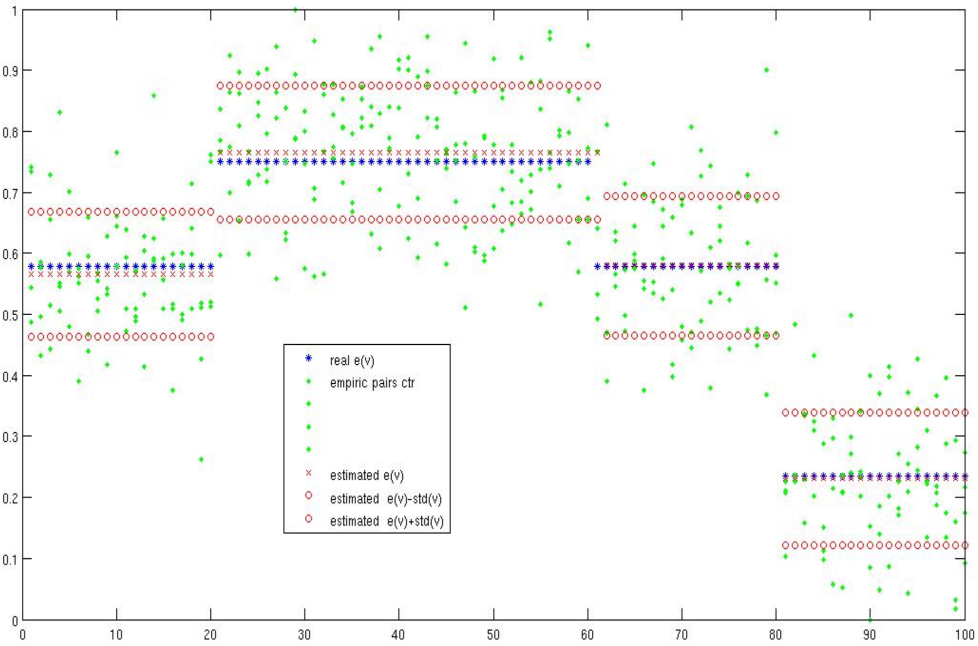
Given a corpus of items $X=\{x_1,x_2,\ldots,x_n\}$, a recommendation heuristic $H(x_s, x_t)$ is a function which returns a number between 0 and 100 for any ordinal pair of items (x_s, x_t) where x_s represents a source item (an item the user is watching) and x_t represents a target item (a candidate item for recommendation).

For a pair of items (x_s, x_t), the CTR (click through rate) is defined as follows:
$ctr(x_s,x_t) := P(\text{user will click on } x_t \text{ when it is recommended next to } x_s)$

We would like to estimate the CTR mean and standard deviation of a random pair for which the heuristic H returns a given score v. That is, we would like to estimate the following functions:

$E(v) = E(ctr(x_s,x_t) \mid H(x_s,x_t) = v)$, $Var(v) = Var(ctr(x_s,x_t) \mid H(x_s,x_t) = v)$

A. Suppose we collected and stored in our database, CTR information for some pairs with a specific score v (e.g. 70). Suggest estimations for E(v) and Var(v) for that v.
B. Let's assume that E(v) and Var(v) are piecewise constant functions and suppose we collected CTR information for various values of v. Suggest a method for estimating E(v) and Var(v) for any value of v. Your method should be able to deal with noisy or sparse data. That is, assume that for some values v, we don't have any CTR information on pairs with that score or that this information exists for a small number of pairs (and therefore, the CTR which stems from this sample is noisy and does not fit the true CTR value).



The graph above demonstrates the input and output of the problem. The X-axis is in terms of values of the heuristic H and the Y-axis is in units of probability.

The algorithm's input (green dots) consists of empirical CTR of pairs for various values of v. The output (in red), represents the estimated values for E(v) and the estimated standard deviation around it. The real E(v) function appears in blue (and is, clearly, unknown).

Question 3

Suppose we have a corpus of documents with integer identifiers and an inverted index, which maps words to skip lists of document identifiers. Our skip list implements the following interface:

1. getDocId() – returns the identifier of the document in the current position of the skip list cursor.
2. skipTo(docId) – moves the cursor to the first position where the document identifier is equal or greater than the docId parameter.
3. skipToNext() – move the skip list cursor to the next position. A possible implementation of this function is by calling skipTo(getDocId() + 1).

A. Describe an algorithm which, given a query of two words, returns all documents which contain the two words.

B. Describe an algorithm which, given a query of N words, returns all documents which contain all N words.
C. Describe an algorithm which, given a query of N words, returns all documents which contain at least K words.

An emphasis will be given to the time and space complexity of the solutions.

Question 4

Design a generic read-only local cache with the following property: if the cache is asked for a key that it doesn't contain, it should fetch the data using an externally provided function that reads the data from another source (database or similar).

What features do you think such a cache should offer? How, in general lines, would you implement it?

Question 5

Recall the user history data discussed in Question 1.
Explain how would you design a sub-system which collects user data in a manner which will enable having it available for model construction (offline) and for prediction (online).
You may assume that all metadata attached to articles and recommended items (such as classification, article text etc.) is available both online and offline. However, fetching this data requires a db call.
Please explain which technologies you would choose to use and how would you implement your solution.