This assignment will make use of the previously computed item-item lifts, in order to recommend items (movies) to users in a streaming fashion. You will make use of HBase and Storm. In parallel, a global recommendation quality score $Q$ will be tracked.

Technicalities:
1. Continue using the Hortonworks Sandbox on your machine.
2. Read the following references about using Storm and HBase:
   b. https://github.com/ptgoetz/storm-hbase
3. Continue using MovieLens 1M dataset.

As a preliminary step, load the output computed in Homework 1 onto HBase, where the key of each record is {movie-id x, [pos|neg]}, and the value is the array of top-k PosLift [NegLift] values computed given movie x. The value of each array entry is a pair <y, PosLift value> (or NegLift value).

Then, using Storm, process in streaming manner the records indicating movies rated by users in the MovieLens ratings document stored in HDFS. Your mission is to generate recommendations for the users and then to evaluate your recommendations based on the users rating. For this you are asked to maintain

1. A global score $Q$ initialized to 0 (see below how it is updated)
2. For every user $u$, maintain in HBase a sliding window of the last 5 movies the user has rated. Keep track, for each movie in the window, whether it was rated positively (3 starts and higher) or negatively (2.5 starts and below) by the user. Note that this requires processing each user’s activity stream serially, although different users should be processed in parallel.

Upon reading a record of $<\text{user u, movie x, rating}>$:
1. If the user’s window is not yet full, insert the current movie and the sign of the rating into the window.
2. For a user $u$ whose window is full (i.e. has 5 previous ratings in their HBase window) perform the following steps:
   a. Generate a recommendation list: compute some aggregation $A^1$ of the PosLift and NegLift arrays of the five movies in the user’s window (use the PosLift arrays for movies the user liked, and NegLift arrays for movies

---

$^1$ An example of a very simple and naïve aggregation function would be a “max score” scheme, where the score of each recommended movie would be its max PosLift value (or minimal NegLift value) in the five arrays of the window.
the user disliked). The aggregation may not “peek” at the movie x or its rating. The output of this step is a ranked list of recommendations for user u, given the 5 movies in the window. The ranked list should cover all movies mentioned in the PosLift and NegLift arrays of the five movies in the window.

b. **Evaluate the recommendations:** search for x in the ranked list you computed above, and look at the rating given by the user to x. If x is not in the list, continue straight to step c. Otherwise, assume you found x in rank r (r=1 is the top recommendation). If u liked x, add 1/r to Q. If u disliked x, subtract 1/r from Q.

c. **Maintain the sliding window:** remove the oldest movie from u’s window, and insert the current movie x and the sign of its rating into the window.

**The required output:** a plot of the dynamics of Q/#recommendations over time. Specifically, you should aggregate ΔQ in batches of 1000 recommendations each, so that the plot consists of points Q_t/#rec_t where Q_t = Q_{t-1} + ΔQ, and #rec_t=1000t.

You are required to come up with **two different aggregation functions**, and to **document your rationale** for each of them.

In addition to submitting working code and the two outputs, please submit clear and comprehensive external documentation of your solution. Emphasis will be put on the elegance, efficiency and scalability of your solution. Like any wet assignment, there are multiple ways to solve the given problem, often with various trade-offs. The documentation should call out and justify the design choices you made, and any assumptions they are based on.

**Good Luck!**