Explanation of Q1:

Step1:

Extracted required fields from 5.5gb corpus and dump it into small file "compressedTweets.txt" of 137mb.

Step2:

Now, again fetch data from small file and dump tweetId's into adjacency list of their parent TweetId's if available, else make it as one of the head tweetId's for start of conversation. The adjacency list will store parent child relationship between tweetId's and a Hash Set will store the head tweetId's for the start of conversation.

How to run the code:-

Step1:

For creating small file from large file, compile and run class "TrimTweetData.java".

Step2:

For creating tree from small file, compile and run class "populateTweetInfo.java".

Description of Q2:

1. For plotting graph, we need to do Depth Order Traversal or Breadth Order Traversal to take count of various parameters required for constructing graph.

2. Results obtained are:

1. Distribution of #participants. Plot freq v/s #participants.

b) Distribution of conversation length. Plot freq vs conversation length. Conversation

length is the number of nodes (tweets) in a conversation tree.

c) Distribution of conversation duration. Plot freq vs conversation duration. Duration is the difference in minutes between the first and the last tweet in the conversation tree.

Approach for Q3:

Step1:

Fetch tweets information into our memory and construct the tree.

Step2:

Initialize HashSet for every tweet which will contain all the text tokens of pre-processed Tweet's.

Step3:

Traverse tree structure (DFS/BFS), calculate Jaccard's Similarity between tweets Set and query Set using set operations.

Step4:

During traversal, keep track of the tweets with at least one reply having maximum Jaccard's score.

Step5:

Randomly choose a reply tweet for the tweet with maximum similarity.

How to run the code:-

Step1:

Compile and run the class "ChatBot.java".

(Will take few seconds to load data and construct tree, and then it waits for the next user query in a loop)

Answer to Q4:

1. What is a good way of improving the above social bot system for continuous multi‐turn dialogues?

1) Stemming and lemmatization can improve our search accuracy by comparing root words only.

2) Using word similarity measures (like Word Net) in terms of meaning can also contribute to better results.

3) #HashTags should be given more preference while calculating similarity score.

1. Imagine that you have a new search engine which shows Twitter conversations as results rather than webpages as results. What factors should be taken into consideration when ranking the results?
2. Exploring twitter #HashTags present in search term.
3. Giving Proper weightage to #HashTags while tweet comparison.
4. No. of followers of that user.
5. No. of Retweets of that tweet.
6. No. of replies for that tweet.
7. No. of users participated in that conversation.
8. Is that user a well-known personality.
9. Imagine that you have a new search engine which shows Twitter conversations as results rather than webpages as results. How would you generate snippets to be shown on search results page for such a search engine?