Answer 1 - Given we have only the data presented, the best we could do to compare the mean for both the cases. 0.2 for A and 0.21 for B. So I would propose that B does slightly better than A.
However only able to compare mean may not give any clue about statistical significance of the results and would need more information to find out the same.

Answer 2 - I would divide the whole process into following phases -

a. Collecting the stream - Depending on the infrastructure we are using for collecting tweet, we may need an API to collect the tweets from twitter. This should get the tweets and format them into the JSON format required.
b. Transformations - We would need to convert the tweets in JSON format to the feature vectors which can be fed into some machine learning algorithm. I can think of two such transformations. Either we can use n-grams to capture the short sequence of words which can work as a feature vector with some context. Or we can try to use deep learning auto-encoders to learn the compact representations for using further for similarity analysis.
c. Last stage would be to use some clustering algorithm (k-means or hierarchical agglomerative clustering) on feature vectors to create the clusters of tweets. Now based on the tweets clusters, we can assign different clusters to the users (with one to one correspondence) . One scheme would be to assign a user to cluster where most of its tweets lie.

Answer 3 - The method used to avoid overfitting may be dependent on model we are using (e.g. for neural networks, we can use regularization technique like stochastic dropout for avoiding overfitting which may not be applicable to other models).
Some generic techniques would be to reduce the capacity of the model (e.g. use smaller k in knn, use polynomials of smaller order in case of linear regression, use lesser layers in case of neural network) and using regularization. One thing we should take care is to use cross validation scheme (preferably stratified cross validation) to avoid overfitting due to a specific split of test and training data.

Answer 4- As the system has to learn the behavior for each user, we cannot use any “rule based system” here. it would make sense to use some supervised or reinforcement learning strategy. However, it would degrade the experience of user if he/she has to sit for a long time to provide data for training the model before starting to use the system. Hence I rule out offline supervised machine learning strategies. It would be better to start the user with a simple context menu in the start. Then we can use the online learning algorithm to improve the experience. One possible setup would be to predict the user action based on its experience and if the user actually uses the menu that was predicted, it gets high score and if user looks for something else then we decrease the score for this prediction.
This setup sounds like a reinforcement learning system would be a great fit here. The agent would sense the states i.e. “user menu selection” and immediate rewards “whether the

predicted menu item was used by user". We can use simple Q learning algorithm for the system described.

Answer 5 - Regularization helps in the case of overfitting and help model to generalize better. If we see that our training error is consistently low however the validation error is high, it points to the case of overfitting and we should use regularization technique.

However, adding regularization in case of underfitting model might affect adversely.

Answer 6 - We need to design a recommendation engine which learns based on the user history and preference and then predicts the likelihood of the objects the user would like to buy. One way to do the same would be to find similarities between the users. K nearest neighbour can be used to find the "k similar users" to the user. Then based on preference of the chosen k users, we can recommend the products. One simple criteria would be to choose the object which has been used most frequently by the chosen k users.
To evaluate the performance of the our model, we can divide the users into a training and testing set and then use a cost function to evaluate the model (e.g. root mean square error). For example, if we recommend a test user to use a product and that product was not bought (based on history of the test user), we will count that as score 0 and if it was selected, we can use score 1. Then calculating rmse over all test users would give an estimate of performance of our model.

Answer 7 - I have a broad range of experience in different projects in semiconductor industry working majorly on performance enhancement for GPGPU and DSPs and I am highly enthusiastic about machine learning. One role that excites me is to improve the performance of deep learning algorithms on different hardware and software platforms. Nvidia is one of the companies that is working on the similar projects quite extensively. Another profile that I am hugely interested would be a research profile to explore the possibility of using machine learning techniques to make programming easier and more efficient.