Recommender Systems

SUGGEST A RECOMMENDER SYSTEM THAT YOU BELIEVE WE COULD USE AT FETCH REWARDS

A content based recommender system can be deployed for new product and user:

1. Bag of words and TF-IDF for product:
   A common way to classify a new product is to use its attributes, and the attributes of the product can be extracted from the product description. For example, the description of a vacuum machine might contain keywords such as cleaning, house, electronic equipment, etc. Then we can classify products based on these keywords. This technique is usually related to the use of bag of words or term frequency-reverse document frequency (TF-IDF). As a result, when a user purchases or browses a cleaning related product, we can recommend a vacuum machine to him / her.

2. Machine learning based on attributes of user:
   We can ask the user to conduct a simple survey and extract attributes from the survey, and then use those attributes to predict which products are interesting for him / her. For example, Netflix asks new subscribers to choose some movies they are interested in and then recommends other movies related to those selected movies.

I suggest that both of the above methods should be deployed because content based system do not have cold start and are helpful for new user and product. We should pay more attention to the user part, because the growing rate of new users should be higher than the growing rate of new products. In addition, I build a simple convolutional neural network model called CNNBased.java to predict whether a user will buy a product based on his/her attributes.

A memory based recommender system can be deployed for old product or user:

1. Use USER-ITEM similarity matrix and USER-USER similarity matrix:
   USER-USER similarity matrix is used to find a group of users that are similar to each other, and we use USER-ITEM similarity matrix to find products that are frequently bought by users in the group, and recommend these products to users who are in the group but do not buy the products. This method is tend to recommend best sale products to users.

2. Use ITEM-ITEM similarity matrix
   We use this matrix to find a group of products related to the product that a user had bought, then recommend those related products to the user. Similarity could be derived from Jaccard Similarity or Cosine Similarity.

I suggest using ITEM-ITEM similarity matrix for memory based system, because ITEM-ITEM is relatively easy to recommend long tail products, which will have a beneficial effect on the company's overall revenue. Also, User-User requires large matrix size and the calculation is time consuming because I assume the number of users is higher then the number of products in Fetch Rewards. At this time, ITEM-ITEM can be used for faster calculations. Finally, USER-USER may have bias when the size of each group of users are not uniform distributed, that is, we may have a group with only a few users, then the prediction based on this group is error-prone.

A complete recommender system should implement both content based and memory based systems, there are two reasons listed below:

1. Memory based system gathers opinions of various people for recommendation and thus has more credibility than content based system.
2. Memory based system has Cold Start problem, the USER-USER matrix can't be used when new users have no purchase or browsing records, and they cannot calculate the similarity with other users. The ITEM-ITEM collaborative filtering method is the same. As this point, we should consider using "content based systems" to cover Cold Start problem.

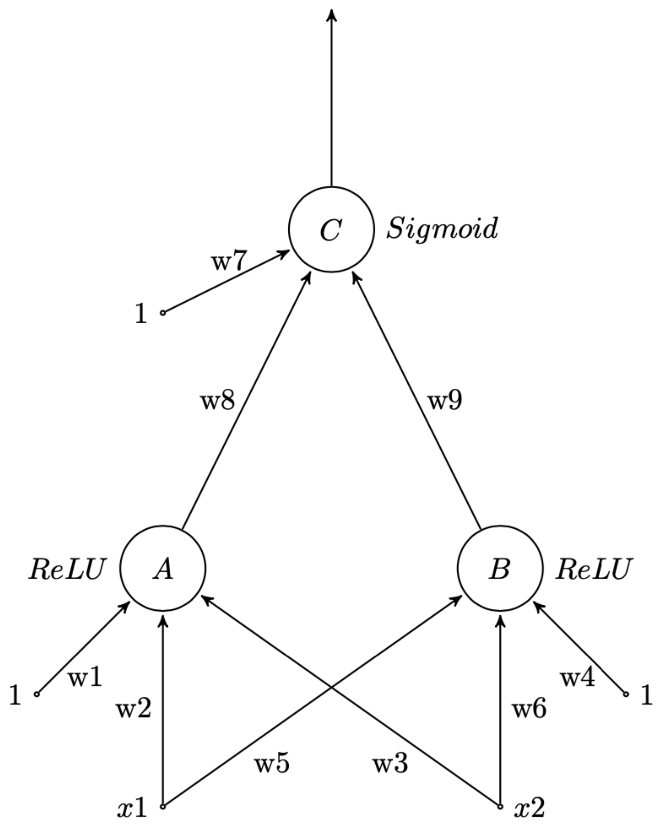
A model based recommender system can be deployed when the computation resources are limited: Memory based recommender system needs to calculate the similarity of each cell in the table, and consumes a lot of computing resources, so we can use model based system such as KNN or CNN to reduce the resources of calculating and storing the tables. I suggest using CNN since it involves non-linear computation, which is more similar to human behaviors.

DESCRIBE THE STEPS YOU WOULD FOLLOW TO DEVELOP THE SYSTEM

1. Design the system and make assumptions about its inputs and outputs. For example, the inputs of a CNN-based system are two attributes of the user, and the output is a prediction as to whether the user will buy a particular product. I can then train a simple neural network based on the users' attributes and submitting receipts. If the model predicts that a user will buy a product, we recommend that product to him / her.

2. We need to do some data wrangling before training a CNN model. Data wrangling includes gathering, assessing, cleaning about the data we want and it usually repeats several times before we are satisfied with the data. And we might even change the system design during the step of data wrangling, usually, data wrangling is the most time consuming step.

3. I am going to build a simple CNN model as some details listed below:
   - It is a neural network to perform binary classification with inputs x1 and x2.
   - Assume that we have three clean data sets about whether a vacuum machine is bought by users. They are CNNBased_train.txt, CNNBased_eval.txt, and CNNBased_test.txt.
   - Final layer use Sigmoid to output final result, 0 represents buying while 1 represents not buying.
   - Hidden layer consists of unit A and B, and they use ReLU to output non-linear result.
   - For each user, x1 and x2 are his/her two features.
   - Each unit also has a constant bias 1 input with the corresponding weight, w1, w4 and w7.
   - Set w1 = 0.1, w2 = 0.2, ..., w9 = 0.9, η = 0.1, T = 10000, where η is step size for updating weights, and T is the maximum epochs for training this neural network.
   - Diagram of the simple neural network is drawn below.
   - Please check CNNBased.java for details.



EXPLAIN HOW YOU WOULD DEFINE SUCCESS
A good definition of success for the recommender system:

1. Among the people who are recommended a product, the percentage of users who buy the product is high.
2. Among the people who are not recommended a product, the percentage of users who do not buy the product is high.

However, in my simple neural network, I define success as the accuracy of its prediction, in other words, users who bought the product should be estimated as 0 and who did not buy the product should be estimated as 1. The result of my simple network are listed below:
Accuracy: 0.96000
Evaluation Error: 0.87178
With total epochRun is 788, and final weights as w1: -3.51459, w2: 1.20379, w3: 4.45346, w4: -2.72310, w5: 0.93258, w6: 3.44859, w7: -3.99435, w8: 2.11219 and w9: 1.63381.

Usually, a small CNN model has at least three convolutional layers and pool layers, and the required sample size of each class / product should be at least 1000, and the required accuracy depends on different industries, products, etc. For example, the accuracy of my last CNN project in Foxconn is 92% with typeI error 0.01 and typeII error 0.326. Nevertheless, since I just want to present the key concept of CNN, the sizes of my made up sample sets are small and only one hidden layer is used.

[BONUS] DEMONSTRATE ONE OR MORE OF YOUR SUGGESTED APPROACHES IN ANY TOOL, ON ANY DATA SET OF YOUR CHOOSING, FROM ANY DOMAIN, REAL OR MADE-UP

Please check CNNBased.java