Abstract

To promote academic and practical research on big data, Chinese Association for Artificial Intelligence, IEEE-China and Toutiao organized an International Machine Learning Competition. The challenge problem is to match community raised questions with domain experts. Toutiao Q&A now has tens of thousand users answering on a daily basis, while user-created answers are viewed tens of millions times a day. An important question is how to match questions with interested users' expertise. If the matching strategy is not accurate enough, to ensure that questions get enough top quality answers, the system will have to send out invitations to more expert users who may be disturbed.

1 Introduction

The goal of this project was to develop a recommender system for the Chinese Q&A platform Toutiao Q&A . Toutiao Q&A is a mobil social platform with 580 million users and 300,000 professional freelance writers. Toutiao uses recommender systems to find the right respondents to questions and also the best audience to read the answers. A good recommender system makes meaningful recommendations of products or items to groups of people. The domain and the characteristics of the data are the two main aspects to be considered while developing a recommender system. The challenge at hand was to match community raised questions with domain experts. This paper describes the various recommender systems developed to solve the given challenge.

1.1 Problem

The data provided by Toutiao Q&A was used to conduct analysis on specific questions. The data consisted of Question tags, Expert tags and the question descriptions. Using this data, a recommender system had to be developed to predict which experts were more likely to answer which questions.

1.2 Data

The data consisted of three types of information :

i. Expert tag data: It consisted of the Expert IDs, their interest tags and also a description of their profile. The profile description was given as an ID sequence of words and an ID sequence of characters.

ii. Question Data: For each question, it consisted the Question ID, its category, preprocessed question description, total number of answers, number of top quality answers and the total number of upvotes. The Question description was also given as an ID sequence of words and an ID sequence of characters.

iii. Question Distribution Data: It consisted of 290,000 records of question push notification data. Each record consisted of the Question ID, the ID of the expert to whom the question was sent and a field indicating whether or not the question was answered.

iv. Validation and testing data without the labels were also provided.

1.3 Data Cleaning

The question distribution data consisted of a number of duplicate records. In order to eliminate duplicates, a combination of Question ID and Expert ID was created as a new field. The records were then sorted based on the values of this new field and duplicates were removed.

1.4 Creation of Validation and Training Set

In order to create an unbiased training and validation set, the given training set was divided into 80-20 ratio to train set such that the ratio of class labels in both train and validation remains same.

The training and validation set are randomly selected and thoroughly shuffled for cross validation and future usage or application of machine-learning algorithms.

2 Approaches

The following section defines and discusses about the machine-learning algorithms tried and implemented.

2.1 Content based recommendation system

These systems recommend items to a user based on the content of the items that were previously used or liked by the user. It mainly focuses on the content of the item and the user's preference profile.

In order to develop a content based system, we used Gensim's Word2Vec library. Word2Vec is a two-layer neural network that takes a text corpus as an input and produces a set of vectors. The vectors are nothing but word vectors for each word in the corpus. Word2Vec considers the semantics of the word. It also groups words of similar meanings together in the vector space and hence helps find similarities mathematically. Accurate guesses about a word's meaning can be made if we have enough data and context. The main idea behind using word2vec in this project was to capture the semantics of the question and user descriptions. Using this, one can predict the similarity between two or more questions and also similarities between the expert's areas of interest and the question domain. If a user had answered a question before, there could be a good probability of the same user answering a similar question. We considered the fields question category and the question description and expert interest tags and the expert profile description to obtain word vectors of size 100. Thus word vectors for each question's description and each user's description were obtained.

These word vectors were used as additional features in the question distribution data. These features were used to calculate the similarity between questions and users and accordingly recommend questions to experts. XGBoost classifier was used to predict whether an expert answers a question or not.

2.2 Collaborative Filtering Recommendation Systems

The systems collect large amounts of data to analyze the behavior of users and then predicts items based on the past ratings of all users collectively. They can be further classified into item-item based collaborative filtering and user-user based collaborative filtering.

In user-user based collaborative filtering, given an active user, we find k users with highest similarity to the active user. We then recommend items to the active user based on the items preferred by the users who are similar to the active user. This system might be computationally intensive when applied to millions of users as it requires computing similarities between all pairs of users. In Item-Item based Collaborative Filtering, rather than matching similar users, we match a user's rated item to similar items.

2.3 Cosine Similarity

$$sim(i,j) = cos(i,j) = \frac{\overrightarrow{i} * \overrightarrow{j}}{||\overrightarrow{i}||^2 * ||\overrightarrow{j}||^2} \tag{1}$$

User-User based Collaborative Filtering using cosine similarity: Here for every user u, we obtained a vector u = [q1, q2, ...qn] where qi = 1 if the user u has not answered a question i, else qi = 0. Thus we are representing a user by a n-dimensional vector. We can compute the similarity between two users by computing the cosine of the angle between their vectors as follows, For a given active user, we find the top k most similar users. Now, predictions are made from the weighted combination of the selected similar users.

Item-Item based Collaborative Filtering using cosine similarity:

In the context of our project, the question to be answered represents an item. Here for every question q, we obtained a vector q = [u1, u2, ..., um] where uj = 1 if a user u has answered question j else uj = 0. Thus we are representing a question by a m-dimensional vector. The similarity between two questions can be computed by obtaining the cosine of the angle between their vectors. The disadvantage of using cosine similarity is that the difference in the mean and variance of the probability of being answered for question i and question j is not considered.

2.4 Item-item based Collaborative System using Pearson similarity:

This system ranks an item based on its similarity to other items observed for the given active user. Thus, we are computing the similarity between two questions by using the observations of the users who have answered both questions under consideration. Given the similarity between two questions i, j, the probability of user u, answering question is calculated using the weighted average of the user's previous interactions. The similarity between two questions can be computed using Pearson Correlation as follows:

$$sim(i,j) = \frac{\sum_{u \in U}(R_{u,i} - \bar{R_i})(R_{u,j} - \bar{R_j})}{\sqrt{\sum_{u \in U}(R_{u,i} - \bar{R_u})^2 \sum_{u \in U}(R_{u,j} - \bar{R_u})^2}} \tag{2}$$

where, U is the set of users who answered both questions i and j, represents whether user u has answered question i or not. Predictions using Pearson similarity are made as follows,

$$P_{u,i} = \frac{\sum_{allsimilaritems,N}(S_{i,N} * R_{u,N})}{\sum_{allsimilaritems,N}(\|S_{i,N}\|)} \tag{3}$$

where, R represents the user's previous observations.

2.5 Matrix Factorization :

Matrix factorisation is one of the most promising method for recommendation engines. We have used various methods for the same. Non-negative matrix factorisation using scikit was the first and straightforward approach . However due to large training set and single threaded program we switched to a parallel matrix factorisation library LIBPMF, the Non-negative matrix factorisation was several folds faster. Finally we used a sophisticated library called Graphlab for the implementation of Matrix factorization

2.6 Content Boosted Collaborative System:

Here predictions were made by using the above mentioned content based system. These predictions were then applied to convert the sparse users rating matrix into a full ratings matrix. Then, item-item based collaborative filtering was used to make predictions. Content Boosted Collaborative systems help us overcome the two fundamental problems of Collaborative Filtering Systems, i.e., Sparsity and First – rater problem.

3 Summary

Below is the summary of all the algorithms tried and implemented. All the algorithms have been run on our beforehand created validation and train sets which are randomized and shuffled as mentioned before.

Also, in order to simulate the ndcg score, the given ndcg.py script was manipulated to suit our needs such that its results will help us improve our accuracy rather than wasting our validation submissions and test submissions in the competition website.

3.1 Summary in Tabular Format:

Algorithm	Brief Description	Score on Validation Set
Neural Network	For multiple layer neural network, we used 3 hidden layers, with the first layer having large number of nodes, and reduced the number of nodes by a factor of 2. We used the tag features (din=143), with Relu activation for the hidden layers and softmax for the output. We used the keras libraries with theano setting on google cloud.	0.21586
Neural Network grid search	The multiple layer neural network with 3 hidden as described above was used but now the output layers was changed to sigmoid activation. Additionally, we tried to use the grid search for the best hyper parameters. We used the keras libraries with theano setting on google cloud	0.22436
All unanswered	It was observed that most of the labels in the given data was found to be 0, we wanted to try what if we have all the invited questions left unanswered by the users.	0.22211
Average probability based mix 1s	Further it was observed, the average of the questions been answered by users is less than 0.12, we tried to randomly mix some very small amount of 1s' in the labels and see the impact. All the labels were either 1 or 0.	0.22497
User based CF with Cosine similarity (qid-qid)	Cosine similarity is the simplest and popular technique applied in collaborative filtering. For a given question we used the top 10 most similar question and used that for computing the probability to answer the question. We used the sklearn cosine similarity for computing the similarity.	0.46512
Item based CF with Cosine similarity (uid-uid)	Similarly, we considered the item-based collaborative filtering. As most of the data was zero, we wanted to pick only the answered similar questions with equal probability instead of how closely are they similar.	0.38083
User based CF with Modified Cosine similarity qid-qid (top 10)	As the user based collaborative filtering is more promising than the item-based collaborative filtering. We modified the prediction formula to treat each similar questions with equal weightage and by not multiplying with the similarity between the questions. This resulted in given better result, which can be attributed to the sparsity of ground truth available and also the low average probability of questions getting answered.	0.48686
User based CF with Pearson (uid- uid)	As the Pearson correlation gives us better similarity measure compared to the cosine and jaccard, we wanted to compute our prediction based on it. The result showed a lot of improvement. We used graphlab for this computation as the scipy one was taking too much time.	0.48881
User based CF with Matrix Factorization	The latent factor based collaborative filtering had shown good result in the famous Netflix competition and we wanted to proceed with the same idea. We used graphlab for these computation on google cloud. The graph lab uses the appropriate solved – stochastic gradient descent or adaptive sgd for the computation.	0.50036

Algorithm	Brief Description	Score on Validation Set
Simple recommender System	Graphlab offers a number of recommendation system flavors for content based and collaborative based. It also offers a simple recommender system which tried but it did not perform that well compared to matrix factorization.	0.45399
User based CF with MF and hybrid approach	So far we were only predicting based on the question-user matrix in line with the training data. Further, we wanted to supply the user information and the question information master data. The computation was much slower compared to just the matrix, and converged very slowly. It was observed that the solver may get stuck in lower minima more often with this approach.	0.40632
Content based recommendation with xgboost	The content based model was developed using the tags, words and characters to predict the probability of the question been answered by the user. We use converted these features in to vectors using the genism library. We used the xgboost algorithm from scikit to formulate and predict the probability.	0.27322
Neural network (Deep Learning)	The multiple layer neural network with 3 hidden as described above was used but now the output layers was changed to sigmoid activation. With different set of features with relu, sigmoid activations in hidden and final layers	0.224469
Optimizing hyper parameters - User based CF with MF	For optimising hyperparameters using cross validation with ranking based Matrix factorisation on graphlab on all parameters without limiting to any	0.500518
Optimizing hyper parameters - User based CF with MF	Optimising hyperparameters to only REGULARIZATION coefficient only improves accuracy or ndcg score	0.501058
Content Boosted Collaborative Filtering	The predictions were made using content based recommender. These predictions were used to fill up the ratings matrix. Then, Collaborative Filtering was done using cosine similarity.	0.46802

References

[1] Recommender Systems, Encyclopedia of Machine Learning, Prem Melville and Vikas Sindhwani (http://www.prem-melville.com/publications/recommender-systems-eml2010.pdf)

[2].Collaborative Filtering Recommender Systems, Michael D. Ekstrand, John T. Riedl and Joseph A. Konstan (http://files.grouplens.org/papers/FnT%20CF%20Recsys%20Survey.pdf)

[3]. A survey of collaborative filtering techniques, Xiaoyuan Su, Taghi M. Khoshgoftaar. (https://www.hindawi.com/journals/aai/2009/421425/)

[4]. BPR: Bayesian Personalized Ranking from Implicit Feedback, Steffen Rendle, Christoph Freudenthaler, Zeno Gantner and Lars Schmidt Thieme (http://arxiv.org/abs/1205.2618)

[5]. Everyone Likes Shopping! Multi-class Product Categorization for e-Commerce, Zornitsa Kozareva (http://www.aclweb.org/anthology/N15-1147)

[6]. A Survey of Accuracy Evaluation Metrics of Recommendation Tasks, Asela Gunawardana and Guy Shani (http://jmlr.csail.mit.edu/papers/volume10/gunawardana09a/gunawardana09a.pdf)