Recommendation System for Toutiao Q&A

Disha Punjabi
Department of Computer Science
University of Southern California
Los Angeles, CA 90007
dpunjabi@usc.edu

Saba Chaudhary
Department of Computer Science
University of Southern California
Los Angeles, CA 90007
sabachau@usc.edu

Sharanya Vishnampitte Raju
Department of Computer Science
University of Southern California
Los Angeles, CA 90007
sharanyr@usc.edu

Abstract

The 2016 BYTECUP challenge problem is to come up with a matching strategy to match community raised questions with individual experts. This is an interesting problem that can possibly be modeled by using supervised learning algorithms, training neural networks, or using recommendation algorithms. Content-Based Filtering coupled with Non-Negative Matrix Factorization algorithm to detect latent features gave us the best results compared to the other approaches that we implemented. The main challenge was identifying features that would give good results on the validation set while ensuring that the model did not overfit. In this paper, we present all the approaches involved in the process of coming up with the optimal matching strategy.

1 Introduction

Recommendation systems have been existing for long and have been applied to various domains. Various approaches in the area of recommendation systems itself are prevalent. The Toutiao Q & A is a mobile social platform that promotes short form content creation and interaction on mobile devices. Our task deals with coming up with a better recommendation algorithm that matches information with the right people, precisely matching questions with interested users' expertise.

2 Dataset Analysis

The dataset consists of expert attributes and question attributes along with the training file. There are 8094 number of unique questions of which there are 7708 questions in the training file. There are 28762 unique experts of which 27127 are in the training file. Features like Word ID and Character ID, Question Tag and Expert Tag were categorical while no. of upvotes, total questions answered, top quality answers were numerical. On converting to binary features, we get a total of 168 features each for both questions and experts. The training data provided for answered questions forms only 11.118%(27323) of the total 245752 training records whereas the ignored/not answered questions formed 88.8814% (218428) of the same.

3 Data preparation

3.1 Preprocessing

3.1.1 Converting categorical features to binary features

Given a combination of categorical and numerical features, we converted our categorical features into binary features using one-hot encoding indicating whether a particular feature is present in the training sample or not.

3.1.2 Scaling features

Training data was standardized by subtracting the mean value of numeric features, then scaled by dividing non-constant features by their standard deviation, resulting in normalized numeric features, compatible with binary ones.

3.1.3 Removing redundant features

We removed redundant columns that did not give us a difference in the results and was constant such as the word and character descriptions of questions and experts. This reduced our training data size to 168 features.

3.1.4 Dimensionality reduction

Training data was huge with many features having skewed distribution of classes(as discussed in the Data Analysis above). This necessitated identification of the most informative features for which we applied dimenstionality reduction, by which we reduced the number of features under consideration and transforming the overall feature-space, by obtaining a set of principal variables. We attempted to apply Autoencoders since the characteristics of the classes were not reflected in the variance of the features. This prohibited us from applying simpler techniques like Principal Component Analysis. The enormity of dataset also restricted us from using Singular value decomposition. Due to time constraints, we could not efficiently use the Autoencoder on the entire data to get a good reduced representation of the training data.

3.2 Formatting

Both the expert and user information were processed to extract user and question IDs, and other features. Different combinations of features were used for different algorithms as outlined below:

- For implementing non-negative matrix factorization, an expert-questions matrix was built: An expert-list and a question-list was extracted from the given files and corresponding indices of the expert and question were used to form an expert-question matrix, each cell indicating whether the question was answered or ignored.
- In an initial implementation of Content-Based Filtering, a dictionary was used to map expert ID to the no of questions answered by category(i.e. Question Tag) by that expert. Likewise from question to list of expert IDs who answered that question. Ignored questions were not not taken into consideration in this implementation.
- In subsequent iterations, the word ID description of experts was stored in the form of a dictionary making use of question description.

4 Algorithms

4.1 Content based filtering

This methods finds its roots in the field of Information retrieval where each entity has some properties. These features include the properties of questions under consideration like number of up-votes, total number of answers of the question, popular answers given for the question and description of the question. For a given expert under consideration, we can consider features like the categories of interest, description of the expert and so on.

4.2 Collaborative filtering

4.2.1 Neighbourhood-based collaborative filtering

CF Systems are a type of Recommender systems in which in order to recommend an item to a user, the past ratings of all users in his neighbourhood for that item are collectively used to determine the significance of that item, to the user under consideration. In this approach, similarity measures like Pearson correlation coefficient (which measures the extent to which there is a linear dependence between two variables), cosine similarity(which is the cosine of the angle between the m-dimensional ratings vector of two users) etc. are used to identify the similarity between an active user a and any other user u.

In our initial implementation, we took the experts as the users and the Question Tag as the items and the ratings $r_{a,i}$, $r_{u,i}$ as the no. of questions answered by the expert in that tagged with this Tag. Then the $w_{a,u}$ scores were calculated using equation as below-

$$w_{a,u} = \frac{\sum_{i \in I}(r_{a,i} - \bar{r_a})(r_{u,i} - \bar{r_u})}{\sqrt{\sum_{i \in I}(r_{a,i} - \bar{r_a})^2 \sum_{i \in I}(r_{u,i} - \bar{r_u})^2}}$$

Then a prediction of the no of questions with that Question Tag that would be answered was computed as the weighted average of deviations of $r_{u,i}$: no of questions answered by neighbour u in category i, from his own mean: $r_u bar$ as below:

$$p_{a,i} = \bar{r_a} + \frac{\sum_{u \in K}(r_{u,i} - \bar{r_u}) * w_{a,u}}{\sum_{u \in K} w_{a,u}}$$

Finally the values were normalized for each user across all categories represented by question tags to get the probability of answering any question tagged with that Tag. The probability of answering a question was taken as the probability values predicted using the above equation per Tag. Only Question Tag (which indicated question category) was used as a feature for arriving at the prediction in the initial implementation. Only the questions that were answered by an expert in invited_info data were used in arriving at the predictions, the questions that were ignored by the expert were not included in the calculations(and instead considered in next iteration) which is why we believe that the ndcg score on the validation set came out to be 0.29.

In subsequent implementations, the item was taken as the question, user as the expert and the rating as whether the question was answered or not. The ndcg rating on the training set using this approach was found to be ..... Improvements made in the subsequent iterations included subtle modifications in the calculations of prediction $p_{a,i}$ by taking the mod of $w_{a,u}$ in the denominator of the equation and also using the questions that were ignored by the expert from the training set in our calculation.

The steps undertaken are generalized as follows:
(i) Assigned a weight to all users with respect to similarity with the active user.

(ii) Selected k users that have the highest similarity with the active user (the neighborhood).

(iii) Computed a prediction from a weighted combination of the selected neighbors' ratings.

4.2.2 Non-negative matrix factorization

The expert-question matrix that resulted from data formatting was sparse with 88% zeroes and 11% ones. Using this matrix, we used the following equation to capture the hidden factors that influence the response of the expert to a given question.

$$score(i, j) = \mu + w_i + w_j + u_i^T v_j$$

Here, u_i represents the feature set of the experts and v_j represents the features for a given question. Using law of matrix multiplication, the dimensionality of u_i^T is same as v_j which is equal to 'k'. This k, determines the number of latent factors mentioned above. By splitting the matrix in this fashion, we not only try to find u_i and v_j but also attempt to predict the missing values in the matrix(say matrix $y_h at$) or $score(i, j)$ matrix (all being non-negative)

This matrix is obtained as a result of taking a dot product of u_i and v_u and adding the weights of an expert's weight/ prior probability of answering any question as w_j and question's probability of being answered as w_i.

4.3 Hybrid recommender

We then combined the content based filtering and non-negative matrix factorization which generated better predictions(also the best amongst all) for an expert answering the question under consideration. This approach is referred to as Hybrid approach in the recommendation system world.

$$score(i,j) = \mu + w_i + w_j + a^T x_i + b^T y_j + u_i^T v_j$$

The terms x_i and x_j represent the question and expert vectors that capture the content-based feature information. the vectors a^T and b^T capture the weights estimated by our model.

For purposes of evaluation, we attempted to minimize the RMSE between the new and original matrix only on the values that were present in the ground truth matrix. For simplicity, we have assumed the unknown values to be 0.

The main task here is not to just reproduce the values from the ground truth matrix but to also predict the missing values, which would act as the chances of an expert responding to an unseen question.

$$min_{w,a,b,V,U} \frac{1}{|D|} \sum_{(i,j,r_{ij}) \in D} L(\text{score(i,j)},r_{ij}) + \lambda_1(||w||_2^2 + ||a||_2^2 + ||b||_2^2) + \lambda_2(||U||_2^2 + ||V||_2^2))$$

We attempt to reduce chances of overfitting by regularizing the equation as below. We model the loss function as sum of RMSE and adding the penalty of having a complex model, by multiplying lambda with square of modulo 2 for each feature vector for question and expert. We also regularize the side feature vectors and weights of the experts and questions.

$$L(\hat{y}, y) = (\hat{y} - y)^2$$

Optimization There are multiple ways to attempt optimization of the above loss function. We used adagrad as our solver since it converges easily and is historically known to generate more reliable results as compared to simple SGD, when there can be uneven scaling of weights. Ada grad decreases the step-size as a function of time.

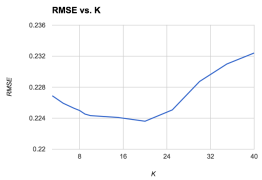

Figure 1: Effect of number of latent features on RMSE

Results The algorithm non-negative matrix factorization with 10 fold cross-validated hyper-parameter tuning, resulted in a RMSE value of 0.2606 on the training set. Combining this approach with the content/features of the experts and questions, the RMSE dropped to around 0.2214.

4.4 Decision tree

We tried to model this problem as a classification problem by implementing Decision Tree Algorithm to calculate probability of an expert answering a particular question. Decision trees are predictive models where leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values are called regression trees.

Parameters We used gini index as a criterion for splitting and did not set any max-depth for it.

Results Decision trees did not perform well with this problem. We evaluated our model using ROC and the values ranged between (0.38, 0.50) which is a random score. We got an NDCG score of 0.234681168292444 on validation data

5 Conclusion

In this paper we have presented a range of collaborative and content-based approaches to expert recommendation on questions. Our algorithms were evaluated on NDCG scores. For content-based filtering, using tags for calculating item similarity alleviates sparsity and results in better performance. The approach that outperformed Collaborative, Content and other supervised algorithms is the Hybrid Recommender with an NDCG score on validation of 0.49636 and test of 0.48196

Acknowledgments

This paper is a part of mini project assigned in CSCI 567 Machine Learning course at University of Southern California. We would like to thank our Professor, Dr Yan Liu and Teaching Assistants, Nitin Kamra and Parissa Monsouriford for giving us the opportunity to practically implement various machine learning techniques on a real-world problem and for guiding us with the right resources and in the right direction with their valuable inputs throughout the project.

6 References

[1] Prem Melville & Vikas Sindhwani (2010) *Recommender Systems, Encyclopedia of Machine Learning.* Encyclopedia of Machine Learning.

[2] Michael D. Ekstrand John T. Riedl & Joseph A. Konstan (2010) *Collaborative Filtering Recommender Systems.* Foundations and Trends in Human–Computer Interaction Vol. 4, No. 2.

[3] Yehuda Koren Robert Bell & Chris Volinsky(2009)*Matrix Factorization Techniques for Recommender Systems.* IEEE Computer, Vol. 42 , pp. 30-37

[4] G. E. Hinton & R. R. Salakhutdinov (2006) *Reducing the Dimensionality of Data with Neural Networks.* SCIENCE, Vol. 313

[5] Toine Bogers & Antal van den Bosch (2009) *Collaborative and Content-based Filtering for Item Recommendation on Social Bookmarking Websites.* ACM RecSys

Table 1: Score Results

Algorithm		
Name	Description	Score (ndcg)
Content Based Filtering	Content-based filtering recommends (Questions) based on a comparison between content of the questions and an expert profile	0.29161
Collaborative Filtering	Collaborative filtering filters information by using the recommendations of other expert	0.4595
Hybrid Filtering	A hybrid approach, combining collaborative filtering and content-based filtering - recommends based on expert information as well as the side content of Questions	0.49656
Decision Trees	Predictive algorithm where each branch represents the outcome of a test, and each leaf (or terminal) node holds the probability of an expert answering a question	0.23142