Expert Finding in Community Question Answering: A Review

Sha Yuan*, Yu Zhang†, Jie Tang*§ and Juan Bautista Cabotà‡
*Knowledge Engineering Lab, Department of Computer Science and Technology, Tsinghua University
†Institute of Medical Information, Peking Union Medical College, Chinese Academy of Medical Sciences
‡Computer Science Department, University of Valencia

***Abstract*—The rapid development recently of Community Question Answering (CQA) satisfies users quest for professional and personal knowledge about anything. In CQA, one central issue is to find users with expertise and willingness to answer the given questions. Expert finding in CQA often exhibits very different challenges compared to traditional methods. Sparse data and new features violate fundamental assumptions of traditional recommendation systems. This paper focuses on reviewing and categorizing the current progress on expert finding in CQA. We classify all the existing solutions into four different categories: matrix factorization based models (MF-based models), gradient boosting tree based models (GBT-based models), deep learning based models (DL-based models) and ranking based models (R-based models). We find that MF-based models outperform other categories of models in the field of expert finding in CQA. Moreover, we use innovative diagrams to clarify several important concepts of ensemble learning, and find that ensemble models with several specific single models can further boosting the performance. Further, we compare the performance of different models on different types of matching tasks, including *text vs. text*, *graph vs. text*, *audio vs. text* and *video vs. text*. The results can help the model selection of expert finding in practice. Finally, we explore some potential future issues in expert finding research in CQA.**

***Keywords*—Expert finding; matrix factorization; deep learning; ensemble learning**

I. Introduction

With the increasing demand of knowledge sharing services, Community Question Answering (CQA) websites, such as Quora, Toutiao and Zhihu, have already obtained the popularization use in reality. It is common to post questions and answers on CQA websites, where users' quest for professional and personal knowledge in various domains can be satisfied. The central task of CQA is to find appropriate users with willingness and relevant expertise to provide high-quality answers for given questions. This problem has been extensively studied in the past decade. Related research includes expert finding for community-based questions [1], [2], expertise modeling [3], and even a comprehensive survey [4]. Though this problem has been studied before [5], the willingness of experts has been often ignored. This problem becomes more and more seriously – more than half of the questions on Quora only have one or even do not have any answers[1].

Expert finding in CQA have generated huge impact to society. It provides a platform to connect questions with experts who can contribute quality answers. Questions about anything can be solved by crowdsourcing in CQA. For example, CQA can help to find a mathematician for a chef with a math problem. At the same time, cooking tips from the chef will be returned to the mathematician if necessary. However, it is often hard for CQA to establish such high-quality expert finding. How to match the questions with interested users' expertise? Can we predict who are the most likely to answer the given questions and what is the probability? Confronting these challenges, the focuses of expert finding in CQA have changed in practice.

Traditional expert finding problem focused on expert finding [1] and expertise ranking [2]. The experts would be found for the given question based on text matching. In recent years, the core value of the problem is not finding expert, but solving problems by crowdsourcing. Moreover, expert finding in CQA often exhibits very different challenges compared to traditional methods. The characteristics of expert finding in CQA is summarised as follows.

First, **crowdsourcing**. The complex and intellectively demanding problems in CQA requires considerable effort and quality contribution. Crowdsourcing is channeling the experts' desire to solve a problem and then freely sharing the answer with everyone. In CQA, the answer of the given question would be obtained by crowdsourcing from a large, relatively open and often rapidly-evolving group of interested experts.

Second, **sparse data**. The known question and answer pairs are rare compared to traditional expert finding applications. On one hand, seekers spend more time on finding the answer of their question. On the other hand, experts need to answer multiple versions of the

§Correspondence: Tsinghua University, Haidian District, Beijing, China. E-mail: jietang@tsinghua.edu.cn.

[1]https://www.quora.com/What-percentage-of-questions-on-Quora-have-no-answers.

same question. This also makes it challenging to directly use a supervised learning approach due to the lack of training samples.

Third, **new features**. The willingness of expert, the historical behavior of expert, and the quality of answer, all these new features have got more attention. They may contribute to further improve the rationality and effectiveness of expert finding in CQA. For example, the expert who often provides answers with high quality is more likely to answer the similar kinds of questions. How to use these features effectively is widely acknowledged as new challenge that can improve the performance of expert finding in CQA.

Despite of the above challenges, once such expert finding in CQA is successfully formed, its impact is usually tremendous. Based on these observations, most well-known CQA websites and competitions, such as Quora, Toutiao and Kaggle are striving to match questions with interested users' expertise, that is, to find the best respondents to the questions. As for this study, we have got the labeled datasets of the competition ByteCup[2] organized by Toutiao. And therefore we will take these datasets of Toutiao as an example to review the methodologies for expert finding in CQA in the following parts of this paper.

In this paper, we firstly review all the existing expert finding solutions in CQA and classify all the solutions into different categories, including matrix factorization based models (**MF-based models**), gradient boosting tree based models (**GBT-based models**), deep learning based models (**DL-based models**) and ranking based models (**R-based models**). In addition, we illustrate the results of all the aforementioned categories of single models on the local validation dataset in the ByteCup competition, and specify the single model obtaining the best performance. The ensemble strategies of the Top 5 teams who won the competition are also analyzed. We use innovative diagrams to clarify several important concepts of ensemble learning. This work will significantly help the correct understanding and proper use of ensemble learning in practice. Further, we investigate the performance of different models on different types of matching tasks. Finally, we statistically analyze the results of all expert finding solutions in CQA, and summarize the work of this paper.

The remainder of the paper is organized as follows. In the next section, we first give a general overview. Sections III, IV, V and VI present the MF-based models, GBT-based models, DL-based models and R-based models, respectively. Section VII specifies the details of ensemble learning. Section VIII, IX and X present the results and the corresponding analysis. Finally, Section XI concludes the paper.

[2]https://biendata.com/competition/bytecup2016/.

II. Overview

A. A Brief History of Expert Finding

Inspired by recent advances in information management systems, expert finding has attracted a lot of attention in the information retrieval (IR) community [6]. The core task of expert finding is to identify persons with relevant expertise for the given topic. Massive efforts have been taken to improve the accuracy of experts finding [7] [8] [9] [10] [11] [12] [13] [14] [15]. Most existing methods for expert finding can be classified into two groups, including the *authority-based methods* [16] [17] [18] [19] [20], which are based on the link analysis of the past expert-topic activities, and the *topic-based methods* [21] [22] [23] [24] [25] [26] [27] [28], which are based on the latent topic modeling techniques. Moreover, the emerging deep learning models are integrated with aforementioned methods to further improve the performance of expert finding [29] [30] [31]. They are capable of effectively learning high dimensional representations of expert information, topic information and expert-topic interactions.

Expert finding has been researched in various areas such as academic [32], organizations [33] [34], social networks [35] [36] [37], and more recently question answering communities [38]. Finding experts with relevant expertise for a given topic has potential in many applications in these areas such as finding appropriate reviewers for a paper [39] [40], finding the right supervisor for a student in academic [41] and finding the appropriate experts for the questions in CQA [42].

CQA websites, which provide users with a platform to share their experience and knowledge, are very popular in recent years. Successful CQA websites include general ones (such as Toutiao, Quora and Zhihu), and domain-specific ones (such as Stack Overflow). Finding persons with relevant expertise for a specific question in CQA can increase the quality of answers and further improve the crucial problems facing by CQA, such as the low participation rate of users, long waiting time for answers and low quality of answers [43]. Expert finding in CQA is a challenging task which may due to the sparsity of the CQA data, and the emerging features. A great amount of studies have been conducted on expert finding in CQA [14] [44] [1] [45] [2]. Before we present the categorization of expert finding techniques, we first describe the notations and definitions used in this paper.

B. Problem Definition

We present required definitions and formulate the problem of expert finding in CQA. Our goal is to find experts to solve a given question in CQA in the way of crowdsourcing. More specifically, given certain question, one needs to find who are the most likely to 1) have the expertise to answer the question and 2) have

TABLE I
PERFORMANCE OF DIFFERENT CATEGORIES OF MODELS ON DIFFERENT TYPES OF MATCHING TASKS.

data type / Model category	text VS text	graph VS text	audio VS text	video VS text
MF-based models	√			
DL-based models		√		√
GBT-based models		√	√	
R-based models	√			

√ means that this category of models perform the best on that type of data.

the willingness to accept the invitation of answering the question.

Definition 1: **Expert** is the user with sufficient expertise for a certain **question** in CQA. The **expertise** are implied in relevant user documents, social interactions, past activities or personal information of each expert.

Given a set of M questions $Q = \{q_1, \cdots, q_M\}$, we need to predict which experts $E = \{e_1, \cdots, e_N\}$ are more likely to answer these questions. For simplicity, we reserve special indexing letters for distinguishing experts from questions, where u, v represent experts, and i, j represent questions.

Problem 1: For a given question i and its candidate expert $u \in E$, one needs to predict the **probability** $\hat{r}_{ui}$ of the expert u answering the question i.

The (u, i) pairs for which r_{ui} is known are stored in the set $Ł = \{(u, i) | r_{ui}\ is\ known\}$. The probability $r_{ui} \in [0, 1]$, high values mean stronger preference of the expert u to answer the question i. $\hat{r}_{ui}$ is the predicted probability that the question i will be answered by the expert u based on the labeled data. Here, it is a supervised learning problem to make prediction with the given labeled data. We need to infer a function from the labeled training examples, and then use the function to label the unknown data. In order to get the function, we need to reduce the error between $\hat{r}_{ui}$ and r_{ui}. Consequently, the objective optimization function is

$$L = \sum l(\hat{r}_{ui}, r_{ui}) \tag{1}$$

where l is the loss function.

Overfitting always happen. If we have too many features, the learned hypothesis may fit the training set very well, but fail to generalize the new examples. There are often two options to solve overfitting. The first is to reduce the number of features. The details is dependent on the specific problem. The second is regularization, which is used to reduce magnitude or values of each feature with parameter θ. It often works well when there are a lot of features, and each of them contributes a bit to the prediction $\hat{r}_{ui}$.

For example, if we use L2-norm for regularization, the optimization problem is transformed into the following problem:

$$\mathbf{\Theta}^* = \underset{\Theta}{\arg\min} \sum (l(\hat{r}_{ui}, r_{ui}) + \sum_{\theta \in \Theta} \lambda_\theta \theta^2) \tag{2}$$

where λ_θ is the regularization coefficient of parameter θ used in the hypothesis function. As it grows, regularization becomes heavier. Then, we need to find an appropriate optimization method to solve this optimization problem. In this way, we get the parameters of the prediction model, which can be used to label the unknown data.

Typical data in CQA implies large interaction between experts and questions. For instance, some experts prefer to answer than others, and some questions are more likely to be answered than others. In order to account for these affects, it is customary to adjust the data with baseline.

Definition 2: The **baseline** for the prediction $\hat{r}_{ui}$ is denoted by b_{ui}:

$$b_{ui} = \mu + b_u + b_i, \tag{3}$$

in which, the overall average probability is denoted by μ; the parameters b_u and b_i indicate the observed average deviations of expert u and question i, respectively. For example, suppose that we want to get a baseline for the probability of the question i answered by the expert u. The average probability over all questions $\mu = 0.6$. The expert u tends to answer question lower than the average with probability 0.3, so $b_u = 0.3 - 0.6 = -0.3$. The question i tends to be answered with probability 0.7, so $b_i = 0.7 - 0.6 = 0.1$. Thus, the baseline for question i answered by expert i is $b_{ui} = 0.6 - 0.3 + 0.1 = 0.4$.

C. Categorization of Expert Finding Techniques

Based on the survey of possible solutions, we categorize the techniques of expert finding in CQA under four subsettings, including MF-based models, GBT-based models, DL-based models and R-based models. Table. I shows the cases where the different approaches are used.

As shown in the Table. I, we summarize the performance of these models on different types of matching tasks to explore the scope of application[3]. In the table,

[3]More details of experiment results will be clarified in Section IX.

text VS text means to match text labels with text data, *graph VS text* means to match text labels with graph data, *audio VS text* is to match text labels with graph data; *video VS text* is to match text labels with video data.

We come to the conclusion that MF-based models usually achieve the best performance in the situation of *text VS text*, while DL-based models are rarely used in these situations and not performing well due to the severe sparsity of the text datasets. In addition, R-based models have significant performance in the situation of *audio VS text*, DL-based models often achieve the best in the situation of both *graph VS text* and *video VS text*, which may due to their outstanding power of capturing high dimensional features from graphs and videos. We will discuss these four category solutions in detail below. In addition, ensemble learning of these models will also be discussed.

III. Matrix Factorization Based Models

Matrix factorization (MF) [46], which is a common technique for collaborative filtering (CF) [47], covers a wide range of applications in recommender system with its variants. The *Problem 1* can be modeled as recommendation problem solved by CF, because similar users may answer the similar questions. Therefore, MF can be applied to exploit latent information from data. In this part, we summarize the MF-based models, including MF, Singular Value Decomposition (SVD), SVD++, Bidirection SVD++, Bidirection Asymmetric-SVD (ASVD++) and Factorization Machine (FM).

A. MF

From the application point of view, MF can be used effectively to discover the latent features underlying the interactions between different kinds of entities. For example, several experts have answered same questions before as illustrated in Fig. 1. If some of them (assume the number is N) answer a new question, others may also answer the question (assume the probability is p). N is larger, p is larger.

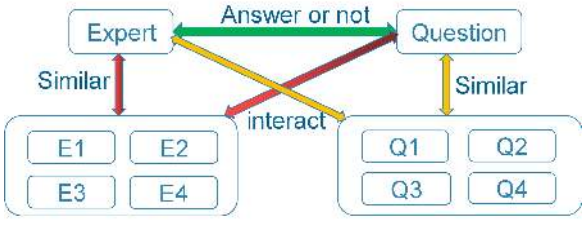


Fig. 1. Implied Information

From the mathematical point of view, MF is used to factorize a matrix obviously as its name suggesting. The original matrix can be represented by the multiply of two (or more) simple matrices with lower dimension. Let U and D be the set of experts and questions, respectively. Let $\mathbf{R}$ be the record matrix of the expert-question pairs. If we would like to discover k latent features, we need to find two matrices $\mathbf{P}$ (a $|U| \times k$ matrix) and $\mathbf{Q}$ (a $|D| \times k$ matrix) such that their product approximates $\mathbf{R}$:

$$\hat{\mathbf{R}} = \mathbf{P}^T \times \mathbf{Q} \approx \mathbf{R}. \tag{4}$$

Thus, matrix factorization maps experts and questions to a joint latent factor space of dimensionality k. Each row of $\mathbf{P}$ would represent the strength of the associations between a expert and the features. Similarly, each row of $\mathbf{Q}$ would represent the strength of the associations between a question and the features.

Matrix factorization maps experts and questions to a joint latent factor space of dimensionality k, such that expert-question interactions are modeled as inner products in that space. The resulting dot product $p_u^T q_i$ captures the interaction between expert u and question i.

$$\hat{r}_{ui} = p_u^T q_i. \tag{5}$$

Then we directly model the observed probabilities only, while avoiding over-fitting through a regularized model. To learn the factor vectors p_u and q_i, the system minimizes the regularized squared error on the set of known probabilities:

$$\min_{P,Q} \sum_{(u,i)\in Ł} (r_{ui} - q_i^T p_u)^2 + \lambda(\|p_u\|^2 + \|q_i\|^2) \tag{6}$$

where aforementioned Ł is the set of the (u, i) pairs for which r_{ui} is known.

B. SVD

One benefit of the matrix factorization approach to collaborative filtering is its flexibility in dealing with various data and other application-specific requirements. Eq. (5) tries to capture the interactions between users and questions without taking the baseline into consideration. Here we combine Eq. (3) and Eq. (5) as follows:

$$\hat{r}_{ui} = b_{ui} + p_u^T q_i \tag{7}$$

The system learns by minimizing the squared error function, and avoids over-fitting through an adequate regularized model:

$$\min_{P,Q,B} \sum_{(u,i)\in Ł} (r_{ui} - \hat{r}_{ui})^2 + \lambda(\|p_u\|^2 + \|q_i\|^2 + b_u^2 + b_i^2) \tag{8}$$

C. SVD++

MF and SVD models only consider explicit feedback which comes from the interaction between a user and a question. However, we can also obtain implicit feedback from the training data. For instance, a user prefers those questions that he answers in the past. Recommender

systems can use implicit feedback to gain insight into user preferences. Indeed, we can gather behavioral information regardless of the user's willingness to provide explicit ratings. Here, we try to integrate both explicit feedback and implicit feedback. We could get more accurate results by a direct modification of Eq. (7):

$$\hat{r}_{ui} = b_{ui} + q_i^T(p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j) \quad (9)$$

where $N(u)$ is the set of questions that user u has received invitation. A user u is modeled as $p_u + |N(u)|^{\frac{1}{2}} \sum_{j \in N(u)} y_j$. p_u is learnt from the given explicit ratings and $|N(u)|^{\frac{1}{2}} \sum_{j \in N(u)} y_j$ represents the perspective of implicit feedback. Here, a new set of item factors are necessary, where question j is associated with $y_j \in \mathbb{R}^f$. Model parameters are learnt by minimizing the squared error function.

$$\min_{P,Q,B,Y} \sum_{(u,i) \in \text{Ł}} (r_{ui} - \hat{r}_{ui})^2 + \lambda \|\theta\|^2 \quad (10)$$

where θ represents the parameters of the model. SVD++ [48] does not offer the benefits of having less parameters, conveniently handling new users and readily explainable results. This is because the model does abstract each user with a factors vector. However, SVD++ is clearly advantageous in terms of prediction accuracy than SVD.

D. Bidirection SVD++ (SVD#)

Appending another part of implicit feedback to the original SVD++ model, a new model named bidirection SVD++ model (also called SVD#) is built. The formula of this model turns to be:

$$\begin{aligned} \hat{r}_{ui} = b_{ui} + (q_i + |R(i)|^{-\frac{1}{2}} \sum_{j \in R(i)} x_j)^T \cdot \\ (p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j) \end{aligned} \quad (11)$$

$R(i)$ is the set of users who answer question i. Here, each question j is associated with $x_j, y_j \in \mathbb{R}^f$. The other parts of the formula are the same as original SVD++ model.

This model shows the power of representing user/question embeddings using the neighborhood question/user embeddings. However, the embeddings here are static and indepent of time. When the time information is available, a more powerful proposed in [49] will be helpful. This method incorporates the embedding co-evolving idea with time series models. The evolution of each user/question embedding depends not only on its old embeddings, but also the embeddings of question/user it interacting with.

E. Bidirection ASVD++

As mentioned in [48], instead of providing an explicit parameterization for users, users can be represented through the items that they prefer. This model named "Asymmetric-SVD"(ASVD) offers several benefits: (1) fewer parameters; (2) handle new users; (3) explainability; (4) efficient integration of implicit feedback. Combining the "bidirection" strategy described in Sec. III-D, there is a new model named bidirection ASVD++ model. The formula is listed as below:

$$\begin{aligned} \hat{r}_{ui} = b_{ui} + (|R(i)|^{-\frac{1}{2}} \sum_{j \in R(i)} x_j)^T \cdot \\ (p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j) \end{aligned} \quad (12)$$

F. Factorization Machine

FM [50] is a generic approach based on matrix factorization to mimic most factorization models. libFM [51] proposed by Steffen Rendle is a software implementation for factorization machines. It combines the generality of feature engineering with the superiority of factorization models in estimating interactions between variables of large domain. FM model has the following advantages. Firstly, variable interactions are embedded in the FM model. Secondly, it is able to reliably estimate parameters under very high sparsity. Thirdly, the equation, which depends only on a linear number of parameters, can be computed in linear time. Forthly, it can be applied to a variety of prediction tasks, including regression, binary classification and ranking. In essence, FM model is a matrix factorization based machine learning model and it is similar to linear regression model. We all know the linear regression model has the following formula:

$$\hat{y}(x) = w_0 + w_1 x_1 + ... + w_n x_n = w_0 + \sum_{i=1}^{n} w_i x_i. \quad (13)$$

where x_i is the feature and $\hat{y}$ is the predicted value.

On the basis of model above, if we consider the feature combination, the formula will be changed to the following form:

$$\hat{y}(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{ij}^{'} x_i x_j. \quad (14)$$

Because the sparsity of the feature, we find that many $w_{ij}^{'}$ will be zero after the training. Thus, in order to reduce the number of parameters, FM models the problem by the following formula:

$$\hat{y}(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (V_i^T V_j) x_i x_j, \quad (15)$$

where V_i is the latant vector of the i^{th} feature. We consider a maximum likelihood problem with Eq. (15).

To avoid over-fitting, we add some regularization terms. That is, we solve the following optimization problem for FM model.

$$\min_{W,V} \sum_{i=1}^{n} (y_i log(\sigma(\hat{y_i})) + (1-y_i) log(1-\sigma(\hat{y_i}))) + \frac{\lambda}{2} \|\theta\|^2 \tag{16}$$

where θ represents the parameters of the model and $\sigma(x)$ is the sigmoid function. The learning algorithm of FM mainly contains [51]: Stochastic Gradient Descent (SGD), Alternating Least Squares (ALS) and Markov Chain Monte Carlo (MCMC).

IV. Gradient Boosting Tree Based Models

Tree ensemble methods are very widely used in practice. Gradient tree boosting is one of them that shines in many applications. The classic gradient boosting tree and its extension are described in [52]. XGBoost [53] is a scalable open source system for tree boosting. The impact of the XGBoost has been widely recognized in a number of machine learning and data mining challenges. One who uses the gradient boosting trees, often chooses XGBoost as the implementation of the Gradient Boosting Regression Trees (GBRT) in the application.

A tree ensemble model uses K additive functions to predict the output.

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), f_k \in \mathcal{F}, \tag{17}$$

where $\mathcal{F}$ is the space of regression trees (also known as CART). The regularized objective function is listed as follows:

$$\mathcal{L} = \sum_i l(y_i, \hat{y}_i) + \sum_k \Omega(f_k), \tag{18}$$

where l is a loss function that measures the difference between the prediction $\hat{y}_i$ and the target y_i. The second term Ω penalizes the complexity of the model:

$$\Omega(f_k) = \gamma T + \frac{1}{2}\lambda \|\omega\|^2. \tag{19}$$

T is the number of leaves in the tree. Each regression tree contains a continuous score on each leaf, ω_i is the score on i-th leaf.

Since the tree ensemble model in Eq.(18) includes functions as parameters but not just numerical vectors, it cannot be optimized using traditional optimization methods such as stochastic gradient descent (SGD) in Euclidean space. In XGBoost, Eq.(18) is trained in an additive manner.

$$\hat{y}_i^{(t)} = \sum_k f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i), \tag{20}$$

where $\hat{y}_i^{(t)}$ is the prediction of the i-th instance at the t-th iteration. Then, the objective function is:

$$\mathcal{L} = \sum_i l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \sum_k \Omega(f_k). \tag{21}$$

Consider square loss and take Taylor expansion approximation of the loss, we get:

$$\mathcal{L}^{(t)} \simeq \sum_i [l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_k) + constant, \tag{22}$$

where

$$g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}), \tag{23}$$

and

$$h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}). \tag{24}$$

Combining Eq.(18) and Eq.(22), we remove constants and get:

$$\mathcal{L}^{(t)} \simeq \sum_i [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \gamma T + \frac{1}{2}\lambda \sum_j \omega_j^2, \tag{25}$$

This is One Variable Quadratic Equation of ω_j. We can compute the optimal weight ω_j^* of leaf j by

$$\omega_j^* = -\frac{\sum_i g_i}{\sum_i h_i + \lambda}, \tag{26}$$

and calculate the corresponding optimal objective function value by

$$\tilde{\mathcal{L}}^{(t)} = -\frac{1}{2} \sum_j \frac{(\sum_i g_i)^2}{\sum_i h_i + \lambda} + \lambda T, \tag{27}$$

In practice, the greedy algorithm, that starts from a single leaf and iteratively adds branches to the tree, is usually used for evaluating the split candidates. It is impossible to efficiently do the exact greedy algorithm when the data does not fit entirely into memory. And then, the approximate algorithm for split finding is proposed in XGBoost. More details can be found in [54].

V. Deep Learning Based Models

Recently, deep learning models have been widely exploited in various matching tasks with remarkable performance. Applying deep learning models into recommender system has been gaining momentum due to its state-of-the-art performances on popular benchmarks for recommender systems, such as MovieLens[4] and Netflix challenge datasets. Among those deep learning based recommender systems, an autoencoder based system "AutoRec" and a neural autoregressive based system "CF-NADE" have been utilized in the *Problem 1*. Moreover, a semantic matching model named

[4]https://grouplens.org/datasets/movielens/1m/.

Match-SRNN, which can model the recursive matching structure between experts and questions, has been also used before.

A. Autoencoder Model

AutoRec [55] is an autoencoder based collaborative filtering model. Similar to traditional CF, AutoRec has two variants: an user-based autoencoder and an item-based autoencoder. They can respectively take user partial vectors and item partial vectors as input, project them into a hidden layer to learn the lower-dimensional representations, and further reconstruct them in the output layer to predict missing ratings for the purpose of recommendation.

While AutoRec is used in the *Problem 1*, experts are regarded as users, questions as items, and the question distribution data as rating matrix. The question distribution data consists of question push notification records that indicate whether the expert answered the question (if answered, the tag is 1; otherwise 0). Then the AutoRec model is deployed to predict the ratings of unknown expert-question pairs.

Both user-based AutoRec and an item-based AutoRec are exploited in expert finding in CQA. Experiment results show that item-based model performs better which may be due to the higher variance of user partial vectors. However, item-based AutoRec is not performing well than MF-based models as before. The reason may be that the dataset of Toutiao is more sparse than the MovieLens dataset.

B. Neural Autoregressive Model

Inspired by the Restricted Boltzmann Machine (RBM) based CF model, an emerging Neural Autoregressive Distribution Estimator (NADE) based CF model named CF-NADE [56] is proposed. It can model the distribution of expert ratings. CF-NADE with only one hidden layer can defeat all the previous state-of-the-art models on recommendation tasks upon the MovieLens 1M, MovieLens 10M and Netflix datasets. Furthermore, CF-NADE can be further extended to a deep model with more hidden layers which can further boost the performance.

CF-NADE, which is designed to model the ordering of the ratings, is a feed-forward and neural autoregressive architecture for CF tasks. Ideally, the order of items should follow the time-stamps of ratings. However, empirical study shows that random drawing permutations for each user also generates favourable performances. Since the expert IDs as well as the question IDs are anonymized and the descriptions of expert and questions in the dataset have been encoded into ID sequences, it is feasible to deploy CF-NADE to this competition without time-stamps information. While training the CF-NADE model, the experts and questions are considered as users and items, and the rating matrix is derived from question push notification records like in Section V-A. Experiment results show that the performance of CF-NADE model in the *Problem 1* is similar to the AutoRec model, in which item-based CF-NADE performs better than user-based CF-NADE but still not comparable to the matrix factorization based models such as SVD++ and ASVD++. Moreover, the CF-NADE model, though worth trying, is not integrated into any final ensemble models because it significantly reduces the performance when incorporated into ensemble models.

C. Match-SRNN

Furthermore, the expert finding problem in CQA can also treated as a text matching problem. Thus, text matching methods can be applied to this task which can take advantage of textual features such as characters and words in the the expert and question descriptions. For the *Problem 1*, a deep text matching model called Match-SRNN [57] is applied to model the interaction information between texts to further predict new expert-question pairs. The Match-SRNN model contains three parts: a neural tensor network to capture the character/word level interactions, a spatial recurrent neural network (spatial RNN) applied on the character/word interaction tensor to capture the global interactions recursively, and a linear scoring function to calculate the final matching score. The Match-SRNN model views the generation of the global interaction between two texts as a recursive process which can not only obtain the interactions between nearby words, but also take advantage of long distant interactions.

VI. Ranking Based Models

The evaluation criterion in this task is normalized discounted cumulative gain (NDCG), thus ranking based model is a natural fit for this target. There are two kinds of ranking based models appearing in the expert finding problem in CQA, including ranking based FM and ranking based SVM.

A. Ranking based FM

The basic idea of this model is coming from the FM method. We modify the objective function to optimize the pair-wise ranking loss. Let N^+ denotes the number of positive samples and N^- denotes the number of negative samples. Besides, x_i denotes the negative instances and x_j denotes positive instances. Then we solve the following optimization problem for ranking based FM.

$$\min_{W,V} \frac{1}{N^+ + N^-} \sum_{i=1}^{N^-} \sum_{j=1}^{N^+} log(1 + exp(\hat{y}(x_i) - \hat{y}(x_j))) + \frac{\lambda}{2} \|\theta\|^2 \tag{28}$$

where $\hat{y}(x)$ is the prediction in the Eq. (15). We expect that those positive samples have higher prediction score than those negative samples.

B. Ranking based SVM

ranksvm [58], which is a linear pairwise ranking model, has also been used in the problem. Specifically, we first build the feature vectors for each user-question pair appeared in the training/test sets. Then those training pairs with same questions are organized together as a list. The pairwise constraints are then built within each list.

VII. Ensemble Learning

During the review of the ensemble learning solutions, we find that many contestants are obscure about the concept of ensemble learning, especially Stacking. These proper nouns are often inappropriately used in ensemble learning. Here, we comb through the relevant concepts of ensemble learning that are widely used in practice. In machine learning, ensemble learning (also called ensemble method [59] before) is a proper noun. It is a method of using multiple learning algorithms to obtain better predictive performance than that could be obtained by any of the component learning algorithms alone. Ensemble learning can be used for classification problems, regression problems, feature selection, anomaly detection and so on. In the following part, we will use classification as an example.

If we use ensemble learning to improve the overall generalization ability of classifiers, the following two conditions should be satisfied. Firstly, differences exist between the base classifiers. The performance of the ensemble classifier will not be improved, if it is just an ensemble of the same kind of base classifiers. Secondly, the classification accuracy of every base classifier must be larger than 0.5. If the classification accuracy of the base classifier is less than 0.5, the classification accuracy of the ensemble classifier will decline with the increasing of ensemble size. If the two aforementioned conditions are satisfied, the classification accuracy of the ensemble classifier will edge up to 1 with the increasing of ensemble size. Generally, the classification accuracy of a weak classifier is just slightly better than random guess, while a strong classifier can make make very accurate predictions. The base classifiers are referred to as weak classifier.

There are two key points in ensemble learning. How to generate base classifiers with difference? How to combine the results of the base classifiers? We will introduce ensemble learning from these two aspects.

A. Types of Ensemble Learning

According to how the base classifiers are constructed, there are two paradigms of ensemble learning, the parallel ensemble learning and the sequential ensemble learning. In the parallel ensemble learning, the base classifiers are generated in parallel, with Bagging [60] as a representative. In the sequential ensemble learning, the base classifiers are generated sequentially, with Boosting [61] as a representative.

*1) **Bagging**:* Bagging (**B**ootstrap **agg**regat**ing**) was proposed to improve classification accuracy by combining classifiers of randomly generated training sets. Fig. 2(a) illustrates the diagram of Bagging.

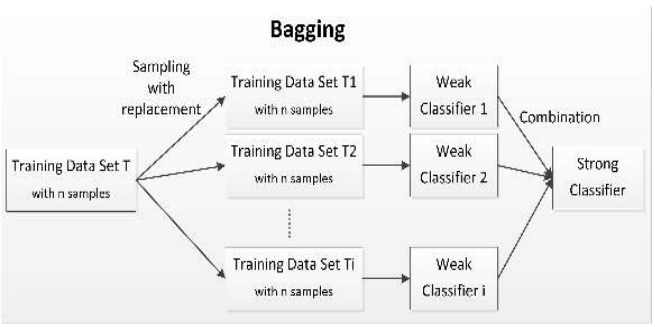
Bagging applies bootstraping [62] to obtain the data subsets for training the base classifiers. In detail, given a training data set containing n training examples, a sample of n training examples will be generated by random sampling with replacement. Some original examples appear more than once, while some original examples are not present in the sample. If we need to train m number of base classifiers, this process will be applied m times. The combination methods used by Bagging are the most popular strategies, that is, voting for classification and averaging for regression. Here, the final classification results are determined by averaging on the respective results of these classifiers.

*2) **Boosting**:* Instead of resampling the training dataset as Bagging does, Boosting adjusts the distribution of the training dataset. Fig. 2(b) illustrates the diagram of Boosting. Boosting is an iterative process to generate base classifiers sequentially, where the later classifiers focus more on the mistakes of the earlier classifiers. In each round, the weight of the samples, which have been classified incorrectly, will be increased in the training dataset. The weight of the samples, which have been classified correctly, will be decreased in the training dataset. Finally, the ensemble classifier is a weighted combination of these weak classifiers.

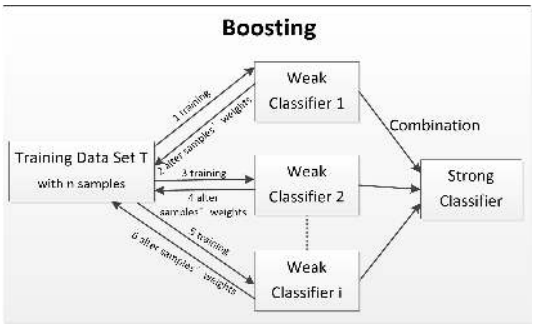
B. Combination Methods

The combination method plays a crucial role in ensemble learning. After generating a set of base classifiers, ensemble learning resorts to combination method to achieve an ensemble classifier with strong generalization ability, rather than trying to find a best single classifier. Generally, the most popular combination methods used in practice are **Voting**, **Averaging** and **Learning**. Voting and Averaging are the most popular and fundamental combination methods for nominal outputs and numeric outputs, respectively. These two methods are easy to understand and use. Here, we mainly focus on the Learning, with Stacking (stacked generalization) as a representative.

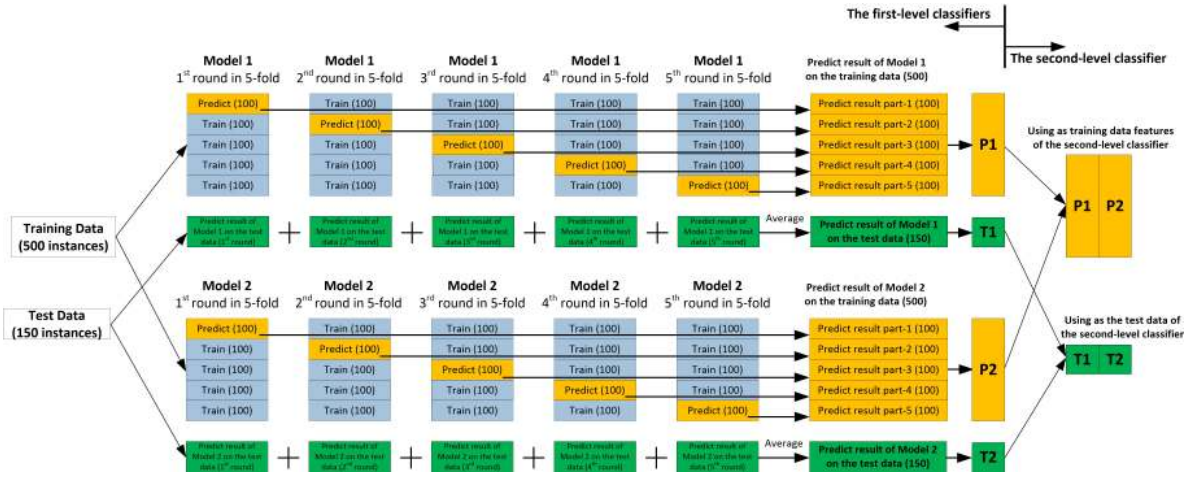
1) Stacking: Unlike Voting and Averaging, Stacking is a general combining procedure where the base classifiers are combined non-linearly in a serial model. In Stacking, the base classifiers are called the first-level classifiers, while the combiner is called the second-level classifier (or meta-classifier). The basic idea of Stacking



(a) Diagram of Bagging.



(b) Diagram of Boosting.



(c) Diagram of Stacking.

Fig. 2. Diagram of Ensemble Learning.

is to train several first-level classifiers using the original training dataset. And then, a new dataset generated from the first-level classifier is used to train the second-level classifier, where the outputs of the first-level classifiers are regarded as the input features of the new training dataset, and the original labels are still the labels of the new training data.

In the training phase of Stacking, if all the instances in the training dataset are used to train the first-level classifiers, and the outputs of the first-level classifiers are used to train the second-level classifier, there will be a high risk of over-fitting. Therefore, the instances used for generating the input of the meta-classifier need to be excluded from the training instances of the first-level classifiers. Generally, a cross validation is used to avoid this problem.

Taking a Stacking model with 2 first-level classifiers and 5-fold cross validation as an example, Fig. 2(c) illustrates the diagram of Stacking. There are 500 instances in the training dataset. Using the Model 1 (the first-level classifier) in Fig. 2(c) as an example, in the 5-fold cross validation, the training dataset is divided into 5 parts, and each part has 100 instances. Four of them (with 400 instances in total) are used to train the Model 1. The remaining one part (with 100 instances) is used to do prediction. The prediction results (5 parts with 500 instances in total) are used as the features of the input of the second-level classifier. In every round in the 5-fold cross validation, the trained Model 1 makes prediction on the test dataset (with 150 instances). After 5 rounds, there are 5 parts of the prediction results on the test dataset. Making an average of these 5 parts, there are still 150 instances in the final prediction result of Model 1 on the test dataset.

Generally, Stacking can be viewed as a specific combination method of the Learning combination strategy. What's more, it can also be regarded as a general framework of many ensemble methods used in practice.

VIII. Results

In terms of the evaluation criteria, NDCG will be used. Specifically, we will rank the experts based on the forecasted probability for a certain question, and evaluate the $NDCG@5$ and $NDCG@10$ of ranking results. The final evaluation formula is: $NDCG@5 * 0.5 + NDCG@10 * 0.5$.

TABLE II
DESIGNED FEATURES.

Name	Notation	Description	Type	+/-
Anonymized expert user ID	*uID*	The unique identifier of each expert user.	id	+
Expert user tags	*uTag*	The tag of user information.	category	+
Word ID sequence of user	*uwordIDseq*	Segmented user description. Each word is replaced by a unique wordID.	category	-
Character ID sequence of user	*ucharIDseq*	Segmented user description. Each character is replaced by a unique charID.	category	-
Anonymized question ID	*qID*	The unique identifier of each question.	id	+
Question tag	*qTag*	The tag of each question.	category	+
Word ID sequence of question	*qwordIDseq*	Same as uwordIDseq instead of question description.	category	-
Character ID sequence of question	*qcharIDseq*	Same as ucharIDseq instead of question description.	category	-
Number of upvotes	*upvoteNum*	Number of upvotes of all answers to this question.	numeric	+
Number of answers	*ansNum*	Number of all answers to this question.	numeric	+
Number of top quality answers	*topAnsNum*	Number of top quality answers to this question.	numeric	+
Implicit expert	*imE*	Expert list with implicit relationship.	category	++
Implicit question	*imQ*	Question list with implicit relationship.	category	++

A. Data Analysis

In this paper, we analyze the problem of expert finding in CQA by taking the data of ByteCup competition as an example. The data provided for the competitors consisting of expert finding records in CQA with three types of information: expert tags, question data and question distribution data:

1) The expert tag data, which contains IDs of all expert users, their interest tags, and processed profile descriptions.
2) The question data, which contains IDs of all questions, processed question descriptions, question categories, total number of answers, total number of top quality answers, total number of upvotes.
3) The question distribution data: 290000 records of question push notification, each contains the encrypted ID of the question, the encrypted ID of the expert user and if the expert user answered the question (0=ignored, 1=answered).

The training set, validation set and test set are divided based on these records. The training set is used for the training of the model. Validation set is used for online real-time evaluation of the algorithm. Test set is used for the final evaluation.

All expert ID and question ID are encrypted to protect user privacy. Also for privacy protection purpose, the original descriptions of the questions and the experts are not provided. Instead, the ID sequence of the characters (each Chinese character will be assigned an ID) and the ID sequence of the words after segmentation (each word will be assigned an ID) are provided. Validation and testing labels have not been published. They are used for online evaluation and final evaluation only.

B. Feature Extraction

We summarise all possible features in Table II. The expert user tags *uTag* may be multiple tags, i.e., 18, 19 and 20 may represent baby, pregnancy and parenting, respectively. In the feature of *uwordIDseq*, user descriptions (excluding modal particles and punctuation) are first segmented, and then each word will be replaced by the Character ID, i.e., 284/42 may represent "Don't Panic". In the feature of *ucharIDseq*, user descriptions (excluding modal particles and punctuation) are first segmented, and then each character will be replaced by the Character ID, i.e., 284/42 may represent "BE". The question tag *qTag* may be a list of single tags, i.e., 2 may represent fitness. The feature *upvoteNum*, *ansNum* and *topAnsNum* may indicate the popularity of the question.

We also study the positive/negative contributions of each feature. As Table II illustrated, four features, including *uwordIDseq*, *ucharIDseq*, *qwordIDseq* and *qcharIDseq*, have negative impact on the model performance. The implicit features *imE* and *imQ*, which have strong positive influence on the model performanceare needed to be considered in the prediction model.

Table. III illustrates the features used by the top 5 teams in the competition ByteCup. The four features including *uwordIDseq*, *ucharIDseq*, *qwordIDseq* and *qcharIDseq*, that have negative impact on the model performance shown in Sec. VIII-B, have not been used by any team. Therefore, we does't include them in Table. III. Although there are nine positive features, simply combining all of them will not lead to the best performance. All top 5 teams use the four features, including *uID*, *qID*, *imE* and *imQ*. The latent features *imE* and *imQ* underlying the interactions between different kinds of entities have important influence on the performance.

C. Results of Single Models

SVDFeature [63] and Factorization Machine(libFM) [51] tools are used for MF-based models. XGBoost [54] is used for GBT-based models. The code based on Theano framework is used for the DL-based models.

TABLE III
DESIGNED FEATURES.

Features / Team	*uID*	*uTag*	*qID*	*qTag*	*upvoteNum*	*ansNum*	*topAnsNum*	*imE*	*imQ*
Team-1	●	●	●	●	○	○	○	●	●
Team-2	●	○	●	○	○	○	○	●	●
Team-3	●	○	●	○	○	○	○	●	●
Team-4	●	○	●	○	●	●	●	●	●
Team-5	●	○	●	○	○	○	○	●	●

● means that the feature is used. ○ means that the feature is not used.

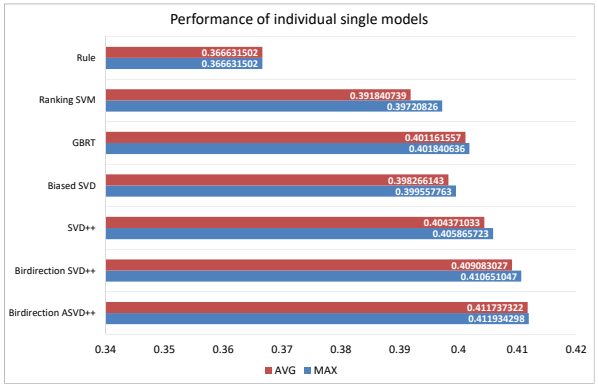


Fig. 3. Individual model performances on local validation dataset.

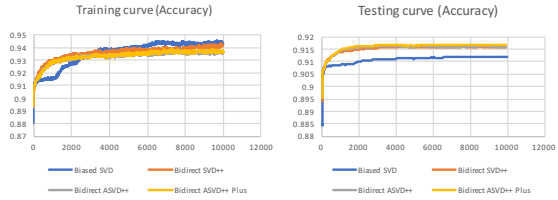


Fig. 4. MF-based models training/testing curve.

The results of all aforementioned categories of single models on the local validation dataset is illustrated in Figure. 3. From the figure we can see that, some single models such as ASVD and bidirectional SVD++ make good performances. However, there are also weak models such as ranksvm and simple heuristic based method. In general, the MF-based models perform better than others including GBT-based models and DL-based models, which performs well in many other kinds of applications. We used different settings of parameters (max depth of each tree, number of trees, and boosting step size) to train several XGBoost models. Based on the experiments on local validation dataset, the performance of these models (refer to the performance of models starting with "GBRT" in Fig. 3) are reasonable, but not as good as MF-based models. Nevertheless, they do improve the performance of the final ensemble model. These models have quite different objective and underlying assumptions than MF-based methods. Therefore, a decent weak model will still improve the final ensemble results.

In the MF-based models, the bidirection ASVD++ performs the best. What's more, if more implicit information is used, such as ratting action in online validation dataset or online test dataset, the model performance could be further improved. This phenomenon is reflected in Fig. 4. The accuracy of the bidirect ASVD++ is highest, followed by the bidirect ASVD++, the bidirect SVD++ and the bidirect SVD in the descending order.

Table. IV illustrates the parameters for the bidirection ASVD++ that achieves the best performance. Markov Chain Monte Carlo (MCMC) is used for the learning method in the model. Table. V illustrates the best performance of the bidirection ASVD++ on the local validation dataset, the online validation dataset and the online test dataset. The results are 0.41193, 0.52412 and 0.50551, respectively.

TABLE IV
PARAMETERS FOR THE BIDIRECTION ASVD++.

Parameters	Value
Learning method	MCMC
#Factor	8
#Iteration	10000
Task	binary classification
Stdev for init. of 2-way factors	0.1

TABLE V
PERFORMANCE OF BIDIRECTION ASVD++.

Test Set	Performance (nDCG)
Local Validation	0.41193
Online Validation	0.52412
Online Test	0.50551*

* Already rank first among all single models.

D. Results of Ensemble Models

Taking the ensemble models of the Top 5 teams who won the competition ByteCup as the example, we analysis the results of the ensemble models.

1) Team-1: As shown in Table. VI, Team-1 combines 45 models linearly with different settings (features, tools or hyper-parameters) using the linear ridge regression.

Specifically, they do 5-fold cross validation on the local validation set. The final ensemble model is trained using local validation set. Note that, the predictions of local validation set are from those models trained on local training set. Thus the training set are not involved in the ensemble step. They also ensemble the predictions from same model with different parameters, such as different latent dimensions or different objective functions of matrix factorization models. The small variations make the single model more robust. To avoid the bias due to different scales, they do whitening for each model's prediction before ensemble.

Team-1 takes the predictions of each candidate model, and does a linear combination of those predicted values to make the final prediction. The score of these candidate models range from 0.367 to 0.412, they tune the weights of them based on the rating prediction on local validation set. The prediction ensemble of a set of base models further improves the performance. Finally, they get the score of 0.50812 on the final leaderboard. Team-1 has also tried to use nonlinear ensemble method, such as the gradient boosting tree, to do the ensemble. However, they found such tree models are very easy to over-fit the training set. It is also hard to regularize the model to get a good test performance.

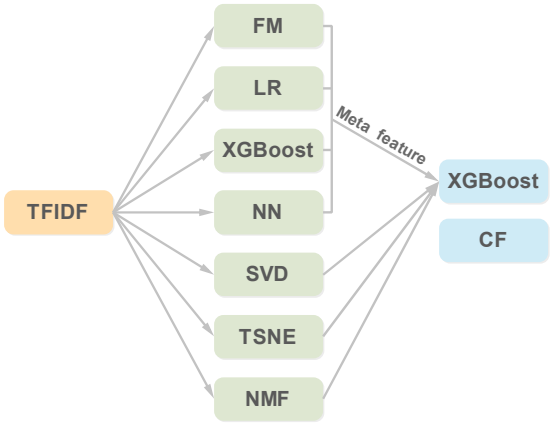


Fig. 5. Diagram of Stacking Used by Team 2

2) Team-2: For every expert, there is a list of questions that have been answered. Here, Team-2 regards the expert-question list as a document, and each question is a term. The TF-IDF of each question is calculated and used as the feature *imQ*. Similarly, The TF-IDF of each expert is calculated and used as the feature *imE*.

Team-2 uses the method of Stacking to integrate several single models. The Stacking strategy used by them is illustrated in Fig. 5. In the Stacking, FM, Logistic Regression (LR), XGBoost and Neural Network (NN) are the first-level classifiers. The results of them are used as inputs of the next layer, called meta features. SVD, TSNE [64], NMF [65] is used to get the dimension reduction features of the original features. Finally, the meta features and the dimension reduction features are combined to train the XGBoost.

The used NN has one hidden layer, in which the activation function is ReLu(Rectified Linear Units), the droupout rate is 0.75. Adam [66] is also used here to optimize the model. XGBoost is trained in the following steps. They uses the social graph to model the relationship between experts and questions $< E, Q >$. The experts and questions are regarded as nodes in an undirected graph. If a expert is invited to answer a question, there will be an undirected edge between them. DeepWalk [67] is used to convert $< E, Q >$ to work vector, which then be used to train XGBoost.

In addition, they find three implied messages of CF based on the observation and analysis of the issues and data.

- If a expert has accepted most of the invitation for answering question, he will be more likely to accept the new invitation to answer question.
- Experts have answered some same questions. If some of them (assume the number is N) answer a new question, others may also answer the question (assume the probability is p). N is larger, p is larger.
- If questions $Q1$ and $Q2$ are given to the same user, $Q1$ and $Q2$ may be involved in the same field. If $Q1$ is answered by an expert, $Q2$ may be answered by the expert too.

And then, they combine the results of Stacking and CF by weight $2 : 1$. Finally, they get the score of 0.50307 on the final leaderboard. Only 1% less than Team-1.

3) Team-3: The weight of the question that is related to the expert *uid* is regarded as the feature *imQ* by Team-3. It is calculated as the reciprocal of the question numbers answered by the expert *uid*. The weight of the expert that is related to the question *qid* is regarded as the feature *imE*. It is calculated as the reciprocal of the expert numbers who answer the question *qid*. FM is achieve by libFM.

In CF, the probability of expert answering question is calculated as the weighted sum of the average similarity between experts and the average similarity between questions. The similarity between questions is calculated as the weighted difference between the positive similarity of the question and the negative similarity of the question. The positive similarity of question is the number of experts who have similar behavior on the specific question and answer the test question. The negative similarity of question is the number of experts who have similar behavior on the specific question and not answer the test question. The similarity between experts is calculated similarly as the similarity between questions.

TABLE VI
ENSEMBLE MODELS USED BY THE TOP 5 TEAMS.

Method / Team	Details of the ensemble model	Final results	Compare with Team-1
Team-1	Linearly combine all models in Fig. 3	0.50812	0
Team-2	Use Stacking strategy illustrated as Fig. 5	0.50307	-1%
Team-3	FM+CF *	0.49905	-1.82%
Team-4	MF+CF	0.49231	-3.21%
Team-5	FM+RFM+(FM+RFM)+MF+SVD+(SVD++)	0.49003	-3.69%

* FM+CF represents the linear weighted sum of FM and CF.

As shown in Table. VI, Team-3 combines the results of FM and CF with the linear weighted sum. Finally, they get the score of 0.49905 on the final leaderboard. 1.82% less than Team-1.

4) Team-4: As shown in Table. VI, Team-4 combines the results of MF and CF with the linear weighted sum. In the scheme of CF, the prediction is calculated as the formula shown below:

$$pred(u,i) = \bar{r}_u + \frac{\sum_{v \in N(u)} sim(u,i) * (r_{v,i} - \bar{r}_v)}{\sum_{v \in N(u)} sim(u,i)}, \tag{29}$$

where $sim(u,i)$ is calculated by

$$sim(u,i) = \frac{\sum_i (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_i (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_i (r_{v,i} - \bar{r}_v)^2}}. \tag{30}$$

$N(u)$ is the set of neighbors of the specific expert u. The number n of $N(u)$ is hyper parameter needed to be tune. They use $n = 5000$ in the final model.

Finally, they get the score of 0.49231 on the final leaderboard. 3.21% less than Team-1.

5) Team-5: Team-5 combines the results of 6 individual models on the validation set, including FM, ranking based FM (RFM), the linear weighted sum of FM and RFM, three MF-based models (MF, SVD and SVD++). Assuming the predictions of the user-question pairs from the 6 individual models are $pred_1, pred_2, pred_3, pred_4, pred_5, pred_6$, respectively. A weight is assigned to every individual model and the final prediction of the user-question pairs is computed by the following formula:

$$\begin{aligned} pred = {} & \alpha_1 pred_1 + \alpha_2 pred_2 + \alpha_3 pred_3 \\ & + \alpha_4 pred_4 + \alpha_5 pred_5 + \alpha_6 pred_6 \end{aligned} \tag{31}$$

After the ensemble, the performance of the model turns out to be better.

What's more, Team-5 finds a rule in the training set, and it can be used in the validation set to improve the model performance. In the training set, a certain user-question pair only appears once or twice and a user answers the question once at most. Therefore, they assume that expert won't answer the same question twice and it is consistent with the reality. When the user-question pair appears in the validation set and it also appears in the training set where the user answers the question, they predict that user won't answer the question again. This rule helps to boost the performance on the validation set again.

Finally, they get the score of 0.49003 on the final leaderboard. 3.69% less than Team-1.

IX. DIVERSE MODELS ON DIFFERENT TYPES OF MATCHING TASKS

In this section, we compare the performance of diverse models on different types of matching tasks to explore the difference among the models on different matching tasks (Figure. 6). Totally, seven matching tasks were involved in the study including:

1) Toutiao: The evaluation metric of ByteCup is $NDCG@5 * 0.5 + NDCG@10 * 0.5$;
2) Movielens: Movie recommendation on MovieLens data with evaluation metric $NDCG@10$;
3) Sohu Contest: Sohu Programming Contest[5] on news pictures data with evaluation metric average NDCG;
4) Lung Cancer: Data Science Bowl 2017[6] on Lung CT images data with evaluation metric LogLoss;
5) MLSP bird: MLSP 2013 Bird Classification Challenge[7] on bird sounds audio data with evaluation metric micro-AUC;
6) YouTube: Google Cloud & YouTube-8M Video Understanding Challenge[8] on YouTube videos data with evaluation metric Global Average Precision@20;
7) MSR-video2text: Video to Language Challenge[9] on MSR-video2text data with evaluation metric $BLEU@4$.

Based on the data type of the tasks, we classified the seven tasks into 4 categories. There are : 1) *text vs. text*, which means to match text labels with text data, includes ByteCup and Movie recommendation; 2) *graph vs. text*, which means to match text labels with graph data, contains Sohu Programming Contest and Data Science Bowl 2017; 3) *audio vs. text*, which aims to match text labels with audio data, includes MLSP 2013

[5]https://biendata.com/competition/luckydata/.
[6]https://www.kaggle.com/c/data-science-bowl-2017.
[7]https://www.kaggle.com/c/mlsp-2013-birds.
[8]https://www.kaggle.com/c/youtube8m.
[9]http://ms-multimedia-challenge.com/2016/challenge.

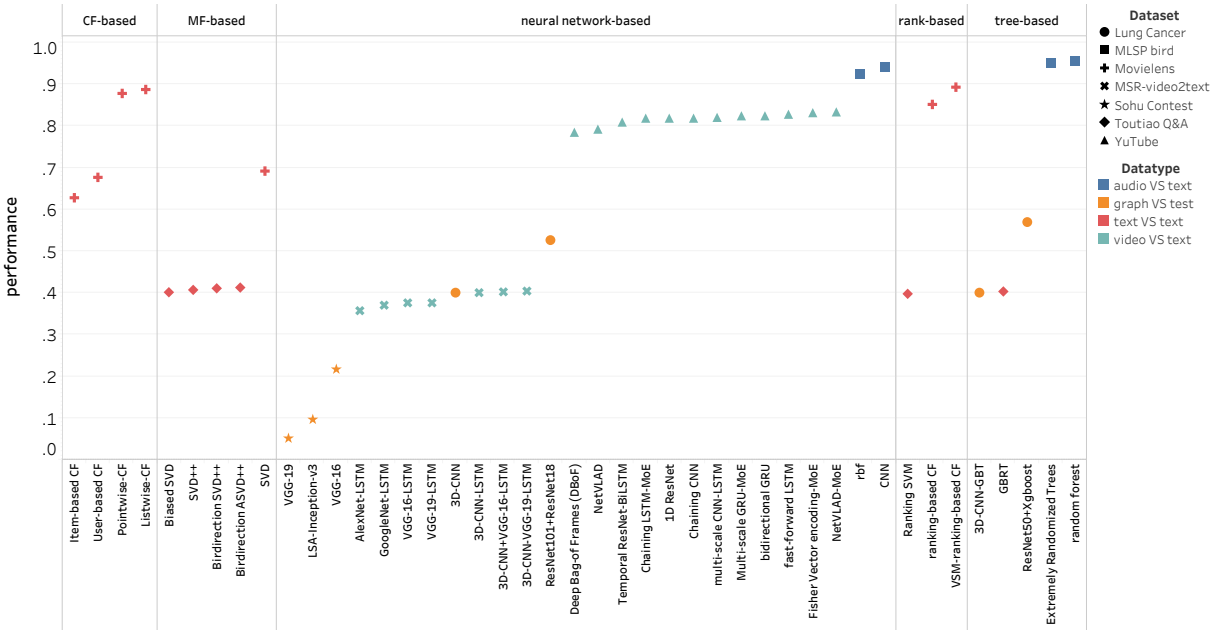


Fig. 6. Performances of diverse models on different type of datasets.

Bird Classification Challenge; 4) *video vs. text*, which is to match text labels with video data, includes Google Cloud & YouTube-8M Video Understanding Challenge and Video to Language Challenge.

The models used in the seven tasks are also classified into four categories including MF-based models, GBT-based models, R-based models and DL-based models. As shown in Figure. 6, MF-based models and rank-based models are used only in *text vs. text* category of matching tasks, while DL-based models are not employed in these tasks since they are not performing well (which may due to the severe sparsity of the datasets). MF-based models usually achieve the best performance in *text vs. text* category of matching tasks. In addition, DL-based models achieve the best performance in *graph vs. text* and *video vs. text* categories, which may due to their outstanding power of capturing high dimensional features from graph and video, and they are also utilized in the *audio vs. text* category. Finally, GBT-based models have significant performance on *audio vs. text* category.

X. Discussion

In this article, we statistically analyze all the existing solutions for the expert finding problem in CQA. We summarise the results analysis and the learned lessons learned in this part.

A. Results Analysis

We describe the different individual methods used in the task, and also introduce several types of ensemble learning. And then, we present the results of both of them. It is worth noting that the different individual methods get scores from 0.3665 to 0.4119 when used independently. The results of ensemble learning range from a score of 0.49003 to a score of 0.50812. Since the data used in the task is the real data from Toutiao with about 580 million users, even minor improvements can affect millions of users.

Based on the analysis of the solutions and the observation of the results, we find that the ensemble methods outperform any of the single models when they were used independently. That is, ensemble learning really outperforms every single component model, if the two conditions mentioned in Sec. VII are both satisfied. Although there are some model with poor performance, the use of them with other different kind of models leads to a considerable improvement of the prediction. YES! A weak model in combination with other different kind of models can still improve the performance of the final ensemble model. In general, the combination of different kinds of models even with a weak model[10] leads to significant performance improvements over every single component model.

B. Important Lessons

As known from the No Free Lunch (NFL) Theorem, none of the algorithms is better than a random one. In the field of machine learning, there isn't an almighty algorithm that is applicable to all situations. Different data sets and different problems have different best algorithms respectively. In previous years, XGBoost

[10]Its accuracy is larger than 0.5.

shows its absolute advantage in the structured data. However, it puts up a poor show than MF-based models in this task. It is a reasonable explanation that the dataset here is more sparse than movie rating datasets used in previous tasks.

As noticed, a single model won't win. This shows that, as expected, the field of machine learning is getting stronger. This paper witnesses the advantage of ensemble learning applied to the combination of different learning models. In addition, many mobile social platforms in China, such as WeChat, Sina Weibo, Toutiao and so on, have hundreds of million users. Even minor improvements of the solution results can affect millions of users.

Moreover, from the survey of the performance of different models on different types of matching types, we learned that MF-based models and rank-based models are more suitable for *text vs. text* matching tasks, DL-based models and GBT-based models achieve the best results for *audio vs. text* matching tasks. DL-based models are appropriate for both *video vs. text* and *audio vs. text* matching tasks.

XI. Conclusion

This survey paper focuses on the expert finding problem in CQA. Given certain question, one needs to find who are the most likely to 1) have the expertise to answer the question and 2) have the willingness to accept the invitation of answering the question. We have reviewed most existing solutions and classify them to four different categories: MF-based models, GBT-based models, DL-based models and R-based models. Experimental results demonstrate the effectiveness and efficiency of the MF-based models in the expert finding problem in CQA.

In the future, several important research issues need to be addressed. First, how to efficiently integrate the implicit feedback is an open problem. Obviously, implicit feedback becomes increasingly important in practical application, because users provide much more implicit feedback than explicit one. In addition, explainability is usually ignored in the research. The existing methods face real difficulties to explain predictions. Finally, how to make sure that the established model is no needed to be re-trained is a crucial issue in expert finding in CQA. We hope that the overview presented in this paper will advance the discussion in the expert finding technologies in CQA.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (61561130160), and the National High Technology Research and Development Program of China (863 Program) ($2015AA124102$).

References

[1] F. Riahi, Z. Zolaktaf, M. Shafiei, and E. Milios, "Finding expert users in community question answering," *Topic Models Expert Recommender*, pp. 791–798, 2012.

[2] Z. Zhao, Q. Yang, D. Cai, X. He, and Y. Zhuang, "Expert finding for community-based question answering via ranking metric network learning," in *International Joint Conference on Artificial Intelligence*, 2016, pp. 3000–3006.

[3] F. Han, S. Tan, H. Sun, M. Srivatsa, D. Cai, and X. Yan, "Distributed representations of expertise," in *Siam International Conference on Data Mining*, 2016, pp. 531–539.

[4] K. Balog, Y. Fang, M. De Rijke, P. Serdyukov, and L. Si, "Expertise retrieval," *Foundations and Trends in Information Retrieval*, vol. 6, no. 23, pp. 127–256, 2012.

[5] X. Liu, M. Koll, and M. Koll, "Finding experts in community-based question-answering services," in *ACM International Conference on Information and Knowledge Management*, 2005, pp. 315–316.

[6] S. Lin, W. Hong, D. Wang, and T. Li, "A survey on expert finding techniques," *Journal of Intelligent Information Systems*, pp. 1–25, 2017.

[7] J. Zhang, J. Tang, and J. Li, *Expert Finding in a Social Network*. Springer Berlin Heidelberg, 2007.

[8] H. Rode, P. Serdyukov, D. Hiemstra, and H. Zaragoza, "Entity ranking on graphs: Studies on expert finding," *Centre for Telematics and Information Technology University of Twente*, 2017.

[9] M. Rafiei and A. A. Kardan, "A novel method for expert finding in online communities based on concept map and pagerank," *Human-centric Computing and Information Sciences*, vol. 5, no. 1, pp. 1–18, 2015.

[10] V. Boeva, M. Angelova, and E. Tsiporkova, "Data-driven techniques for expert finding," in *International Conference on Agents and Artificial Intelligence*, 2017, pp. 535–542.

[11] G. A. Wang, J. Jiao, A. S. Abrahams, W. Fan, and Z. Zhang, "Expertrank: A topic-aware expert finding algorithm for online knowledge communities," *Decision Support Systems*, vol. 54, no. 3, pp. 1442–1451, 2013.

[12] A. Dargahi Nobari, S. Sotudeh Gharebagh, and M. Neshati, "Skill translation models in expert finding," in *International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2017, pp. 1057–1060.

[13] Y. Li, S. Ma, and R. Huang, "Social context analysis for topic-specific expert finding in online learning communities," 2015.

[14] G. Zhou, S. Lai, K. Liu, and J. Zhao, "Topic-sensitive probabilistic model for expert finding in question answer communities," in *ACM International Conference on Information and Knowledge Management*, 2012, pp. 1662–1666.

[15] D. R. Liu, Y. H. Chen, W. C. Kao, and H. W. Wang, "Integrating expert profile, reputation and link analysis for expert finding in question-answering websites," *Information Processing and Management An International Journal*, vol. 49, no. 1, pp. 312–329, 2013.

[16] R. Yeniterzi and J. Callan, "Constructing effective and efficient topic-specific authority networks for expert finding in social media," pp. 45–50, 2014.

[17] H. Zhu, E. Chen, H. Xiong, H. Cao, and J. Tian, "Ranking user authority with relevant knowledge categories for expert finding," *World Wide Web-internet and Web Information Systems*, vol. 17, no. 5, pp. 1081–1107, 2014.

[18] M. Bouguessa and S. Wang, "Identifying authoritative actors in question-answering forums: the case of yahoo! answers," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008, pp. 866–874.

[19] J. Liu, Y. I. Song, and C. Y. Lin, "Competition-based user expertise score estimation." in *Proceeding of the International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2011, Beijing, China, July*, 2011, pp. 425–434.

[20] G. Zhou, J. Zhao, T. He, and W. Wu, "An empirical study of topic-sensitive probabilistic model for expert finding in question answer communities," *Knowledge-Based Systems*, vol. 66, no. 9, pp. 136–145, 2014.

[21] J. Zhang, J. Tang, L. Liu, and J. Li, "A mixture model for expert finding," *Lecture Notes in Computer Science*, vol. 5012, pp. 466–478, 2008.

[22] H. Deng, I. King, and M. R. Lyu, "Formal models for expert finding on dblp bibliography data." in *Eighth IEEE International Conference on Data Mining*, 2009, pp. 163–172.

[23] A. Daud, J. Li, L. Zhou, and F. Muhammad, "Temporal expert finding through generalized time topic modeling," *Knowledge-Based Systems*, vol. 23, no. 6, pp. 615–625, 2010.

[24] S. Momtazi and F. Naumann, "Topic modeling for expert finding using latent dirichlet allocation," *Wiley Interdisciplinary Reviews Data Mining and Knowledge Discovery*, vol. 3, no. 5, p. 346C353, 2013.

[25] J. Liu, L. I. Qi, B. Liu, and Y. Zhang, "An expert finding method based on topic model," *Journal of National University of Defense Technology*, vol. 35, no. 2, pp. 127–131, 2013.

[26] L. Lin, Z. Xu, Y. Ding, and X. Liu, "Finding topic-level experts in scholarly networks," *Scientometrics*, vol. 97, no. 3, pp. 797–819, 2013.

[27] S. H. Hashemi, M. Neshati, and H. Beigy, "Expertise retrieval in bibliographic network: a topic dominance learning approach," pp. 1117–1126, 2013.

[28] Yang, Liu, Qiu, Minghui, Gottipati, Swapna, Zhu, Feida, Jiang, and Jing, "Cqarank: jointly model topics and expertise in community question answering," 2013.

[29] J. Wei, J. He, K. Chen, Y. Zhou, and Z. Tang, "Collaborative filtering and deep learning based recommendation system for cold start items," *Expert Systems with Applications*, vol. 69, pp. 29–39, 2017.

[30] Q. Li and X. Zheng, "Deep collaborative autoencoder for recommender systems: A unified framework for explicit and implicit feedback," 2017.

[31] H. Ying, L. Chen, Y. Xiong, and J. Wu, "Collaborative deep ranking: A hybrid pair-wise recommendation algorithm with implicit feedback," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2016, pp. 555–567.

[32] S. K. Rani, K. Raju, and V. V. Kumari, "Expert finding system using latent effort ranking in academic social networks," *International Journal of Information Technology and Computer Science*, vol. 7, no. 2, pp. 21–27, 2015.

[33] M. Karimzadehgan, R. W. White, and M. Richardson, "Enhancing expert finding using organizational hierarchies," in *European Conference on Ir Research on Advances in Information Retrieval*, 2009, pp. 177–188.

[34] DawitYimam-Seid and AlfredKobsa, "Expert-finding systems for organizations: Problem and domain analysis and the demoir approach," *Journal of Organizational Computing*, vol. 13, no. 1, pp. 1–24, 2003.

[35] A. Bozzon, M. Brambilla, S. Ceri, M. Silvestri, and G. Vesci, "Choosing the right crowd: expert finding in social networks," in *International Conference on Extending Database Technology*, 2013, pp. 637–648.

[36] A. Kardan, A. Omidvar, and F. Farahmandnia, "Expert finding on social network with link analysis approach," in *Electrical Engineering*, 2011, pp. 1–6.

[37] X. Li, J. Ma, Y. Yang, and D. Wang, "A service mode of expert finding in social network," in *International Conference on Service Sciences*, 2013, pp. 220–223.

[38] X. Cheng, S. Zhu, G. Chen, and S. Su, "Exploiting user feedback for expert finding in community question answering," in *IEEE International Conference on Data Mining Workshop*, 2015, pp. 295–302.

[39] D. Mimno and A. Mccallum, "Expertise modeling for matching papers with reviewers," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Jose, California, Usa, August*, 2007, pp. 500–509.

[40] S. Liang and M. D. Rijke, *Formal language models for finding groups of experts*. Pergamon Press, Inc., 2016.

[41] F. Alarfaj, U. Kruschwitz, D. Hunter, and C. Fox, "Finding the right supervisor: expert-finding in a university domain," pp. 1–6, 2013.

[42] H. Li, S. Jin, and L. Shudong, "A hybrid model for experts finding in community question answering," in *International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, 2015, pp. 176–185.

[43] H. B. Mahmood Neshati, Zohreh Fallahnejad, "On dynamicity of expert finding in community question answering," pp. 1026–1042, 2017.

[44] X. Liu, S. Ye, X. Li, Y. Luo, and Y. Rao, "Zhihurank: A topic-sensitive expert finding algorithm in community question answering websites," 2015.

[45] Z. Zhao, F. Wei, M. Zhou, and W. Ng, *Cold-Start Expert Finding in Community Question Answering via Graph Regularization*. Springer International Publishing, 2015.

[46] Y. Koren, R. Bell, C. Volinsky *et al.*, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[47] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," *IEEE Internet Computing*, vol. 7, no. 1, pp. 76–80, 2003.

[48] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 426–434.

[49] H. Dai, Y. Wang, R. Trivedi, and L. Song, "Recurrent coevolutionary feature embedding processes for recommendation," 2016.

[50] S. Rendle, "Factorization machines," *IEEE International Conference on Data Mining*, pp. 995–1000, 2011.

[51] ——, "Factorization machines with libFM," *ACM Trans. Intell. Syst. Technol.*, vol. 3, no. 3, pp. 57:1–57:22, 2012.

[52] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.

[53] T. Chen, "Introduction to Boosted Trees," 2014, http://homes.cs.washington.edu/~tqchen/pdf/BoostedTree.pdf.

[54] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," *arXiv preprint arXiv:1603.02754*, 2016.

[55] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "Autorec: Autoencoders meet collaborative filtering," in *Proceedings of the 24th International Conference on World Wide Web*. ACM, 2015, pp. 111–112.

[56] Y. Zheng, B. Tang, W. Ding, and H. Zhou, "A neural autoregressive approach to collaborative filtering," *arXiv preprint arXiv:1605.09477*, 2016.

[57] S. Wan, Y. Lan, J. Guo, J. Xu, L. Pang, and X. Cheng, "Matchsrnn: Modeling the recursive matching structure with spatial rnn," in *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*, 2016.

[58] T. Joachims, "Training linear svms in linear time," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2006, pp. 217–226.

[59] Z. H. Zhou, *Ensemble Methods: Foundations and Algorithms*. CRC Press, 2012.

[60] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 26, no. 2, pp. 123–140, 1996.

[61] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting," *Annals of Statistics*, vol. 28, no. 2, pp. 337–374, 2000.

[62] R. W. Johnson, "An introduction to the bootstrap," *Teaching Statistics*, vol. 23, no. 2, p. 49C54, 2001.

[63] T. Chen, W. Zhang, Q. Lu, K. Chen, Z. Zheng, and Y. Yu, "SVDFeature: A toolkit for feature-based collaborative filtering," *Journal of Machine Learning Research*, vol. 13, pp. 3619–3622, 2012. [Online]. Available: http://www.jmlr.org/papers/volume13/chen12a/chen12a.pdf

[64] N. Pezzotti, B. Lelieveldt, L. V. D. Maaten, T. Hollt, E. Eisemann, and A. Vilanova, "Approximated and user steerable tsne for progressive visual analytics," *IEEE Trans Vis Comput Graph*, vol. PP, no. 99, pp. 1–1, 2017.

[65] P. Paatero and U. Tapper, "Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values," *Environmetrics*, vol. 5, no. 2, pp. 111–126, 1994.

[66] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Computer Science*, 2014.

[67] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: online learning of social representations," pp. 701–710, 2014.