# Benchmarking Machine Learning Methods for Expert Recommendation

Seyedeh Mina Mousavifar
sem311@mail.usask.ca
University of Saskatchewan
Saskatoon, SK, Canada

Amirabbas Jalali
amj301@mail.usask.ca
University of Saskatchewan
Saskatoon, SK, Canada

Fidelia Orji
fao583@mail.usask.ca
University of Saskatchewan
Saskatoon, SK, Canada

## ABSTRACT

Recommending experts to answer questions, is a core function for community question and answer (CQA) platforms, such as Stack Overflow(SO). In this paper, we implemented six existing machine learning algorithms to recommend experts in CQA platforms. As part of our implementation, we engineered novel features to represent the questions and user's expertise and used them as input for the machine learning algorithms. Finally, we conducted extensive experiments on a large real-world dataset and benchmarked the performance of the six machine learning algorithms on the experiment recommendation task. Empirically, we found that none of the six machine learning models is able to dominate on all types of evaluation metrics, and we discussed the challenges in the expert recommendation task.

## KEYWORDS

community question answering, expert recommendation, naive bayes, support vector machine, linear regression, random forest

**ACM Reference Format:**
Seyedeh Mina Mousavifar, Amirabbas Jalali, and Fidelia Orji. 2020. Benchmarking Machine Learning Methods for Expert Recommendation. In *CMPT820: Machine Learning course, Winter, 2020, University of Saskatchewan, SK.* , 8 pages.

## 1 INTRODUCTION

The main concern in SO platform is the absence of successful coordination among questions and the potential great answerers, which unfavourably influences effective knowledge acquisition. From one viewpoint, a question may experience several low-quality answers without receiving any proper answer in a limited timeframe. Differently, an expert might encounter various new inquiries without being able to distinguish their question of interest rapidly. Under this circumstance, expert recommendation is a promising approach. According to previous studies [1], most answers and knowledge in the communities originate from only a minority of users. A recent study on Stack Overflow and Quora [2] demonstrates that these platforms incorporate profoundly committed domain experts, who target not only the requester question but also a long-lasting answer to a broader crowd. Consequently, expert recommendation would bring experts' attention to relevant questions effectively and instantly, leading to stronger communities in CQA. The purpose of the expert recommender system is to suggest relevant experts for a given question. Ranking experts based on experts' features, questions' features and their interaction require effectively adjusting the importance of these features and discovering proper ranking. However, due to the complexity of this problem, we cast our recommendation problem into a classification problem, in which the recommender would predict whether a question will be answered by an expert or not. Classification methods can easily utilize multiple aspects of features from the expert, question, and answer compared to "language-based" and "network-based" models which only consider characteristics extracted from text and social network relation respectively [29].

This paper explores the application of machine learning methods rather than recommendation based on networks and textual features such as tags that has been done by many previous work. We applied methods such as Naive Bayes [10], Support Vector Machine [20], Logistic Regression [14], and Random Forest [9] for expert classification on 2019 SO dataset, using question and answer features(e.g., community score, text similarity, tags, view count) and expert features such as (e.g., community badges, reputation, upvote).

We summarize this project contributions as follows:

- We proposed a traditional machine learning pipeline for expert recommendation in online social collaborative platform. As part of the proposed pipeline, we investigated and designed features that are representative of users and questions in SO for expert recommendations.
- We explored and benchmarked multiple traditional machine learning models for the expert recommendation task.
- We evaluated our proposed pipeline by conducting extensive experiments on a large real-world dataset.

## 2 LITERATURE REVIEW

One of the strategies used in community question and answer (CQA) platforms to ensure the provision of timely and quality answers is the expert recommendation. The issue of recommending experts in these platforms has over the years attracted the attention of researchers. According to previous research [5], people join these platforms as a result of their timely and quality response to questions posted on them. Recommendation on these platforms involves linking questions to platforms users who are knowledgeable in a specific question domain so that they can provide quality-answers to the question. Several research works have tried to find experts for questions in different CQA platforms using various approaches such as graph-based algorithms [24], social network analysis or link analysis [16], ranking metrics [32], and statistical models [23]. These research works reported reasonable success in the area of experts finding using their various methods. However, recommending experts in CQA platforms is still posing challenging problems as various experts in a platform needs to be identified with very high accuracy. As a result, other methods such as machine learning algorithms are exploited to improve accuracy. Machine learning algorithms have been shown to improve classification and prediction problems in various areas, but limited research exists on assessing state-of-the-art machine learning algorithms for expert finding in CQA.

Currently, researchers are exploring machine learning and deep learning models for recommendation systems. Research [31] has shown that using traditional language models for recommending experts on CQA could lead to word mismatch in question routing as a result of data sparseness. Liu et al. [17] addressed the issue of expert findings in an online CQA – Wenwo. The authors explored expert findings through the construction of a binary classifier using post style (for questions and answers) and profile-based features. They reported that the approach resulted in a better routing of questions as many active and new users responded to routed questions. Pal et al. [20] investigated the use of selection bias of questions in grouping users in a CQA platform. Using the Gaussian classification model, the authors evaluated their model with the SO dataset. The results of the research reveal that the approach can effectively differentiate experts among other users. Dijk et al.[10] transformed early detection of topical expertise issues into a classification problem. They extracted textual, behavioural, and time-aware features from the SO dataset to predict whether a user will be an expert in the future. According to the authors, experts are users with ten or more accepted answers on the SO platform. Their goal was to classify users as experts or non-experts. The authors created topic profiles using tags associated with questions. The questions, answers, and comments associated with topics were profiled and ranked using the td-idf scoring system. They implemented and evaluated three classification algorithms: Gaussian Naïve Bayes, Random Forest, and Linear Support Vector. The results of their research revealed that Random Forest performed better than Gaussian Naïve Bayes and Linear Support Vector. Our research is related to Dijk et al.'s [10] work. However, the way we designed our features is different from their own. Their feature construction included time-based features while our work involves users' features and questions' features. In addition, our work will evaluate the performance of four traditional machine learning models: Naïve Bayes, Support Vector Machine, Logistic Regression, and Random Forest. To the best of our knowledge, no existing research with similar features construction has evaluated the five mentioned machine learning models using the SO dataset.

Based on existing research, the performance of expert finding models is highly dependent on feature selection. For example, [16], [30] in their research on expert findings established that reputation and domain knowledge of users are relevant features that could be employed in expert finding models. Research [16] derived the knowledge profile of users from the history of previously accepted questions and answers of the users. In their research, question-answer pairs were converted to vectors using the tf-idf approach and the approach considered the relevance of past questions to the target question. Liu et al.'s [16] work is closely related to our research because our research employed the same approach in deriving the knowledge domain of SO expert users. However, our research differs from Liu et al.'s work as their features include authority and they employed the following methods KGrade, ExoertHITS, ExperPRAnk, VSM, and CBDM which are quite different from the machine learning algorithms we investigated in this research (Naïve Bayes, Support Vector Machine, Logistic Regression, and Random Forest). In addition, their methods were evaluated with dataset obtained from Yahoo! Answer Taiwan while ours uses the Stack Overflow dataset.

## 3 ALGORITHM DESCRIPTION

We first present the feature engineering process, in which we extract features from questions and answers textual data. Then, we elaborate on different classification methods that would predict whether a question will be answered by an expert or not.

### 3.1 Feature Engineering

Evaluation of similarity in questions and experts' answers, which is a textual data is one of the most critical tasks for expert recommendation in SO. Question textual features were extracted based on the question title. The user textual features were extracted based on the titles of questions that the user has previously answered. The raw data, as a sequence of symbols with variable length and sequential dependency, cannot be fed directly into the ML methods. On one hand, if we treat this text as nominal data, many methods would expect numerical feature vectors with a fixed size. On the other hand, if we manage this data as categorical input, we would have numerous features without any measure of importance between them. Hence, we use methods from information retrieval domain. The most popular technique for text feature extraction term frequency inverse document frequency (TF-IDF) [26], which provides a numerical statistic to show importance of a word to a document in a corpus. This model, convert the textual representation of information into the sparse features. In this project, we extract textual features from "question titles". To obtain a more meaningful corpus, we first tokenize question titles, then remove stop words and finally stem the words to their root using Porter Stemmer algorithm [21].

TF-IDF consists of two statistics, term frequency and inverse document frequency. Term frequency(tf) measures frequency of a word in a document, which is the number of occurrences of a term in a document divided by the whole number of terms in the document for normalization to remove the dependency of frequency on document length. Inverse document frequency(idf) measures the importance of the document in the corpus. Document frequency is the number of occurrences of a term in the documents in the corpus divided by the whole number of documents for normalization. The inverse of document frequency shows the rareness of a term in the corpus. For instance, if the word is very common and appears in many documents, this number will approach 0. Otherwise, it will approach 1.

$$tf(t,d) = \frac{f_{t,d}}{\sum_{t'} f_{t',d}}$$

$$df(t) = \sum_{d'} f(t,d')$$

$$idf(t,D) = \log(\frac{N}{1+df(t)})$$

$$tfidf(t,d,D) = tf(t,d) \cdot idf(t,D)$$

Where $t$ and $d$ denotes term and document respectively, $N$ is the total number of documents in the corpus, and $D$ shows the corpus. At last, these TF-IDF vectors are normalized by *"Euclicidian Norm"*.

## 3.2 Machine Learning Models

The input to our machine learning models is the feature vector, that we denote by $\mathbf{x}_i$, representing vector of question features $\mathbf{q}_m$, the vector of expert features $\mathbf{u}_n$, and the label indicating whether an expert has answered the question $r_i$.

$$\mathbf{x}_i = (\mathbf{q}_m, \mathbf{u}_n, r_i)$$

*3.2.1 Naive Bayes.* This method is a simple probabilistic model that applies *"Bayes Theorem"* with the strong assumption of "conditional independence" between features. This model fits our problem for its high scalability and linear time complexity because the number of parameters is linear with the number of features. We only have nominal features, so we use Gaussian Naive Bayes for our binary classification.

$$P(y_j|\mathbf{x}_i) = \frac{P(\mathbf{x}_i|y_j)P(y_j)}{P(\mathbf{x}_i)}$$

Where $\mathbf{x}_i$ is the feature vector of samples $i$, $i \in 1, 2, ..., n$, and $y_j$ is the notation of class $j$, $j \in 0, 1$. Classes are binarized 1 or 0. 1 means expert will answer question , and 0 otherwise. $P(\mathbf{x}_i|y_j)$ is the probability of observing sample $\mathbf{x}_i$ given that it belongs to class $y_j$.

Further, the class condition probabilities of individual features $d$ are as follows because of conditional independence assumption.

$$P(\mathbf{x}|y_j) = P(x_1|y_j)...P(x_d|y_j) = \Pi_{k=1}^{d} P(x_k|y_j)$$

Where $d$ is the total number of features and $x_k$ represent each feature.

Finally, the decision rule is:

$$predicted\ class\ label \leftarrow arg\ max\ P(y_j|\mathbf{x}_i) \quad for\ j = 0, 1$$

*3.2.2 Random Forest [7].* This approach consists of a large number of individual decision trees. Decision tree [8] is a supervised learning method that aims to predict the target variable by inferring decision rules from features. In this tree, leaves are the class labels and branches are the decision rule. The tree is built from the top by a feature that best splits the set of items. There are two methods for measuring better split, Gini impurity[8] and information gain [22]. Gini impurity measures the ratio of errors when an item is labelled randomly based on the distribution of labels in the set. This method is computed as follows:

$$I_G(p) = \sum_{i=1}^{J} p_i(1-p_i) = 1 - \sum_{i=1}^{J} p_i^2$$

Where $J$ is the number of classes, and $p_i$ is the fraction of items labelled with class $i$ in the set.

Information gain measures the amount of information a feature conveys about the class, which indicates the expected amount of information required for labelling an item. This information is measured by entropy that shows the ratio of each class in the child node obtained from the split in the tree. This method is computed as follows:

$$H(T) = -\sum_{i=1}^{J} p_i \log_2 p_i$$

Where $H$ indicates entropy and $p_i$ shows the ratio of each class in the child node.

$$S_a(v) = \{\mathbf{x} \in T | x_a = v\}$$

Where $S_a(v)$ is the set of training samples of $T$ training set, in which feature $a$ is equal to the value of $v$.

$$\begin{aligned} I_G(T,a) &= H(T) - H(T|a) \\ &= H(T) - \sum_{v \in vals(a)} \frac{|S_a(v)|}{|T|} H(S_a(v)) \end{aligned}$$

Where $I_G(T,a)$ is the mutual information on the training set, which measures the total entropy for a feature, in case a unique classification can be made based on each value of the feature.

The set of decision trees operate as an ensemble, which uses multiple learning algorithms to obtain better predictive performance compared to each learning algorithm separately. This algorithm aims to utilize the wisdom of crowds in which each tree in the random forest outputs a class prediction, and the class with the most votes becomes the model's prediction. Random forest allows each tree to randomly sample from the dataset with replacement, which results in different trees. These different trees would use diverse sets of features for prediction and add variability to this model. The essential characteristic in the random forest method is obtaining trees with low correlation with each other, which leads to better performance. While the algorithm via feature randomness tries to establish these low correlations, the selection of features and the hyper-parameters also impact this correlation. The hyper-parameters in random forest are number of trees, maximum depth of tree, maximum number of features for trees to randomly sample from, maximum number of nodes allowed as leaf nodes, minimum sample split which indicates number of samples needed to split a node, bootstrap and bootstrap features which shows sampling with or without replacement.

*3.2.3 Logistic Regression [18].* This method is a supervised binary classifier that uses a logistic function to model a binary dependent variable based on features. Logistic Regression presumes a linear relationship between the features and the log-odds of each class. Logistic Regression has equations similar to *"Linear Regression"* model, but it limits the cost function between 0 and 1, so it uses *"Sigmoid function"* to map predicted values to probabilities.

$$Z = \mathbf{w}^T\mathbf{x} = w_0 + w_1x_1 + \cdots + w_nx_d$$

$$P(Y=1|\mathbf{x}_i) = \frac{1}{1+e^{-Z}} = \frac{1}{1+e^{-(w_0+w_1x_1+\cdots+w_nx_d)}}$$

Where $w_i$ is a weight parameter for each feature tuned during learning. In the learning process, the model aims to minimize the error between the predicted and original values using gradient descent algorithm [25]. This error can be both calculated by Manhattan norm($l1$) or Euclicidian norm$l2$. Gradient descent is an optimization algorithm for approaching a local minimum by taking steps proportional to the negative value of the gradient of the function at the current point.

$$l2\,norm: \; error(\mathbf{w},\mathbf{X}) = \sum_{i=1}^{n}(y_i-\hat{y}_i)^2$$

Where $i$ denotes training set samples and $\hat{y}_i$ is the predicted value.

$$\mathbf{w} = \mathbf{w} - \text{gradient of error} = \mathbf{w} - [\alpha\mathbf{x}(y_i-\hat{y}_i)\odot\hat{y}_i\odot(1-\hat{y}_i)]$$

For optimizing the error function, various approaches can be applied. For instance, *"Newton's Method"* [12] considers both first and second partial derivatives for optimization; however, this method is computationally expensive due to the second derivatives calculation. *"Limited-memory Broyden Fletcher Goldfarb Shanno Algorithm"(LBFGS)* [15] method is similar to Newton's method, but the second derivatives are approximated by gradient evaluations. This approach has the best performance in small datasets but might not converge in large datasets. *"A Library for Large Linear Classification"(LIBLINEAR)* [11] approach utilizes a coordinate descent algorithm that performs approximate minimization along coordinate directions or coordinate hyperplanes. This method is recommended for high dimensional datasets but lacks in parallel execution. *"Stochastic Average Gradient descent"(SAG)* [27] method optimizes the sum of a finite number of smooth convex functions. This method is a faster solver for large datasets because of utilizing a memory of previous gradient values. Due to the large size and high dimensionality of our data, we examined both LIBLINEAR and SAG solver.

We should note that logistic regression outputs the probability of the class, so we should use a threshold to map probabilities higher than this threshold to the current class and lower probabilities to the other class.

$$P(Y=1|\mathbf{x}_i) = 0.5 \longrightarrow \text{expert will answer the question}$$
$$P(Y=1|\mathbf{x}_i) \leq 0.5 \longrightarrow \text{expert won't answer the question}$$

*3.2.4 Support Vector Machine [6].* SVM is a non-probabilistic supervised classifier, intending to find a hyperplane in D-dimensional space(D — the number of features) that classifies the data points. SVM is capable of both linear and non-linear classification. This model fits our problem because it can utilize kernels to map inputs into high-dimensional feature spaces implicitly. In this algorithm, we aim to maximize the margin between the data points and the hyperplane. So our objective is minimizing the following function by taking partial derivatives with respect to the weights to find the gradients.

$$Z = [\frac{1}{n}\sum_{i=1}^{n} max(0, 1-y_i(\mathbf{w}^T\mathbf{x}_i))] + \lambda\,\|\mathbf{w}\|^2$$

$$\frac{\partial}{\partial w_k}\lambda\,\|\mathbf{w}\|^2 = 2\lambda w_k$$

$$\frac{\partial}{\partial w_k} max(0, 1-y_i(\mathbf{w}^T\mathbf{x}_i)) = \{0, -\mathbf{y_i}\odot\mathbf{x_i}\}$$

Where $\lambda$ is the regularization parameter for preventing overfitting by penalizing for additional and higher-order weights, and *alpha* is the learning rate. So the weights would be updated based on the gradients.

$$No\ misclassification \longrightarrow \mathbf{w} = \mathbf{w} - \alpha\cdot 2\lambda\mathbf{w}$$
$$Misclassification \longrightarrow \mathbf{w} = \mathbf{w} - \alpha\cdot(2\lambda\mathbf{w} - \mathbf{y_i}\odot\mathbf{x_i})$$

SVM model utilizes kernel functions for the decision function. Kernel is an approach for computing the dot product of two vectors in the transformed space. The idea is mapping the non-linear separable dataset into a higher dimensional space such that a separating hyperplane can be found.

$$K(\mathbf{x},\mathbf{y}) = \phi(\mathbf{x})^{\mathbf{T}}\phi(\mathbf{y})]$$

Where $\mathbf{x}$ and $\mathbf{y}$ are vectors of features in the input space. This means if we use a mapping function that maps our data into a higher-dimensional space, then the maximization and decision rule depends on the dot products of the mapping function for different samples. So, we need to know K and not the mapping function itself. This function is known as Kernel function, and it reduces the complexity of finding the mapping function.

SVM supports linear, polynomial and radial basis function kernel. Linear kernel is the simple inner product of the input space, which is suitable for linearly separable data.

$$Linear\ kernel: K(\mathbf{x},\mathbf{y}) = \mathbf{x}^{\mathbf{T}}\mathbf{y} + c$$

Where $c$ is the regularization parameter as the sum of a regularization term and the misclassification rate. On the one hand, for large values of $c$, a smaller margin is specified for classifying all training points correctly. This results in a lower misclassification rate on the training rate and possibly overfitting. On the other hand, a smaller $c$ encourages a more considerable margin, therefore a more straightforward decision function in a trade-off with lower training accuracy.

Polynomial kernel [6] is useful when the data points are not linearly separable. This kernel represents the similarity of vectors in a feature space over polynomials of the original variables, allowing the learning of non-linear models. However, this kernel requires normalized training data.

$$Polynomial\,kernel: \; K(\mathbf{x},\mathbf{y}) = (\mathbf{x}^{\mathbf{T}}\mathbf{y} + c)^d$$

Where $d$ is the order of the kernel, and $c$ is a parameter that allows the trade-off between the influence of higher-order terms versus lower-order terms in the polynomial. Higher-order kernels tend to overfit the training data and thus do not generalize well.

Radial Basis function[28] kernel (RBF) is a general-purpose kernel appropriate for when there is no prior knowledge about the data. We use the RBF kernel for our SVM model because of its generalization and capacity to handle our unbalanced dataset.

$$RBF\,kernel : K(\mathbf{x}, \mathbf{y}) = exp(-\frac{||\mathbf{x} - \mathbf{y}||^2}{2\sigma^2})$$

Where $||\mathbf{x} - \mathbf{y}||^2$ is the squared Euclidean distance between the two feature vectors, and $\sigma$ is a parameter that acts as a smoothing parameter that determines the influence of each of the points. This parameter indicates the range of influence of a single training sample with low values meaning far and high values meaning close. $\sigma$ can be seen as the inverse of the radius of influence of samples selected by the model as support vectors. When $\sigma$ is low, the curve of the decision boundary is shallow, and thus the decision region is vast. When $\sigma$ is high, the curve of the decision boundary is high. So this parameter adjusts the curvature of the decision boundary.

## 4 DATA DESCRIPTION

We use Stack Overflow dataset [1], which has approximately 1 million users and 19 million posts. Due to limitations on computational power, we only used 2019 posts, and consider users who have answered more than 15 questions during last year as experts. Furthermore, we only consider the accepted answer for a proposed question, because most of the unaccepted answers have a low quality, which do not contribute to expert identification. We use 80% of our records for the training set, which will be further used for 5-fold cross-validation [13] for parameter tuning, and the remaining 20% as the test set.

### 4.1 Negative Sampling

Negative sampling[19] is a method used for machine learning models that have numerous negative observations than positive ones. Furthermore, we don't have records of experts who didn't answer a question directly. Consequently, we apply negative sampling to obtain negative interactions, such that our machine learning methods would learn both positive and negative interactions. For each question, we randomly sample 4 experts who didn't answer the question as negative sample.

### 4.2 Feature Summary

- 392,798 Questions
  - Vote: Voting is how the community indicates which questions and answers are most useful and appropriate, which can be either affirmative or negative(as upvote and downvote).
  - Score: The difference between question upvotes and downvotes, between -25 to 491.
  - View Count: Number of times this question has been viewed on SO, between 6 to 94,473.
  - Title: String of question title, which is further described as tf-idf weights of most 100 frequent terms.
- 5,448 Experts, and table 1 provides summary of experts' features.

**Table 1: Summary of Active experts 2019 SO dataset**

| Feature | Min | Mean | Max |
|---|---|---|---|
| Experts Answers | 21 | 72 | 4507 |
| Reputation | 1 | 25,278 | 1,166,685 |
| Views | 23 | 3,551 | 1,911,062 |
| Upvote | 0 | 1,218 | 53,163 |
| Downvote | 0 | 716 | 77,537 |

  - Reputation: a user is awarded 10 reputation points for receiving an up vote on an answer given to a question and 10 points for the up vote of a question.
  - Views: Number of times this user has been viewed on SO.
  - Upvote: Number of affirmative votes the user has obtained in previous questions and answers.
  - Downvote: Number of negative votes the user has obtained in previous questions and answers.
  - Mean of answers' score: Average of user provided answers' score, which is the difference between answer upvotes and downvotes.
  - Previously answered questions titles: Previously answered questions titles which is described as tf-idf weights of most 100 frequent terms.
- Question and Expert Interactions
  - Interaction Matrix: The info about a question with mentioned features, accompanied by user mentioned features who has answered the question, a total of 212 features.

## 5 RESULTS

Our pipeline consists of data collection, feature engineering, and modelling phase. In the data collection phase, we retrieved the StackOverflow archive and choose the recent one year span. Further, we cleaned our textual data by removing all unnecessary characters and limiting question languages only to English characters. In feature engineering phase, we aim to extract the expert-question interaction matrix. First, we applied the TF-IDF method. Our corpus consists of 52,326 terms, which led to a gigantic sparse matrix. This large number of dimensions would be troublesome due to its vast dimension. So we limited our corpus to 100 most common terms. At this stage, we splitted our questions into 80% training set and 20% test set to extract experts' features from the provided answers. Further, we obtained experts' TF-IDF terms as the average of question titles they had answered and compute the average score of these answers.

In modelling phase, we designed models taking into cognizance the imbalanced class distribution (four negative samples per each positive sample) in our dataset. Random Forest hyperparameters were optimized using a randomized search cross-validation method [4], which executes a randomized search over parameters that samples each setting from a distribution over possible parameter values. We picked this method over the exhaustive search for its efficiency. We used f1-weighted scoring metric for optimization because of the importance of positive class; additionally, it considers class imbalance by calculating metrics for each class, and finding their average weighted score. For the Logistic Regression method, we examined

[1]https://archive.org/details/stackexchange

both *SAG* and *LIBLINEAR* solvers due to their ability to handle high dimensional datasets. Furthermore, we adjusted the class weights inversely proportional to the class frequencies considering our unbalanced dataset. For the SVM model, we utilized the radial basis function kernel for its generalization and for the value of gamma, which specifies the impact of each data point, we analyzed both auto and scale. Auto specifies gamma as inversely proportional to the number of features and scale considers the variance of samples in addition to the number of features. Moreover, we adjusted the class weights inversely proportional to the class frequencies regarding the imbalanced class distribution in our unbalanced dataset.

In the evaluation phase, we bench our models based on various metrics considering that our positive class is more critical regarding that the primary purpose of our classifiers is predicting experts who answer the question. Firstly, balanced accuracy is computed as the accuracy of samples weighted based on the inverse occurrence of its true class. Accuracy would obtain inflated results for predicting the negative class and wouldn't be an appropriate metric for our problem. Secondly, we computed precision, which measures the ability of the classifiers not to classify negative samples as positive and calculate recall to classify all positive samples correctly. Thirdly, we estimated the weighted f1-score due to our imbalance dataset, which specifies harmonic mean between precision and recall for each class and find the average weight based on the number of true samples for each class. Fourthly, we calculated the precision-recall curve by computing precision and recall at the sequence of every data point and obtain the score as the weighted average precision over two consecutive recalls. We used this metric because it focuses on the performance of the classifiers on the minority class. Finally, we computed sensitivity to find the performance of our models on predicting the positive class.

Figure 1 presents a comparison between the performance of our models. Logistic regression has the best performance in predicting the experts who will answer the question (SAG solver handles our sparse data appropriately). In contrast, SVM and Random Forest have the best accuracy and precision-recall score, indicating the ability of the models in discriminating classes and predicting both classes correctly. Interestingly Random Forest and Naive Bayes have the best precision, recall and f1-score. However, the sensitivity of Random Forest and Naive Bayes shows that these models are biased over negative class by predicting negative for most of the test set. Thus, the f1-score, precision and recall are hyped by predicting our unbalanced negative class and do not measure the performance of our models appropriately. Logistic Regression results indicate the model's capability in learning the positive class with the lowest ability to learn the negative class correctly. Therefore, this model is useful for finding the experts who will answer the question but might also offer experts who would not answer the question. SVM results show the model's acceptable performance in inferring both positive class and negative class. This model has inflated f1-score but acts the best with having acceptable sensitivity along with the highest average precision-recall score. This model performs better than other models in recommending an expert who would answer the question and not recommending an expert who would not answer the question.

## 6 DISCUSSION

Expert recommendation in SO is arduous because of the complex structure of the network, such as various topics that also have semantic relation to each other. For instance, the loop concept is repetitive in many languages. However, if an expert has answered a loop question in Python language, there is no guarantee that this user would also answer loop related questions in Go language. Thus, the programming language of the answered question is more critical than the topic of the question. Furthermore, in the current state of SO, numerous experts are knowledgeable in one area but didn't have the chance to answer any questions on this topic, which makes prior knowledge about experts misleading. Consequently, designing a model that can consider all these aspects is challenging.

Dijk et al. [10] reports f1-score of 0.74 on the former SO dataset with a different set of features, and our Random Forest classifier obtains f1-score of 0.72. The imbalanced class label in our dataset impacts f1-score, and sensitivity only considers the positive class. Therefore, the average precision-recall score is a more reliable metric, accounting for both negative and positive classes with a focus on the negative class. Naive Bayes and Random Forest are profoundly impacted by our imbalanced dataset. Random Forest has the most capability in working with a dimensional dataset, but decision trees are biased toward negative class and lead to the biased forest over the majority class. Logistic Regression and SVM overcome this overfitting by regularization.

The main problem with our dataset is its dimensionality, where the small number of TF-IDF terms result in different subjects overlap with each other, and the vast number of terms leads to the *curse of dimensionality problem*[3]. Furthermore, we compared our models using many metrics and recognize the best metric for our problem. Therefore, optimizing Random Forest classifier parameter using average precision-recall score might help in obtaining less biased trees. When Random Forest performs more logical, we can use this method for identifying the significance of features, that can be further used for dimensionality reduction.

## 7 FUTURE WORK

In this project, we proposed a machine learning pipeline for expert recommendation in StackOverflow and designed features that are representative of users and questions. We bench Naive Bayes, Support Vector Machine, Logistic Regression, and Random Forest methods on the 2019 SO dataset. SVM model performs the best in recommending an expert who would answer the question and not recommending an expert who would not answer the question, and Logistic Regression has the best performance in finding the experts who will answer the question. We examined these methods using textual features from question titles, and experts answered question titles that result in a close performance compared to the prior work on the earlier version of the dataset.

For future work, we aim to investigate the applicability of dimension reduction methods such as Principal Component Analysis in our problem. Besides, we can design a mixture of both SVM and Logistic Regression methods, where the result of Logistic Regression probability is also fed into the SVM model. Furthermore, we can design a model to recommend based on tags and the respective order of them and compare its performance to our current results.

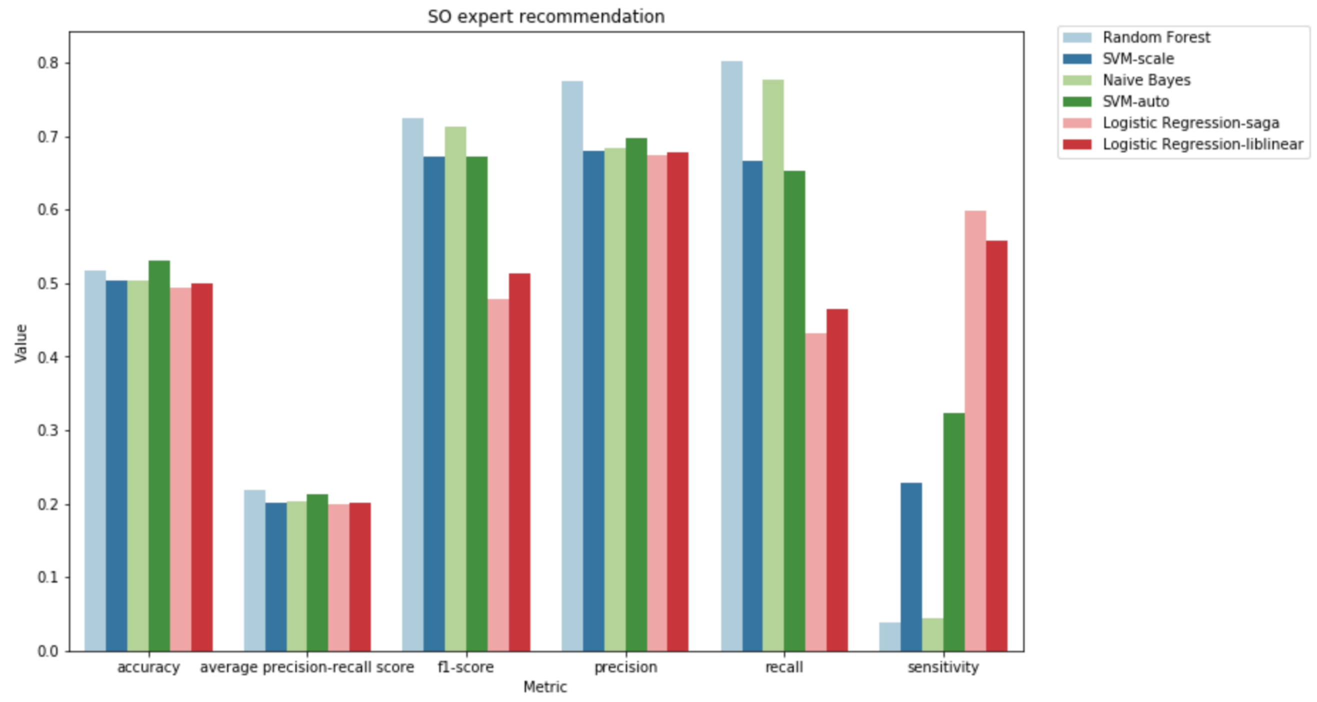

**Figure 1: Results**

## REFERENCES

[1] Lada A. Adamic, Jun Zhang, Eytan Bakshy, and Mark S. Ackerman. 2008. Knowledge sharing and yahoo answers. *Proceeding of the 17th international conference on World Wide Web - WWW 08* (May 2008), 665–674. https://doi.org/10.1145/1367497.1367587

[2] Ashton Anderson, Daniel Huttenlocher, Jon Kleinberg, and Jure Leskovec. 2012. Discovering value from community activity on focused question answering sites. *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD 12* (Aug 2012), 850–858. https://doi.org/10.1145/2339530.2339665

[3] Richard Bellman. 1966. Dynamic programming. *Science* 153, 3731 (1966), 34–37.

[4] James Bergstra and Yoshua Bengio. 2012. Random Search for Hyper-Parameter Optimization. *J. Mach. Learn. Res.* 13 (2012), 281–305. http://dblp.uni-trier.de/db/journals/jmlr/jmlr13.html#BergstraB12

[5] Vasudev Bhat, Adheesh Gokhale, Ravi Jadhav, Jagat Pudipeddi, and Leman Akoglu. 2014. Min(e)d your tags: Analysis of Question response time in StackOverflow. *2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014)* (2014), 328–335. https://doi.org/10.1109/asonam.2014.6921605

[6] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. 1992. A training algorithm for optimal margin classifiers. *Proceedings of the fifth annual workshop on Computational learning theory - COLT 92* (1992). https://doi.org/10.1145/130385.130401

[7] Leo Breiman. 2001. Random Forests. *Machine Learning* 45, 1 (2001), 5–32. https://doi.org/10.1023/A:1010933404324

[8] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. 1984. *Classification and Regression Trees.* Wadsworth and Brooks, Monterey, CA.

[9] Morakot Choetkiertikul, Daniel Avery, Hoa Khanh Dam, Truyen Tran, and Aditya Ghose. 2015. Who Will Answer My Question on Stack Overflow? *2015 24th Australasian Software Engineering Conference* (Sep 2015), 155–164. https://doi.org/10.1109/aswec.2015.28

[10] David Van Dijk, Manos Tsagkias, and Maarten De Rijke. 2015. Early Detection of Topical Expertise in Community Question Answering. *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR 15* (Apr 2015), 995–998. https://doi.org/10.1145/2766462.2767840

[11] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A Library for Large Linear Classification. *Journal of Machine Learning Research* 9 (2008).

[12] Begnaud Francis Hildebrand. 1987. *Introduction to Numerical Analysis: 2nd Edition.* Dover Publications, Inc., USA.

[13] R. Kohavi. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International joint Conference on artificial intelligence*, Vol. 14. Citeseer, 1137–1145.

[14] Long T. Le and Chirag Shah. 2016. Retrieving Rising Stars in Focused Community Question-Answering. *Intelligent Information and Database Systems Lecture Notes in Computer Science* (Mar 2016), 25–36. https://doi.org/10.1007/978-3-662-49390-8_3

[15] Dong C. Liu and Jorge Nocedal. 1989. On the Limited Memory BFGS Method for Large Scale Optimization. *Math. Program.* 45, 1–3 (Aug. 1989), 503–528.

[16] Duen-Ren Liu, Yu-Hsuan Chen, Wei-Chen Kao, and Hsiu-Wen Wang. 2013. Integrating expert profile, reputation and link analysis for expert finding in question-answering websites. *Information Processing & Management* 49, 1 (2013), 312–329. https://doi.org/10.1016/j.ipm.2012.07.002

[17] Jansen J Bernard Liu Zhe. 2014. Predicting potential responders in social Q&A based on non-QA features. *CHI'14 Extended Abstracts on Human Factors in Computing Systems* (2014), 2131–2136. https://doi.org/10.1145/2559206.2581366

[18] P. Mccullagh and J. A. Nelder. 1989. An outline of generalized linear models. *Generalized Linear Models* (1989), 21–47. https://doi.org/10.1007/978-1-4899-3242-6_2

[19] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS*. Curran Associates, Inc., 3111–3119. http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf

[20] Aditya Pal and Joseph A. Konstan. 2010. Expert identification in community question answering. *Proceedings of the 19th ACM international conference on Information and knowledge management - CIKM 10* (2010), 1505–1508. https://doi.org/10.1145/1871437.1871658

[21] Martin F Porter. 1980. An algorithm for suffix stripping. *Program* 14, 3 (1980), 130–137.

[22] J. R. Quinlan. 1986. Induction of Decision Trees. *MACH. LEARN* 1 (1986), 81–106.

[23] Fatemeh Riahi, Zainab Zolaktaf, Mahdi Shafiei, and Evangelos Milios. 2012. Finding expert users in community question answering. *Proceedings of the 21st international conference companion on World Wide Web - WWW 12 Companion* (2012). https://doi.org/10.1145/2187980.2188202

[24] Hiemstra Djoerd Zaragoza Hugo Rode Henning, Serdyukov Pavel. 2007. Entity ranking on graphs: Studies on expert finding. *CTIT Technical Report Series* (2007).

[25] Sebastian Ruder. 2016. An overview of gradient descent optimization algorithms. arXiv:cs.LG/1609.04747

[26] Gerard Salton and Michael J. McGill. 1987. *Introduction to modern information retrieval*. McGraw-Hill Intern.

[27] Mark Schmidt, Nicolas Le Roux, and Francis Bach. 2017. Minimizing Finite Sums with the Stochastic Average Gradient. *Math. Program.* 162, 1–2 (March 2017), 83–112. https://doi.org/10.1007/s10107-016-1030-6

[28] JP. Vert, K. Tsuda, and B. Schölkopf. 2004. *A Primer on Kernel Methods*. MIT Press, Cambridge, MA, USA, 35–70.

[29] Xianzhi Wang, Chaoran Huang, Lina Yao, Boualem Benatallah, and Manqing Dong. 2018. A Survey on Expert Recommendation in Community Question Answering. *Journal of Computer Science and Technology* 33, 4 (2018), 625–653. https://doi.org/10.1007/s11390-018-1845-0

[30] He Tingting Wu Wensheng Zhou Guangyou, Zhao Jun. 2014. An empirical study of topic-sensitive probabilistic model for expert finding in question answer communities. *Knowledge-Based Systems* (2014), 136–145. https://doi.org/10.1016/j.knosys.2014.04.032

[31] Liu Kang Zhao Jun Zhou Guangyou, Lai Siwei. 2012. Topic-sensitive probabilistic model for expert finding in question answer communities. *Proceedings of the 21st ACM international conference on Information and knowledge management* (2012), 1662–1666. https://doi.org/10.1145/2396761.2398493

[32] Cai Deng He Xiaofei Yueting Zhuang Zhou Zhao, Qifan Yang. 2016. Expert finding for community-based question answering via ranking metric network learning. *IJCAI International Joint Conference on Artificial Intelligence* (2016), 3000–3006.