Automated Question Answering System based on Multiclass Prediction of Yahoo!Answers

Jiali Zhou, Jirou Xu, Lizhen Tan

***Abstract*—Seeking answers from web has been a common way for research. Our project is trying to predict the category of a question and seek to find the most similar queries in the archive given a new query entered in the database. After prediction of the question category, we would look into the archive for the predicted class, and then retrieve the best answer to the most similar question for the new query question. Categories were set to be four of the most frequent asked categories in a subset of 58,317 entries in the Yahoo! Answers corpus as of 10/25/2007. We transformed the questions into vectorized count and binary form. With the transformed vectors, we applied different machine learning models (including Random Forest, One-vs-All multi-classification with support vector machine(SVM), multinomial Nave Bayes, Bagging of multinomial Nave Bayes, and Ensemble of the mentioned models) for prediction. By feature engineering and hyper-parameter tuning, most of the models we obtained achieved a test accuracy around 80%. When choosing the most similar question, instead of using cosine similarity as the measure of two questions, we uses Okapi BM25 model, which gives a better result.**

I. Introduction

OUR project aims at searching the similar question that has previously been asked on the Internet-based knowledge exchange website Yahoo!Answers when a new question is asked. If a similar question is found, then the similar question's answer can be provided, which saves large quantity of time. In contrast to the usual search paradigm, where the question is used to search the database of potential answers, in this project, we present a model to search the database to find similiar questions, which in turn are associated with the answers we are looking for.

II. Task Definition

The process of the project is divided into two stages. Stage 1: apply machine learning algorithms to predict the question category based on Yahoo!Answers. The categories is a multi-class classification problem. The input is a questions subject and content, and we want the output to be a value of 0,1,2 or 3 indicating the categories of the questions.

An example of stage I input and output would be as follows:

Input: I havent seen Jen yet. I know her through her answers only. What should I do?

Output: 2 (indicating the category is Society & Culture, which corresponds to a predicted value of 2.)

Stage 2: provide an answer to a question. First, use Stage 1s model to predict the questions category. Next, we calculate the similarities between the given question and all the questions in the predicted category of the archive and find the question with the highest similarity score. After getting the highest-score question, we can retrieve its associated best answer. This best answer will be the output of the second stage. An example of stage II would be as follows:

Input: is there anyway to get less pigment in ur eyes?

Output: "Sorry your eye color is hereditary,make an appointment with an optometrist and have him write you a Rx for contact lenses.You can change your color with them and you can wear them even if you don't need to."

Sections Data Preparation and Feature Extraction were steps taken for Stage 1. Details of Stage 2 will be presented in the Models and Experiments subsection Okapi BM25 Model.

III. Data Preparation

The raw data is from Yahoo! Answers corpus as of 10/25/2007 in an xml file, which contains 4483032 questions and their answers. What's the format of a question in the dataset? A tipical question contains the following components: (1) Subject of the question. (2) Optional content which contains additional detail about the question. (3) The best answer selected either by the asker or by the participants through voting. (4) All answers posted for this question. (5) Optional classification tags like main category, category and subcategory. (6) The id of the user who asked the question. (7) The id of user who provided the best answer. (8) The date when the question was posted.

Although this is a huge dataset, we did not need to use the whole dataset for training purpose. We first randomly selected a portion of it as a sample dataset. We are only interested in the question and the answers. The rest attributes of the dataset can be dropped. For each question, the subject of the question is the question itself and the content which is the detail and description of the question. We combined subject and content columns into one feature because the new feature gives us more information about the question. A particular user might not ask two identical questions. Even if s/he asks a lot questions, his/her identity does not affect the question content. Under this circumstance, the id of the user as well as other non-question-related columns could be dropped.

On the other hand, main category, category and subcategory are almost the same. We chose the main category of each question as our target variable and dropped the other two. There are more than 20 different categories in the main category column, which means there are over 20 labels in the model. We noted that, as number of classes increases, it gets harder to get a well-trained model. Also, a lot of the labels are quite similar, we decided to chose four of the most frequently asked categories as our labels: Family

& Relationships, Entertainment & Music, Society & Culture, and Health (see Fig. 1 for the top 10 most frequent classes in the randomly selected data). Among these four categories, number of each of them did not differ too much; such that we had avoided unbalanced data. With the formatted data, our training set contained 58,317 entries and the test set contained 8,260 entries.

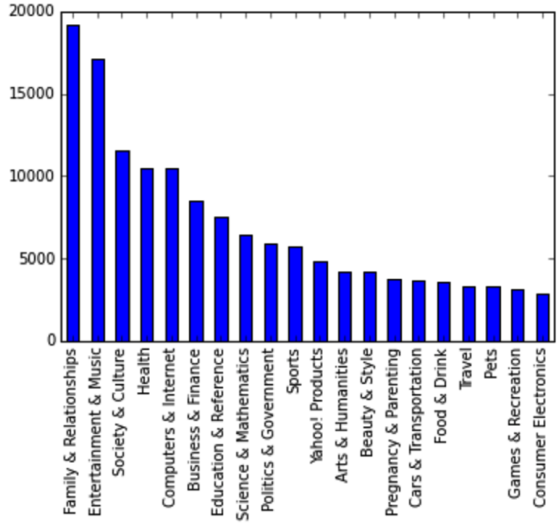

Fig. 1. Statistics of different categories.

IV. Feature Extraction

The input data are raw texts. They cannot be used for mathematical calculation and evaluation. We had to first transform them into numeric numbers. The first attempt was to use term frequency, namely it was the count of a particular term in a document (by just using the word frequency, we obtained a bag of 57,620 words for our training set). However, many words such as "the", "he", "she", etc. and punctuation do not provide additional useful information for a class prediction. Capitalized letters might not provide additional information as well (except there are a lot of proper nouns. In our project, since questions were not formally written, human typing might be sloppy. We decided to ignore the importance of capitalization), we converted all letters to lowercase. In order to preserve word order, we also experimented with using unigram and bigram. The above steps has help formed the first version of feature vectors.

Other than using only term frequency, we also wanted to know ow important a word is to a question in a large collection. We constructed a new feature vectors for the words by using term frequencyinverse document frequency[7] (tf-idf).

$$tf - idf_{i,j} = tf_{i,j} \times idf_i$$

where term frequency:

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

and inverse document frequency:

$$idf_i = \log \frac{|D|}{|\{j : t_i \in d_j\}|}$$

with $|D|$ is the total number of documents in the corpus. $|j : t_i \in d_j|$ is the number of documents where the term t_i appears. (Note: there are variations of idf)

V. Evaluation Metrics

The major evaluation metric we focused on was the prediction accuracy of the test set for all models. The accuracy is based on the confusion matrix:

TABLE I
Confusion maxtrix

Prediction	$class_i$	all other classes
Actual		
$class_i$	tp	fn
all other classes	fp	tn

The final accuracy is the mean accuracy of all classes. We experimented with different methods, and tried to look for models which would achieve high accuracy. Table II shows the accuracy for each model we used for the project.

In addition, since accuracy relates to true positive rate and false positive rate, the evaluation metric we chose was AUC (area under curve) using ROC (receiver operating characteristic) curves for each model. (see Fig. 4)

VI. Models and Experiments

As for the first stage, the problem was to solve a multi-class prediction (four classes). We first tried three models which are commonly used for multi-class prediction: a) multinomial Naive Bayes algorithm, b) SVM(One-Vs-All) classification and c) random forest. Based on these three models, we built two additional ensemble models, which were bagging of multinomial Naive Bayes and an ensemble of all previously mentioned models.

A. Multinomial Naive Bayes Model (MNB)[1]

Naive Bayes has strong independence assumption for each word in a given class. For each class of y, it has a different distribution for different set of multinomial parameters. Under each parameter, we calculated the posterior probability of all the words in a document, and chose the parameter which gives the largest probability.

In addition, the parameters were estimated by a smoothed version of maximum likelihood:

$$\hat{\theta}_{ci} = \frac{N_{ci} + \alpha_i}{N_c + \alpha}$$

where N_{ci} - number of times word i appears in the documents in class c, N_c - number of word occurrences in class c, α_i is the something factor, and α is the sum of α_i's.

When extracting features from the raw documents, we chose from using unigram or bigram. Then we compared the result of using inverse-document-frequency or not. Finally, we set

different values to the smoothing parameter of Multinomial Naive Bayes model. After the entire parameter adjusting process, the model was built with parameters which give the highest accuracy. (Test accuracy is shown in Table II)

B. Bagging of Multinomial Naive Bayes Model

Taking the parameters selected from section A, we drew 10 bootstrap samples from the training set to train 10 different multinomial Naive Bayes classifiers. For each bootstrap sample, we set the sample size to be 80% of the total training set size. The final prediction was given by the consensus class, i.e., the mode of all predictions from the 10 MNB classifiers.

C. Random Forest Model[2]

The random forest model was built from 50 trees with max features to be $\sqrt{n}$ where n is the total number of features. By removing features, we had added more bias to avoid overfitting. The final prediction is the majority vote by the 50 trees in the forest.

The baseline test accuracy using frequency counts solely (no stop word removal and no bigram) as features was 62.2%, which was not good. Transforming feature vectors from frequency to tfidf increased the the accuracy by 4%. Further improvement was done by removing stop words and adding bigrams. The final result achieved by random forest algorithm was 74.5% accuracy.

Fig. 2 has shown the precision-recall curve[6] for the final improved random forest model.

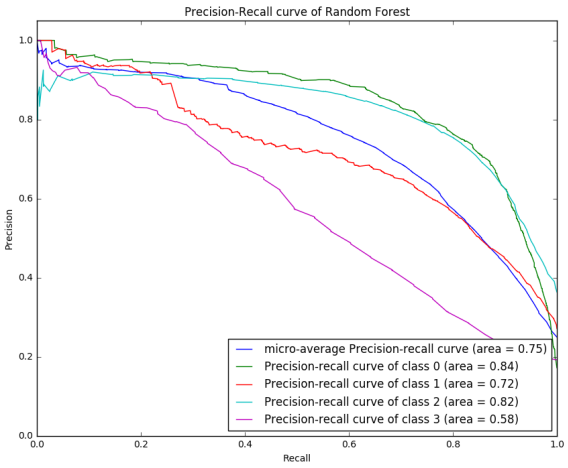

Fig. 2. precision-recall curve of Random Forest model.

D. SVM(One-Vs-All) Classification Model[3]

For SVM, we experimented both one-vs-one and one-vs-all, it turned out that one-vs-one algorithm runs much more slowly than one-vs-all, and both have almost the same accuracy, thus we decided to present only the result of one-vs-all algorithm here.

One-vs-all is a method of fitting a class against all of the other classes. For example, if the true class is Family & Relationships, then the other three are all marked as false.

The SVM one-vs-all(OVA) model was built by minimizing the hinge loss with L2 regularization (L2 regularization to control the weight of each feature. By applying the constrain, it decreased overfitting of our feature-rich model). We looked for separating planes with largest margins which separate each class from all of its adversaries.

When using term frequency solely as feature, the prediction accuracy in the test set increased from 66% to 74.4% after removing stop words. Further improvement was done by transforming term frequency feature to tfidf feature. The prediction accuracy then increased to 79.9%. Bigram instead of unigram also help increase a little bit. The final tuned SVM (OVA) model has achieved an accuracy about 80.8%.

The performance of SVM (OVA) on the four classes is shown in Fig. 3.

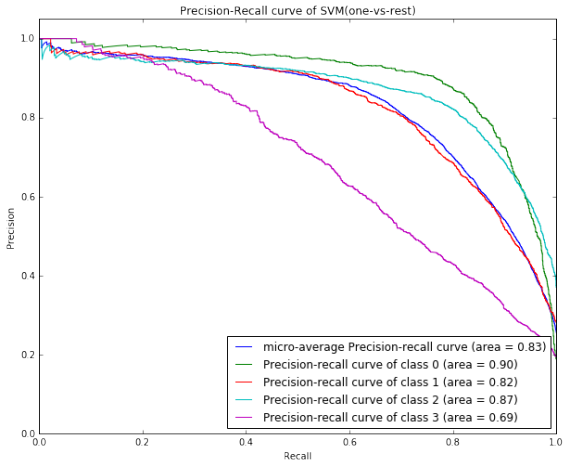

Fig. 3. precision-recall curve of SVM model.

E. Ensemble of Models

The good performance of bagging the multinomial Naive Bayes helped sprout the idea of whether bagging the already-built models would give better test accuracy. We gathered the predicted results from all models mentioned in the previous sections, then took the majority vote from the ensemble as the final prediction.

TABLE II
MODEL PREDICTION ACCURACY

Classifiers	Test accuracy(%)
Random forest	75.0726
SVM(one-vs-all)	80.7990
Multinomial Naive Bayesian	80.7143
Bagging of Multinomial Naive Bayesian	81.5811
Ensemble of models	80.7854

F. Okapi BM25 Model (Stage II)[4]

The goal of the second stage was to look for an answer to a newly asked question from the archived dataset. After

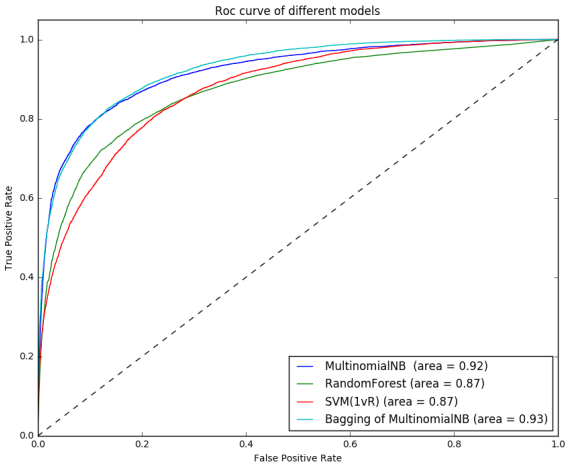

Fig. 4. ROC curves of different models.

obtaining a well-trained multi-class model from Stage 1, when a new question was asked, this model could be used to narrow the query search down to a smaller domain. This could save a huge amount of time compared to search all entries in a big (in the sense of big data) archive.

In our project, we built a small archive by using a new subset of the Yahoo! Answers corpus only with labels in the four categories we used for the classification model. The model we used for the second stage is Okapi BM25, which is a bag of words model and doesn't count the position into consideration.

Let's demonstrate how this model gets into the scene. Say a new question is entered as query A, and with the classification model, we predict it to be in class C. Now as query A being one of the documents in class C, Okapi BM25 calculates the IDF weight of each word in query A in the predicted class. Then Okapi BM25 calculates the term frequency of each word in query A. Finally the model calculates the BM25 score between query A and all other queries (namely the questions) in class C. The score of a query B in Class C is calculated as the following:

$$Score(B,Q) = \sum_{i=1}^{n} IDF(q_i) \frac{f(q_i,B)(k_1+1)}{f(q_i,B) + k_1(1-b+b\frac{|B|}{avgQL})}$$

q_i is the word in query A. $f(q_i, B)$ is q_i's term frequency in the query B,$|B|$ is the length of query B in words. avgQL is the average query length in the class C, k_1 and b are two free parameters. $IDF(q_i)$ is the IDF weight of q_i, It is usually computed as:

$$IDF(q_i) = log\frac{N - n(q_i) + 0.5}{n(q_i) + 0.5}$$

Where N is the total number of queries in class C.And $n(q_i)$ is the number of queries in the class that contain q_i.

However, since a longer query has more words, which causes the BM25 score to be larger. As a result, the distribution of the BM scores for all the queries becomes too fragmented. We made some adjustment to reduce effect. That was dividing the BM25 score with the length of query A. Finally, the best-answer to the archived query which has the highest BM25 score would be output to the screen.

Table III and Table IV below shows sample query results for each category. Query A is the new input question, query B is the question with the highest BM25 score in the archived category data.

TABLE III
QUERY RESULTS

Category	Query A
Family & Relationships	There is this girl that I really like in my school How do I ask this really cute girl out Im in the 6th grade
Entertainment & Music	Anybody on the east coast who watched the american idol finale who won was it Kat or Taylor
Society & Culture	My wife expects me to literally worship the ground she walks on Would that make me a pagan
Computers & Internet	how can i know where from is this IP address 62162239162

TABLE IV
QUERY RESULTS CONTINUED

Most similar query B	Best answer
How do I ask a girl out	simple, will you go out with me?
american idol who should win	KATHARINE!!! BUT TAYLOR WILL WIN!!!!!
are Catholics Literally pagan idol worshipers	Yes, literally
how do you find where an IP address originated	Yes I can, post the IP address and I will post your answer.

VII. DISCUSSION

In Stage 1, we have found that the random forest had the lowest accuracy. By comparing the precision-recall curves between random forest and SVM (OVA), we could easily see that SVM (OVA) beat random forest for every class prediction. This is also true for the other models when comparing with random forest.

Usually, data scientists use Naive Bayes (NB) as the baseline model for text analysis. So did we. We originally hoped to get a general idea from the data by using multinomial NB, however, after the parameter tuning process, multinomial NB performed surprisingly well in our project setup. The reason might be that, most text inputs in this problem are just short sentences. Even with the content which give further description of the question, the final input is still concise compared to long documents, such that the strong independence assumption does not hurt so bad in our case. We could even get a little bit most boost by applying bagging the multinomial NB models. (see Table II)

From the inspiration of bagging the multinomial NB models, we tried applying the same method, but the base models for the new ensemble were all the trained models we have obtained so far. However, it did not perform better than the multinomial bagging model. A possible reason would be that

the relatively degenerative random forest model dragged down the performance.

Besides the performance of the models, we have noticed that for each model, Computer & Internet was the hardest one to get a correct prediction. When checking the entries in the category, we found that a lot of questions refer to webpage links and computer related proper nouns. Since the links are not common words, it might bring a lot of confusion to model training. Also, the computer-related proper nouns might be assigned a little weight after we transfer them into lowercase words.

In Stage 2, we have tried to check how similar a query (an input question) to the corresponding similar question retrieval from our self-built archive by the BM25 model by eye. We found that most of the results were truly similar to the input query, such as 'Anybody on the east coast who watched the american idol finale who won was it Kat or Taylor' and 'american idol who should win'. The model worked quite well on finding the proper answer to the input query. However, we also noticed that some queries did not have satisfying results as well.

Queries being predicted into the category 'Society & Culture' performed the worst. The reason might be that queries in category 'Society & Culture' tend to contain large quantity of personal names and proper nouns, which will interfere the calculation of BM25 score. Besides, one of the most important disadvantages of this model is that the sentence pattern matters too much. For instance, there are two queries 'What is the difference between A and B?' and 'What is the difference between C and D?'. The model will find that these two questions to be most similar, even though C and D are entirely different from A and B.

VIII. Conclusion and Future Work

All final models generate quite good performance in our project setting, however, further improvement is still possible.

As for the webpage link problem for categorize Computer & Internet, new features which represent the link property may be add in for training. In addition, certain proper nouns should be preserved to avoid weight reduction on important terms.

Another feature extraction method would be using the word2vec algorithm to get a different way to present the word information.

As for the querying part, it took a long time to select the most similar query from database as we had to compare the query entered with each of the query in the database. The execution time for all the score calculation gets longer and longer as the number of queries in the database increases. We wonder if we can use mapreduce or other methods to improve this situation.

IX. Acknowledgement

We would like to thank our advisor Bonnie Ray for offering us valuable and inspiring suggestions about how to implement the project and what to do when we encountered problems. And we would also like to thank professor David Rosenberg for his great and joyful lectures as well as his clear and detailed explanations in class and on piazza. In addition, we would also like to thank Yahoo! Answers for providing us the dataset for this project.

References

[1] Rennie, Jason D. M., Lawrence Shih, Jaime Teevan, and David Karger. "Tackling the Poor Assumptions of Naive Bayes Text Classifiers." *Tackling the Poor Assumptions of Naive Bayes Text Classifiers (n.d.)* : n. pag. Web. 10 May 2016.

[2] Tennyson, Elizabeth M., Shauna Sallmen, and Eric Korpela. "Space Sciences Laboratory, University of California at Berkeley, Berkeley, CA, 94720." *Proceedings of the Wisconsin Space Conference Proc. Wisc. Space Conf. 0.0 (2011)*: n. pag. Jan. 2001. Web.

[3] Tewari, Ambuj, and Peter L. Bartlett. "On the Consistency of Multiclass Classification Methods." *Learning Theory Lecture Notes in Computer Science (2005)*: 143-57. Web.

[4] "Implementation of Okapi BM25 on Python." - Backyard of LixinZhang. N.p., n.d. Web. 10 May 2016.

[5] "Gensim: Topic Modelling for Humans." Gensim: Models.word2vec Deep Learning with Word2vec. N.p., n.d. Web. 10 May 2016.

[6] "Precision-Recall." Precision-Recall Scikit-learn 0.17.1 Documentation. N.p., n.d. Web. 10 May 2016.

[7] "Term Frequency by Inverse Document Frequency." (n.d.): n. pag. Web. 11 May 2016. http://disi.unitn.it/ bernardi/Courses/DL/Slides_11_12/measures.pdf.