Finding Problem Solving Threads in Online Forum

Zhonghua Qu and **Yang Liu**
The University of Texas at Dallas
{qzh,yangl}@hlt.utdallas.edu

Abstract

Online forum is an important source that people use to find answers to questions. Most search engines simply retrieve “relevant” threads, but those are not necessarily good threads in terms of providing quality answers. In this paper we propose a two-step approach to classify online forum threads according to their informativeness in terms of question answering. We use statistical models to first categorize posts inside a thread. Then, a variety of features including post level information and other meta-data information are used to classify the thread. We show promising results using the online support forum data we have collected.

1 Introduction

Online forums often contain useful information, such as answers to questions. When people use search engines to search the forum for answers, the returned threads can be of very low quality in terms of providing informative answers. Some threads are either long conversations without any final conclusive answers or contain answers that do not work. This is especially the case in technical forums. There has been some work on extracting information from online forums. For example, Cong et al. (Cong et al., 2008) used labeled sequential patterns to detect question sentences in online forums. Ding et al. (Ding et al., 2008) used Conditional Random Fields to extract context of questions for answer detection. Huang et al. (Huang et al., 2007) used SVM to automatically extract and rank title-reply pairs from online discussion forums for chatbot knowledge. These previous studies provide a relatively good foundation for answer finding and extraction. In online environments like mailing list or forums, question answering is carried out in conversations. In this paper we take advantage of the structure of conversation and language cues to answer a basic yet important question: Does the thread actually solve the problem?

We will make use of the conversation within the thread to determine how users think of the answers. To classify the usefulness of a thread, we propose a two-step approach using statistical models. First we classify the posts inside a thread into different categories (e.g., problem description, solution, feedback). Different models using both content and contextual information are evaluated. Then, we develop features generated from the post categories together with forum meta-data to classify a thread’s usefulness. In our collection of forum threads, we show that our proposed methods achieve good results, significantly better than the rule-based baseline.

2 Data Collection and Annotation

We created our own data collection and annotation.[1] We crawled 20,000 threads from an active online forum (Oracle database support forum – general section). As an initial study, we selected 200 threads randomly and asked two annotators to label the threads and posts in them. The two annotators are computer science students with adequate knowledge to understand the content of those posts. For thread level annotation, we asked the annotators to label the thread based on whether they think it solves the problem and to what extent. We did not give annotators detailed instructions, but rather let them read the threads and make their own judgment. Each thread is given a score from 1 to 5, 1 being least helpful and 5 being most helpful. In the 200 threads we used, 50 have a usefulness score of 1, 19 with 2, 7 with 3, 61 with 4, and 83 with a score of 5. The distribution has a U

[1]Please contact the authors for data sharing.

shape – many threads are annotated as absolutely solved the problem (usefulness of 5), or absolutely not helpful at all (usefulness of 1). This also shows that the confidence of annotators is very high.

Another annotation we performed is for the posts inside each thread. We defined four classes for the posts based on their main purpose. The categories and brief explanation for them are shown in Table 1, along with the number of posts for each category. Note that sentences inside a post may have different purposes, however, we instructed the annotators to label it with its main purpose. The post classes are very imbalanced in the annotated corpus. The feedback classes account for only about 10% of all the posts, while the majority classes are problems and solutions.

Post category	Description	number
Problem	Ask a question or ask for answer clarification.	399
Solution	Give answers/clarifications to previous questions.	557
Good Feedback	State that the problem is solved.	78
Bad Feedback	State that the answer did not solve the problem.	18

Table 1: Annotated categories for posts inside threads.

Table 2 shows the Kappa statistics on thread and post annotation using 20 threads that we randomly selected and let both annotators label them. For the thread annotation agreement, we use binary labels – usefulness greater than 3 is considered as 1, and usefulness less than 3 is considered as 0. The Kappa scores are very reasonable, suggesting that humans do not have much trouble with this task definition.

Classes	Kappa Coefficient
Thread Usefulness	0.93
Post - Problem	0.86
Post - Solution	0.70
Post - Good Feedback	0.93

Table 2: Kappa statistics between two annotators for thread and post category annotation.

3 Approaches

For the ultimate goal of classifying how likely a thread provides a good solution (its usefulness), we propose to use a two-step approach. First we determine the categories for the posts, then we classify the threads using information from the first step along with other information.

3.1 Post Level Classification

We use the content of a post as well as its context for post classification. First we used Naive Bayes classifier with a bag-of-word model for post classification. Naive Bayes classifiers have been proved to be robust and perform well in document classification. After performing tokenization using space, we use the top 600 words from the training set as the dictionary.[2] Each post is thus represented by a feature vector of tokens that appear in the dictionary. We use binary features for these words. This has been shown to perform better than using word frequency information for many classification tasks. We use the implementation of multinomial Naive Bayes classifier provided by Weka (Hall et al., 2009).

We expect that posts in a thread, like turns in dialogs, are very dependent on the context. For example, a “problem description” post is more likely to be followed by “solution” posts than a “feedback” one. People are more likely to give “feedback” after some “answers” are provided rather after a “question” is asked. Hence we evaluate using an HMM for this post categorization task, in order to take advantage of context information. Figure 1 shows a first order HMM for this setup.

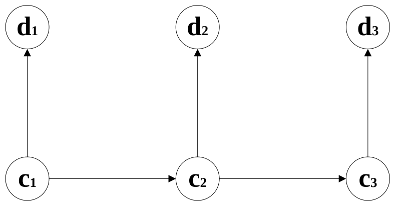
$$\begin{tikzcd} d_1 & d_2 & d_3 \\ c_1 \arrow[u] \arrow[r] & c_2 \arrow[u] \arrow[r] & c_3 \arrow[u] \end{tikzcd}$$

Figure 1: Illustration of HMM for post categorization. An observation d_i is a post, and c_i corresponds to its category.

The generative process is as follows. The category of current post is generated according to a multinomial distribution $P(c_i|c_{i-1})$ conditioned on its previous post category. Then, given the category of current post, words in the post are generated according to multinomial distribution $P(d_i|c_i)$ based on the post's category.

The problem can be formulated to find the most likely state sequence C (post categories) given all

[2] We varied the dictionary size, but observed performance degradation.

the words in all the posts D. The posterior probability of C given D is:

$$\begin{aligned} P(C|D) &= P(c_1, .., c_n | d_1, .., d_n) \\ &\propto P(d_1, .., d_n | c_1, .., c_n) \times P(c_1, .., c_n) \\ &= \prod_{i=1}^{n} P(d_i|c_i) \times P(c_i|c_{i-1}) \end{aligned} \quad (1)$$

The last equation is due to the first order HMM assumption where a post document is only dependent on its category (in a generative process), and a post category is only dependent on its previous category. For parameters in this HMM, the transition probabilities are estimated from the training set, similar to training a bigram language model (LM) for the post categories.

It is possible to use an n-gram language model to model the generation of text in posts given its category. However, since our data set is rather small, there is a data sparsity issue for these n-gram LMs. Hence, we only use a unigram-like model for a subset of the words (same as those in the Naive Bayes model), and each word is represented in binary form: whether or not it appeared.

In practice, when combining the state transition probabilities and emission probabilities, we use a weighting factor γ that is determined empirically. This is needed because the two scores are in different scales and contribute to the final hypothesis differently.

3.2 Thread Level Classification

The second step in our system is thread level classification, where the system determines whether a thread is useful or not. Using our annotated threads, we grouped them into binary classes: useful when the label is equal to or greater than 3; and not useful otherwise. We use a variety of features for thread level classification. Table 3 lists all the features we use for thread classification.

Since the post level classifier is optimized for the classification accuracy for that particular task, its hypotheses might not be optimal for subsequent thread classification. For example, if a post "looks" like both a "good feedback" and a "problem", we may prefer "good feedback" to "problem" since the former is more related to our final decision for the thread usefulness. To achieve this, we adjust the priors in the post classification process for different categories. We found that by increasing the prior for "good feedback" by 1.5 times, the final performance is improved – even though the classification precision decreases, the improved recall seems to be helping.

Name	Description
num_solution	The number of solution posts
feedback_p	Number of positive feedbacks
num_prob_sol	Times of back and forth between problem and solutions
author_post	Number of posts posted by author
author_postend	How far the author's last post is from the end
length	Length of a thread (measured using number of posts)

Table 3: Features used for thread level classification. 'Author' means the user who first posted the problem and started the thread.

We use a maximum entropy model for thread classification. This classifier is chosen because of its good performance in many language processing tasks, as well as our own preliminary experiments when comparing to other classifiers including decision trees and SVMs.

4 Experiments

4.1 Post Level Classification Results

We performed leave-one-out-cross-validation (using threads as the units) for post classification. Table 4 shows the results using different methods. We developed a rule-based baseline system for this task. Posts containing typical question words like "what","where" are classified as "problem" posts. Those containing cue words for positive feedbacks, like "thanks", "solved", and having length of less than 50 words are classified as "positive feedback" posts. The rest are classified as solutions.

The results shown in the table include the F_1 score for 3 individual post categories: problem (F-P), solution (F-S), and good feedback (F-F), as well as the micro averaged F_1 score. Here we ignored the classification results for "bad feedback" category since it is not very useful in later thread classification.

We can see that the basic Naive Bayes (NB) classifier can achieve reasonable performance already. It is significantly better than the baseline. This shows that lexicon features (bag-of-word model) are strong and reliable in classifying post type. We also tried using an SVM (with linear kernels) for post level classification, but its performance is worse than the Naive Bayes clas-

Method	F_{micro}	F_1-P	F_1-S	F_1-F
Baseline: rule	44.58	49.59	46.53	18.58
NB	77.57	76.81	85.52	28.57
HMM	83.17	83.21	87.75	36.04

Table 4: Post classification results using different methods. Results are the F-measures for problem, solution, and good feedback categories, and the overall performance.

sifier. After using context information through HMM, post level classification performance increased substantially. There is consistent improvement for all the categories. In particular, there is a relatively larger gain for the "feedback" class, which we expect may have a great impact on subsequent thread level classification. This performance gain using HMM demonstrates that contextual information is useful and HMM can well model such information (e.g., an 'answer' post is likely to follow a 'problem' one). We also increased HMM to higher order in order to model the transitions using more previous post categories, but found there is no additional improvement. In fact, it is slightly worse than using the first order HMM. This can be explained by either that the long distance relationship is not very crucial for this task, or more likely that we do not have a large training set to learn high order transition information reliably.

In general, we can see that the classifiers perform relatively well for the majority categories, such as problems and solutions. However, for minority classes, e.g., "feedback" (that is only 9.8% among all the posts), the classifiers are not able to learn well. We also observe that many "problem" types are classified as "solutions". This may be explained by two reasons. First, problem description posts use very similar vocabulary as solution posts. Second, there are more solution posts in the training data.

4.2 Thread Level Classification Results

For thread level classification, we use a binary setup, useful vs. not. We used leave-one-out-cross-validation for the 200 labeled threads. Inside each fold, first we use the training set to train the post level classifiers. Then we relabel all the posts in the training set as well as in the testing set with the classifier just trained to obtain the post category hypotheses. After the post level labeling, we extract features for thread classification as described in Section 3. We then train the maximum entropy classifier and test it for the final classification of threads. We use the precision, recall, and F_1 score as the evaluation metrics. Results are shown in Table 5. For a comparison, we also show the performance when using reference post labels for both training and test sets (last row). This is expected to give an upper bound performance regarding the use of post level information for thread classification.

Post Classifier	Precision	Recall	F_1
Baseline: rule	68.50	97.16	80.35
NB	78.21	86.52	82.15
HMM	79.14	91.32	84.97
Ref	84.62	93.62	88.89

Table 5: Thread level binary classification results using features extracted from output of different post level classifiers.

We can see from Table 5 that in general thread classification results depend heavily on the performance of post level classification. HMM achieves the best performance. Using reference post category shows the upper bound of how post level classification could affect thread level classification. We also evaluated using the reference post tags in training the thread classifier, and automatic tags for testing, but that did not perform as well as using automatically obtained tags for both training and testing, suggesting a matched training and testing condition is better. Overall we achieved very good performance – an F_1 score of about 85% using HMM for post categorization. This is a promising result given that we have a quite small data set for training.

5 Conclusion

In this paper, we described a two-step classification approach to determine whether a thread is helpful to users seeking solutions. We perform post level classification in the first step, then use the generated post tag information with other global features for thread level classification. We showed in the experiment that our approach worked well in a technical support online forum. For future work, we plan to use joint optimization for the two tasks. In addition, instead of using predefined categories for posts, we may derive post categories automatically from the corpus that suit better for the thread classification task.

References

Gao Cong, Long Wang, Chin-Yew Lin, Young-In Song, and Yueheng Sun. 2008. *Finding question-answer pairs from online forums*. In *SIGIR*, pages 467–474.

Shilin Ding, Gao Cong, Chin-Yew Lin, and Xiaoyan Zhu. 2008. *Using conditional random fields to extract contexts and answers of Questions from online forums*. In *ACL*.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The weka data mining software: an update. *SIGKDD Explorations*, 11(1):10–18.

Jizhou Huang, Ming Zhou, and Dan Yang. 2007. *Extracting Chatbot Knowledge from Online Discussion Forums*. In *SIGIR*, pages 423–428.