Identifying the Role of Individual User Messages in an Online Discussion and Its Use in Thread Retrieval

Sumit Bhatia*
IBM Almaden Research Centre, 650 Harry Road, San Jose, CA 95120. E-mail: sumit.bhatia@us.ibm.com

Prakhar Biyani and Prasenjit Mitra
College of Information Science and Technology, Pennsylvania State University, IST Building, University Park, PA 16802. E-mail: prakharbiyani@gmail.com; pmitra@ist.psu.edu

Online discussion forums have become a popular medium for users to discuss with and seek information from other users having similar interests. A typical discussion thread consists of a sequence of posts posted by multiple users. Each post in a thread serves a different purpose providing different types of information and, thus, may not be equally useful for all applications. Identifying the purpose and nature of each post in a discussion thread is thus an interesting research problem as it can help in improving information extraction and intelligent assistance techniques. We study the problem of classifying a given post as per its purpose in the discussion thread and employ features based on the post's content, structure of the thread, behavior of the participating users, and sentiment analysis of the post's content. We evaluate our approach on two forum data sets belonging to different genres and achieve strong classification performance. We also analyze the relative importance of different features used for the post classification task. Next, as a use case, we describe how the post class information can help in thread retrieval by incorporating this information in a state-of-the-art thread retrieval model.

Introduction

Online discussion forums have recently become popular because they provide an easily accessible platform to users in different parts of the world to come together, share information, and discuss issues of common interest. Thousands of web forums devoted to a multitude of topics exist where millions of users regularly participate in various discussions. People use web forums to discuss and ask questions about various topics such as news, sports, technology, health, and so on. The archives of web forums contain millions of such discussion threads and act as a valuable repository of human generated information that needs to be managed efficiently.

A typical discussion thread in a web forum consists of a number of individual *posts* or messages posted by different participating users. Often, the thread initiator posts a question to which other users reply, leading to an active discussion. Different participants in the thread may offer possible solutions to the topic initiator's problem, ask for details, provide feedback about the proposed solutions, and so on. As an example, consider the thread shown in Figure 1 where the thread starter describes his problem about the missing headphone switch in his Linux installation. In the third post in the thread, another user asks for some clarification, and in the next post the topic starter provides the requested details that makes the problem more clear. On receiving additional details about the problem, another user provides a possible solution to the problem (fifth post). The topic starter tries the suggested solution and reports his experience in the next post (sixth post). Thus, we see that each individual post in a discussion thread serves a different purpose in the discussion and we posit that *identifying the purpose of each such post is essential for intelligent and effective utilization of the information contained in the thread*. Even though there have been efforts to develop customized retrieval models for searching discussion threads (Bhatia & Mitra, 2010; Elsas & Carbonell, 2009; Xu & Ma, 2006) and extracting useful information such as question–answer pairs from discussion threads (Cong, Wang, Lin, Song, & Sun, 2008; Ding, Cong, Lin, & Zhu, 2008; Hong & Davison, 2009), and so on, to the best of our knowledge, there has been no work that utilizes the different nature of individual user posts in a discussion thread.

*The work was performed in its entirety while Sumit Bhatia was a graduate student at Penn State.

Received January 28, 2014; revised June 22, 2014; accepted June 23, 2014

 DOI: 10.1002/asi.23373

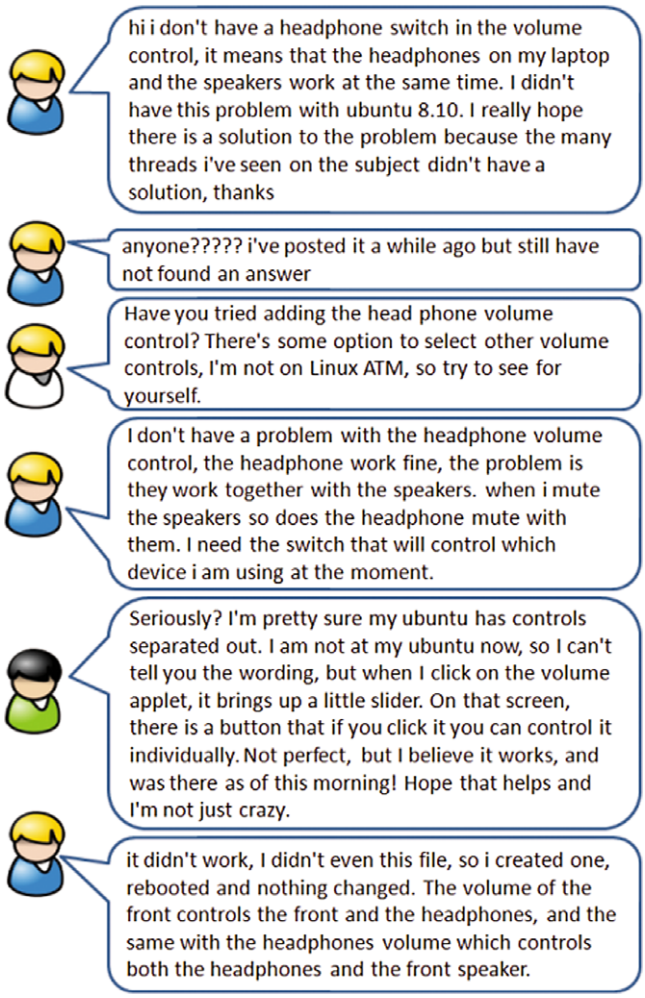

FIG. 1. An example thread illustrating different roles played by each post in the discussion. Different users are indicated by different colors. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

Why Identifying the Role of Each Post Is Essential

Identifying the role of each individual post in the discussion thread can be useful for various applications as we discuss below.

1. Systems for searching web forums can utilize this information for thread ranking. Threads containing solutions to a given problem can be assigned a higher weight than the threads that do not have a solution post. Likewise, threads that contain posts providing a positive feedback about the solutions proposed in the thread can be ranked higher than the threads that have no feedback information or that contain negative feedback posts.
2. Classifying forum posts according to their role can be utilized for assessing user roles in the discussions (e.g., finding *solution providers* or *experts*, identifying spammers, etc.). It can also be utilized for improving information extraction and intelligent assistance techniques (Kim et al., 2006) as well as for question-answer detection algorithms (Feng, Shaw, Kim, & Hovy, 2006).
3. Knowing the role and importance of different posts in a given thread is also useful for summarizing a discussion thread. For example, a concise summary of the thread can be constructed by using only the posts in which the question is being asked and the posts in which the solutions are provided. Zhou and Hovy (2006) discuss challenges in summarizing dynamically created textual information (as is the case with online discussion forums) and argue that identifying text segments (posts in case of forum threads) belonging to different categories (e.g., question, answers) is essential for creating effective summaries.
4. Usually threads in a web forum are displayed to users sorted by the time of posting of last message in the thread. Instead, an alternative scheme could be to present threads with unresolved questions first. This scheme can be useful especially for technical forums where people ask a lot of questions. Experienced users in the forum who generally provide answers to many questions (Bhatia & Mitra, 2010) can then easily find threads with unanswered questions and provide the necessary information. Further, this information can also be useful for thread recommendation systems that recommend threads to users for participation (Zhao et al., 2010).

Our Contributions

In this article, we build on our previous work (Bhatia, Biyani, & Mitra, 2012) addressing the problem of *classifying individual posts per their role or purpose in the discussion thread*. Our major contributions are as follows:

1. For post classification, we propose features that utilize content of the post, structure of the discussion thread and properties of the participating users. Further, we also employ features based on sentiment analysis of the post text.
2. We evaluate the effectiveness of the proposed features on data set derived from two real world forums belonging to different genres and achieve strong classification results using the proposed feature set. We also study the relative importance of individual feature types and analyze their performance on two different data sets.
3. As a use case for demonstrating the utility of identifying roles of user posts, we incorporate post class label information in a state-of-the-art forum thread retrieval model and achieve statistically significant improvements in retrieval performance as measured by different metrics.

Related Work

There exists a large body of work dealing with various problems related to web forums. Here, we provide a brief survey of the representative literature that is most related to our work.

Information Extraction From Online Forums

Yang et al. (2009) use the linkages and relationships between pages in an online forum site to extract structured metadata such as post title, post content, and so on. Huang,

DOI: 10.1002/asi

Zhou, and Yang (2007) extracted high-quality replies in forum threads. They first identified replies relevant to the initial post (of the thread) using SVM classifier and then ranked the identified replies using ranking SVM. Forums have also been used as a data source for question answering systems. Cong et al. (2008) proposed techniques to extract question answer pairs from online forums. Their question detection algorithm used sequential pattern features called labeled sequential patterns as features to distinguish between question and nonquestion sentences. Their answer extraction algorithm ranked the posts in a forum thread based on similarity with questions and user information to output a ranked list of candidate answers. Building on this work and using the same question detection approach, Ding et al. (2008) used conditional random fields to identify relationships between different posts in a thread to extract context and answers of the questions posed in a single thread. The model was improved by Yang, Cao, and Lin (2011) who used a more generalized graphical representation using structural support vector machines to model dependencies between question, answer, and context sentences more effectively. Hong and Davison (2009) described a classification-based approach for detecting whether the first post of a thread is a question and then finding the potential answer post from the remaining posts in the thread. A translation language model and query likelihood-based retrieval model for question answer archives was proposed by Xue, Jeon, and Croft (2008). Wang, Tu, Feng, and Zhang (2009) used analogical relationships between questions and answers for ranking answers in community question answer. A general ranking framework for factual question answering was discussed by Bian, Liu, Agichtein, and Zha (2008). There also have been works on modeling forum thread structure to extract useful information such as reply links between posts, type of reply links, and so on. Lin, Yang, Cai, Wang, and Wang (2009) modeled structural and semantic relationships between posts to identify reply relationships between posts, junk posts, and expert users in a thread. Wang, Wang, Zhai, and Han (2011) improved upon the previous work by simultaneously modeling dependencies between all the posts in a thread using conditional random fields. Biyani, Caragea, Singh, and Mitra (2012b) analyzed and predicted subjectivity orientation of online discussion threads by utilizing lexical properties of the discussion text. They further improved the accuracy of proposed models by incorporating dialog act features and structural properties of discussion threads (Biyani, Bhatia, Caragea, & Mitra, 2014).

Forum Post Classification

The most similar work to our post classification work is that of dialogue act classification in natural language processing where the purpose is to classify different utterances according to their role or purpose in a conversation (Murray, Renals, Carletta, & Moore, 2006). Dialogue act classification can be performed for spoken conversation (e.g., work by Stoicke et al. [2000]) as well as written conversation, the latter being similar in nature to our research.

Cohen, Carvalho, and Mitchell (2004) classified e-mail messages according to the purpose of the e-mail message in a business setting. They identified a set of *e-mail verbs* (e.g. request, deliver, propose, commit, etc.) and used text classification methods to detect if a given e-mail message contains a specific e-mail verb. Lampert, Dale, and Paris (2008) propose a well-grounded set of definitions for *requests* and *commitments* in e-mail based on manual annotation experiments carried out with the Enron e-mail corpus. Jeong, Lin, and Lee (2009) and Joty, Carenini, and Lin (2011) used supervised and unsupervised machine learning techniques, respectively, to identify 12 types of dialogue acts in sentences of e-mails and forum posts. The work on dialogue act tagging for online forums by Kim, Wang, and Baldwin (2010) online forums is most similar to our work. Their work focused on uncovering the thread content structure in the form of post-post linkages, that is, identifying the (previous) posts in a thread to which a post responds to and the type of relationship between the linked posts. On the other hand, our focus is on identifying the role each post plays in the overall discussion going on in the thread.

Forum Thread Retrieval

There also have been few works to develop customized retrieval models for searching web forum data. Elsas and Carbonell (2009) were among the first to review strategies for thread retrieval on a test collection of 48 < query, relevant document > pairs. Seo, Croft, and Smith (2009) described how the reply structures in forum threads can be recovered and utilized for thread and post level retrieval. Bhatia and Mitra (2010) used inference networks for combining evidences from different structural units of a thread. They also explored the effect of incorporating various nontextual relevance indicators that could help in an improved ranking of returned results. Duan and Zhai (2011) exploit contextual information of a post for post retrieval. They smoothed a post language model with related posts in the thread thus taking into account the context of the post in retrieval. To the best of our knowledge, our work is the first to use information about the post's role in the discussion for thread retrieval.

Description of Classes

We classify a given user post in a discussion thread into one of the following eight classes according to its role in the discussion thread.

1. Question: The poster asks a question which initiates discussion in the thread. This is usually the first post in the thread but not always. Often, the topic initiator or some other user may ask other related questions in the thread.
2. Repeat Question: Some user repeats a previously asked question (e.g., *Me too having the same problem.*).

DOI: 10.1002/asi

TABLE 1. An example thread with class labels for the posts. Note that the first row contains the thread title, which is included to provide the context for the discussion going on in the thread. Thread title is not one of the target classes.

Class	Post content
Thread title	Newly opened windows hide behind the Gnome panel :(
Question	When I open a new window, it opens at the top left corner. The top of the window hides behind the top panel so that I have to *<Alt + left>* button to move the window first. I am using Compiz-Fusion in Gutsy. Does anybody know of a fix for this? Thank you!
Junk	anybody?
Solution	I'm pretty sure in the Compiz Manager there's a setting that changes where windows open on the screen. Place windows, I believe.
+ve Feedback	You're right! Thank you!!!

3. Clarification: The poster asks clarifying questions to gather more details about the problem/question being asked. For example, *Could you provide more details about the issue you are facing*.
4. Further Details: The poster provides more details about the problem as asked by other fellow posters.
5. Solution: The poster suggests a solution to the problem being discussed in the thread.
6. Positive Feedback: Somebody tries the suggested solution and provides a positive feedback if the solution worked.
7. Negative Feedback: Somebody tries the suggested solution and provides a negative feedback if the solution did not work.
8. Junk: There is no useful information in the post. For example, someone justs posts a smiley or some comment that is not useful to topic being discussed. For example, "*bump*," "*sigh*," etc., or messages posted by forum moderators such as *this thread is being closed for discussion*.

Table 1 shows an example thread with each post of the thread labeled with the appropriate class label. The first seven classes above were described in the Mailing List and Forum (MLAF) track at Forum for Information Retrieval Evaluation[1] (FIRE). We added the eighth class as we observed that a significant number of posts in the threads do not contain any useful information and it is essential to identify such posts. Also, note that even though the MLAF task provides a test data set, we chose not to use it as it did not have the labels for the *junk* class. Further, the data set provided a random set of user posts from discussion threads and their respective class labels. We however, wanted to experiment with certain user level and thread level features (see Feature Description) and the thread level and user level information was not available in the provided data set. Hence, we created our own data set for experiments in this article.

[1]http://www.isical.ac.in/ fire/

Feature Description

We use a variety of features for classifying forum messages into the eight classes as described previously. Table 2 lists all the features used in this work and in the following subsections we describe the motivation behind using the said features.

Content-Based Features

The content of the post should be a good indicator of the nature of the post. For example, we expect the posts that answer the questions and issues raised in the initial post to have a higher similarity with the title of the thread and the initial post. On the other hand, we expect that *off-topic* posts to have lower similarity scores. Based on these considerations, for each post we use as features the cosine similarity scores with the thread title, the initial post of the thread and the whole thread. In addition to content similarity, the presence of question marks and any of the 5W1H question words (what, where, why, when, who, how) hints that a question is being asked in the post. Likewise, if one of the previous posts is being quoted in the post,[2] it is often the case that the user is responding to the quoted post.

User Features

Different users in a forum exhibit different message posting behavior. Although some of the more experienced and knowledgeable users act as *information providers* and answer many questions, there are many users who mainly act as *information seekers* asking for solutions to their problems. Hence, intuitively the class of a post should have some dependency on the user. To capture this dependency, for each post we use as features the authority score of the poster (Bhatia & Mitra, 2010), total number of messages posted by the user, and whether the user is the thread initiator. The authority score (Bhatia & Mitra, 2010), $A(u)$ of user u is defined as:

$$A(u) = \lambda_{auth}\left\{\frac{N_p(u) - N_{ip}(u)}{N_p} + \frac{1}{N_u}\right\} \quad (1)$$

where,

$N_p(u)$ is the total number of posts by user u,
$N_{ip}(u)$ is the total number of thread starting posts (first post in a thread) by user u,
N_p is the total number of posts in the collection,
N_u is the total number of users, and
λ_{auth} is a normalizing constant

In the previous equation, the first term inside the bracket measures the contribution of the user to all the replies in the collection. The intuition behind this is that an *information*

[2]Identified by "Quote" box in the threads used in this work. Different forum software provide different mechanisms to quote a post.

DOI: 10.1002/asi

TABLE 2. Description of various features used for the classification task.

Feature name	Description	Type
Content-based features		
IsQuote	Does the post quote a previous post? 1 if yes, 0 otherwise.	Binary
TitleSim	Cosine similarity between the post and thread title.	Real
InitSim	Cosine similarity between the post and first post of the thread.	Real
ThreadSim	Cosine similarity between the post and entire thread.	Real
QuestionMark	Does the post contain a question mark.	Binary
Duplicate	Does the post contain the keywords *same*, *similar*.	Binary
5W1H	Does the post contain a word from {what, where, when, why, who, how}.	Binary
Structural features		
AbsPosition	Absolute position of a post in the thread.	Numerical
NormPosition	Normalized position of a post in the thread. Computed as (Absolute Position/Total number of posts in the thread).	Real
PostLength	Total number of words in a post after stopwords removal.	Numerical
PostLengthUnique	Unique number of words in a post after stopwords removal.	Numerical
PostLengthStemmed	Total number of words in a post after stopwords removal and stemming.	Numerical
PostLengthUniqueStemmed	Unique number of words in a post after stopwords removal and stemming.	Numerical
User features		
UserPostCount	Total number of messages posted by the poster.	Numerical
IsStarter	Is the post made by the topic starter? 1 if yes, 0 otherwise.	Binary
UserAuth	Authority score (Bhatia & Mitra, 2010) of the poster.	Real
Sentiment-based features		
Thank	Does the post contain the keyword *thank*.	Binary
ExclamationMark	Does the post contain an exclamation mark.	Binary
–ve Feedback	Does the post contain the keywords *did not*, *does not*.	Binary
SentimentScore	Sentiment scores of the post as computed by SentiStrength (Thelwall et al., 2010). (Four features, see text for details.)	Numerical

provider or an *expert* asks less questions and answers others' questions more when compared with an *information seeker* or a *novice* user. The second term acts as a smoothing parameter and assigns a uniform default authority to each user in the collection.

Structural Features

We expect the location and position of the post in the thread to be an indicator of the class of the post. For example, ideally the problem being discussed is described in the first post of the thread and the clarifying questions and details are generally being discussed in first few posts whereas the posts containing the solutions and user feedback should be the last few posts of the thread. Hence, we use the absolute and relative positions of the post in thread as features. Similarly, we expect the posts where the problem and the solutions are being discussed to be longer than the posts where the feedback is provided. Hence, length of the post is also an important feature and we use four different versions of this feature (ref. Table 2). While computing these features, stemming was performed using Porter's stemmer (Porter, 1980) and stop words were removed using a general stop word list of 429 words used in the Onix Test Retrieval Toolkit.[3]

Sentiment Features

These features try to capture the emotion and sentiment of the post. We expect the posts in which the users describe their problems and the posts where a negative feedback to a suggested solution is provided to be of a negative tone. Likewise, the posts where users suggest a solution to the problem being discussed in the thread as well as the posts where positive feedback is provided should have a positive tone. We compute sentiment strength scores for each post using the SentiStrength algorithm as described by Thelwall, Buckley, Paltoglou, Cai, and Kappas (2010). We use the implementation of the algorithm as available from http://sentistrength.wlv.ac.uk/. SentiStrength algorithm is specifically developed for sentiment detection and extracting sentiment strengths from short informal texts such as forum posts, blog comments, and so on. The method takes into account the grammar and spelling conventions (e.g., terms such as LOL, OMG, bcoz, etc.) that are common in cyberspace and computes positive and negative sentiment strength scores for a given piece of text using a set of rules and language patterns associated with the sentiment. In a given piece of text, there can be multiple word patterns/rules indicative of positive or negative sentiments. The algorithm computes two versions of positive and negative sentiment strength scores for each piece of text: (a) using the strongest indicative word patterns and (b) using all the indicative word patterns and taking their average. Thus, we get four different

[3]http://www.lextek.com/manuals/onix/stopwords1.html

DOI: 10.1002/asi

sentiment strength scores for each post. In addition to sentiment strength scores, we also use the presence of emotion indicating punctuation such as exclamation marks and presence of keywords such as "thank," and so on.

Using Post Class Labels for Improving Thread Retrieval

As discussed in the Introduction, classifying user posts according to their role in the discussion can be useful in various applications. Here, we describe how the post class information can be utilized in thread retrieval systems. We use a state-of-the-art probabilistic model for forum thread retrieval (Bhatia & Mitra, 2010) as our baseline and incorporate post class label information in the model to see if it helps improve retrieval performance.

The baseline model (Bhatia & Mitra, 2010) that we use is a probabilistic model based on inference networks that utilizes the structural properties of forum threads. Given a query Q, the model computes $P(T|Q)$, the probability of thread T being relevant to Q, as follows:

$$P(T \mid Q) \overset{rank}{=} P(T)\prod_{i=1}^{n}\left\{\sum_{j=1}^{m}\alpha_j P(Q_i \mid S_{j_T})\right\} \quad (2)$$

where:

Q_i is the i^{th} term in query Q,
S_{jT} is the j^{th} structural unit in the thread T,
α_j determines the weight given to component j and $\sum_{j=1}^{m}\alpha_j = 1$.

To estimate the likelihoods $P(Q_i|S_{j,T})$ we use the standard language modeling approach in information retrieval (Ponte & Croft, 1998) with *Dirichlet Smoothing* as follows:

$$P(Q_i \mid S_{j_T}) = \frac{f_{Q_i,j_T} + \mu\frac{f_{Q_i,j_C}}{|j|}}{|j_T| + \mu} \quad (3)$$

Here,

f_{Q_i,j_T} = frequency of term Q_i in j^{th} structural component of thread T,
f_{Q_i,j_C} = frequency of term Q_i in j^{th} structural component of all the threads in the collection C,
$|j_T|$ is the length of j^{th} structural component of thread T,
$|j|$ is the total length of j^{th} structural component of all the threads in the collection C,

μ is the Dirichlet smoothing parameter. In this article, we set μ to be equal to 2,000, a value that has been found to perform well empirically (Zhai & Lafferty, 2001). Thus, the model computes the overall probability of a thread being relevant to the query by combining evidence from different structural units of the thread. Three different structural units of the model are considered—thread title, thread's initial post, and the set of follow-up reply posts.

Incorporating Post Class Label Information in the Retrieval Model

The component $P(T)$ in Equation 2 represents the prior probability of a thread being relevant. We incorporate the post class label information as prior probabilities in the model. We describe three different priors below and experiment by adding each prior one at a time and then by adding different prior combinations (refer to Retrieval Experiments—Does Post Class Information Help Improve Thread Retrieval? section for results).

Intuitively, in the absence of any other information about the thread's content, we expect that threads that contain a solution post will have a higher probability of successfully satisfying the user's information need as compared to threads that do not have any solution post. Likewise, threads containing positive feedback posts are more important than the threads containing negative feedback posts as it indicates that in the former case, the solutions provided in the thread were helpful for previous users whereas in the latter case, the solutions were not that useful. Motivated by these considerations, we define the following three priors:

Solution Prior (S):

$$P(T) = \frac{\text{No. of solution posts in the thread} + 1}{\text{Total no. of solution posts in all threads} + N_T} \quad (4)$$

Positive Feedback Prior (P):

$$P(T) = \frac{\text{No. of + ve feedback posts in the thread} + 1}{\text{Total no. of + ve feedback posts in all threads} + N_T} \quad (5)$$

Negative Feedback Prior (N):

$$P(T) = \frac{1 - \frac{\text{No. of - ve feedback posts in the thread} + 1}{\text{Total no. of - ve feedback posts in all threads} + N_T}}{N_T - 1} \quad (6)$$

The above equations assign a higher weight to threads having a higher number of solution and positive feedback posts. Further, note the different nature of equation describing negative prior. This is because for negative prior, we want to assign a *lower* weight to threads having more negative feedback posts. Further, note that in above equations, one has been added in the numerator for smoothing probability values so that a zero probability value is not assigned to threads that do not have a solution or a feedback post. The denominator is the constant normalizing factor such that $\Sigma P(T)$ over all the threads is one. N_T in denominator denotes the total number of threads in the corpus. We also note that there may be other ways of using the post class label information for thread retrieval. However, the focus of this work

DOI: 10.1002/asi

TABLE 3. Summary statistics of different data sets used.

	Ubuntu	NYC
Total no. of threads	113,277	83,072
Total no. of users	103,280	39,454
Total no. of posts	676,777	590,021
Average thread length (in no. of posts)	5.98	7.10

is to show that post class label information can be utilized for improving thread retrieval. Finding the retrieval model that utilizes the post class labels in the most effective manner is an independent research problem that is left for future work.

Experiments

Data Description

For the experiments in this article, we use the same data set as used in our previous research (Bhatia & Mitra, 2010). The data set consists of threads crawled from two online forums—(a) official forum of the Ubuntu Linux distribution[4] and (b) discussion forum about New York city from Trip Advisor forum.[5] The data set consists of crawled threads from the two forums, a set of 25 queries for each forum and associated relevance judgments. Because the relevance judgments were not available for the two data sets used, we used the help of two human annotators to create relevance judgment pools for the two collections. The evaluators were asked to assign ternary relevance judgments to each thread for a given query—0 for totally irrelevant threads, 1 for partially relevant, and 2 for highly relevant threads. They were asked to consider a thread relevant if the discussions in the thread are about the given query topic. The search page of the Ubuntu forums provides a list of 70 most searched for terms that were used to generate queries for our experiments. Similarly, the TripAdvisor forum provides a list of most frequently searched for topics. The queries for this data set were generated by extracting keywords from the frequently searched for topics. In total, we generated 25 queries for each data set. In all, relevance judgments were assigned for 4,512 threads in the Ubuntu data set and 4,478 threads in the TripAdvisor data set. Table 3 summarizes the statistics about the data set.

For performing classification experiments, we randomly sampled 100 threads from each of the two forums. There are a total of 556 posts in the 100 threads from the Ubuntu data set and 916 posts in 100 threads from NYC data set. To obtain class labels for the posts, we recruited three human evaluators. All the three evaluators were senior year undergraduate students in computer science, native English speakers, and well versed with the Ubuntu operating system. The evaluators were provided with class definitions and were asked to assign the most suitable class label to each post. The evaluators worked independently of each other and assigned class labels to each post in the 200 threads (100 each for Ubuntu and NYC data set). The final class label of each post was then decided by the majority vote. Of 556 posts for the Ubuntu data set, a majority decision was achieved for 529 (95.14%) posts. The Fleiss' Kappa statistic (Fleiss, 1971) used for measuring inter-annotator agreement when more than two annotators are involved was 0.7884, indicating substantial agreement. The remaining 27 posts for which all the three evaluators assigned different class labels were discarded. For the NYC data set, a majority decision was achieved for 884 posts of 916 posts (96.51%) and the Fleiss' Kappa statistic was 0.7928, indicating substantial agreement. Table 4 summarizes the distribution of posts in different classes in the two data sets.

TABLE 4. Distribution of different classes in the two data sets.

	Number of instances	
Class	Ubuntu	NYC
Question	134	135
Repeat Question	17	2
Clarification	29	61
Further Details	55	54
Solution	207	457
Negative Feedback	11	7
Positive Feedback	25	114
Junk	51	54
Total	529	884

Experimental Protocol

For classification experiments, we tried a variety of supervised machine learning algorithms like support vector machine, Naive Bayes classifier, decision trees, multilayer perceptron and the logit model classifier. The experiments were performed using the Weka data mining toolkit (Hall et al., 2009). The performance of all the classifiers was comparable with the logit model achieving the best classification performance. We use tenfold cross validation to optimize the logit model parameters. For classification results, we report overall classification accuracy and for a detailed analysis, we use macro-averaged precision, recall, and the F-1 measure. For a metric M, macro-average M_{mav} is calculated by taking weighted average of M for the different classes for each fold and then taking mean of the weighted averages across all the folds. For N-fold cross validation and C class classification, M_{mav} is mathematically defined as follows:

$$M_{mav} = \frac{1}{N}\sum_{i=1}^{N}\frac{\sum_{c=1}^{C} n_{ci}M_{ci}}{\sum_{c=1}^{C} n_{ci}} \quad (7)$$

[4]http://ubuntuforums.org

[5]http://www.tripadvisor.com/ShowForum-g28953-i4-New_York.html

DOI: 10.1002/asi

TABLE 5. Classification results for the rule based, bag of words (BoW) and proposed classification approach.

Ubuntu			
Metric	Rule based	BoW	Proposed approach
Classification accuracy	58.03%	57.66%	72.69%
Precision	0.442	0.503	0.705
F1-measure	0.471	0.473	0.712
NYC			
Classification accuracy	61.88%	60.98%	75.11%
Precision	0.441	0.596	0.726
F1-measure	0.499	0.529	0.724

where n_c is the number of instances belonging to class c in the test set in the i^{th} fold. M_{ci} is the value of metric M for class c in the i^{th} fold. In our case, $N = 10$ and $C = 8$.

We compare the performance of the proposed approach with the following two baseline classification approaches:

1. Rule Based: We build a naive, rule based classifier that marks the first post in a discussion thread as a question post and labels all the remaining posts as solutions.
2. Bag of Words: We use a bag of words model that tries to capture lexical properties of the text to be classified. For this, we use a Naive Bayes Multinomial classifier that uses frequency of words in a post as features for classification. Bag of Words based classifiers are frequently used for a variety of text classification tasks (Aikawa, Sakai, & Yamana, 2011; Biyani, Bhatia, Caragea, & Mitra, 2012a; Biyani, Caragea, & Mitra, 2013a; Biyani et al., 2013b; Li, Liu, & Agichtein, 2008; Yu & Hatzivassiloglou, 2003).

For retrieval experiments, we used the Indri language modeling toolkit.[6] While indexing, stemming was performed using Porter's stemmer (Porter, 1980) and the same stopword list as described in Structural Features. The queries and relevance judgments available with the data set as discussed in the previous subsection were used for retrieval experiments. For the baseline retrieval model, we used the optimal parameter settings as used in the original work (Bhatia & Mitra, 2010). In order to compare the performance of various retrieval methods, we report Mean Reciprocal Rank (MRR), Precision @ 5, Precision @ 10, NDCG @ 10 and Mean Average Precision (MAP). To assess statistical significance of obtained results, we use Wilcoxon's signed rank test with a confidence interval of 95%, $p < 0.05$.

Results and Discussions

Forum Post Classification Results

Table 5 reports the results of forum post classification using the logit model classifier as well as the two baseline classifiers. We achieved an overall classification accuracy of 72.69% with a precision of 0.705 and F-1 measure of 0.712 for Ubuntu data set and accuracy of 75.11%, precision of 0.726 and F-1 measure of 0.724 for NYC data set. Thus, we see that we obtain very similar classification results on the two data sets using our proposed feature set. We also note that the proposed classifier significantly outperformed the two baseline classifiers across all the metrics. For further analysis of the performance of our proposed classifier, Table 6 summarizes the individual results for each of the eight classes for the two data sets. We report precision, F-1 measures, true positive and false positive rates, and the area under the ROC curve. We observe that for both the data sets, the classifier is able to detect posts belonging to *question* and *solution* categories with a very high accuracy. For the Ubuntu data set, the F-1 measures for question and solution classes are 0.867 and 0.810, respectively. For NYC data set, the F-1 measures for the two classes are 0.879 and 0.837, respectively. Such high F-1 values for the question and solution classes are highly desirable as the posts corresponding to these two classes contain the most important information in the thread—the problem being discussed and its solution. Identifying such posts with high accuracy is crucial for applications such as forum search, thread summarization, question-answer pair detection, and so on. We also note that the classifier performance for the *clarification* and the two feedback classes for the Ubuntu data set is much lower when compared with the other classes. This low performance can be attributed to the small number of posts belonging to these classes in the data set (29, 11, and 25 posts for clarification, negative, and positive feedback classes, respectively) and thus, the inability of the classifiers to generalize over this small amount of data. The low performance of repeat question and negative feedback classes in the NYC data set could also be attributed to the small number of posts belonging to these classes in the data set.

For further error analysis, we report the confusion matrices for the eight classes in Table 7. In the table, each class is represented by a number and the mapping is as follows: 1–Question, 2–Repeat Question, 3–Clarification, 4–Further Details, 5–Solution, 6–Negative Feedback, 7–Positive Feedback, and 8–Junk. By looking at the confusion matrix for the NYC data set, we note that a majority of posts belonging to the Junk category have been incorrectly classified as Solution posts (40 of 54). This indicates that even though these posts had characteristics like solution posts, they did not provide any useful information to the end user, and hence were marked as Junk by human evaluators. Likewise, in the Ubuntu data set, we note that almost 25% of posts belonging to the Junk category were marked as Solution posts. Another interesting observation to make is that for both the data sets, a large fraction of the posts belonging to the Clarification class were incorrectly labeled as solution class by the classifier. It could be attributed to high content similarity of solution and clarification posts with the original question post and thus, the error made by the classifier.

Given the small number of posts belonging to classes other than the Question and Solution classes, we also

[6]http://lemurproject.org

DOI: 10.1002/asi

TABLE 6. Classification results for each class for the Ubuntu and NYC data sets.

Class	Precision	F-1 measure	TP rate	FP rate	ROC area
Ubuntu data set					
Question	0.860	0.867	0.873	0.048	0.959
Repeat question	0.786	0.710	0.647	0.006	0.926
Clarification	0.286	0.186	0.138	0.020	0.930
Further details	0.632	0.643	0.655	0.044	0.909
Solution	0.754	0.810	0.874	0.183	0.912
Negative feedback	0.333	0.143	0.091	0.004	0.583
Positive feedback	0.400	0.356	0.320	0.024	0.910
Junk	0.622	0.583	0.549	0.036	0.901
Overall	0.705	0.712	0.730	0.094	0.917
NYC data set					
Question	0.899	0.879	0.859	0.017	0.978
Repeat question	0	0	0	0	0.402
Clarification	0.588	0.536	0.492	0.026	0.912
Further details	0.549	0.533	0.519	0.028	0.932
Solution	0.770	0.837	0.917	0.293	0.886
Negative feedback	0	0	0	0.001	0.753
Positive feedback	0.667	0.630	0.596	0.044	0.885
Junk	0.500	0.100	0.056	0.004	0.806
Overall	0.726	0.724	0.751	0.163	0.897

TABLE 7. Confusion matrix for classification task for both data sets.

Ubuntu data set								
↓ (True) Class → (Predicted)	1	2	3	4	5	6	7	8
1	117	0	0	3	9	0	2	3
2	0	11	1	0	5	0	0	0
3	1	2	4	0	21	0	0	1
4	6	0	0	36	3	1	4	5
5	8	1	9	5	181	0	3	0
6	1	0	0	3	4	1	0	2
7	2	0	0	4	4	1	8	6
8	1	0	0	6	13	0	3	28
NYC data set								
↓ (True) Class → (Predicted)	1	2	3	4	5	6	7	8
1	116	0	5	5	6	0	3	0
2	0	0	0	0	2	0	0	0
3	2	0	30	0	29	0	0	0
4	4	0	0	28	8	0	14	0
5	4	0	13	8	419	1	10	2
6	0	0	0	1	6	0	0	0
7	3	0	1	7	34	0	68	1
8	0	0	2	2	40	0	7	3

Note. Class name to class number mapping is as follows: 1–Question, 2–Repeat Question, 3–Clarification, 4–Further Details, 5–Solution, 6–Negative Feedback, 7–Positive Feedback, and 8–Junk.

TABLE 8. Classification results by considering only three classes—question, solution, and other.

Class	Precision	F-1 measure	TP rate	FP rate	ROC area
Ubuntu data set					
Question	0.876	0.859	0.843	0.041	0.961
Solution	0.783	0.722	0.670	0.103	0.868
Other	0.745	0.798	0.860	0.189	0.896
Overall	0.791	0.787	0.788	0.121	0.903
NYC data set					
Question	0.870	0.857	0.844	0.023	0.982
Solution	0.728	0.638	0.568	0.105	0.830
Other	0.775	0.829	0.891	0.276	0.884
Overall	0.774	0.770	0.777	0.181	0.881

experimented by combining all the classes other than the question and solution classes into one single class. Thus, the problem now reduces to a three-class problem with three classes: Question, Solution, and Other. Table 8 reports the classification results for this setting. We note that reducing the number of classes does not help in improving the performance. For question class, the F-1 values as achieved by the three class classifier are 0.859 and 0.857 for the Ubuntu and NYC data sets, respectively. For the original eight class classifier, the F-1 values for the question class are 0.856 and 0.879 for the Ubuntu and NYC data sets, respectively. Likewise, for the solution category, the values achieved by using only thee classes are lower than when achieved using all the eight classes.

Relative Importance of Different Features

In this subsection, we investigate the effect of different types of features for the post classification task. We perform the classification experiment using only one type of feature at a time. Table 9 reports the classification results for each feature type. We report precision, the F-1 measure and accuracy values. As before, a tenfold cross validation was performed and results reported are averaged over the ten folds. We note that for both the data sets, the highest individual performance is achieved by content-based features. We

DOI: 10.1002/asi

TABLE 9. Classification results for the Ubuntu data set obtained using content based, structural, user based, and sentiment features individually.

Class	Precision	F-1 measure	Accuracy
Ubuntu data set			
Content	0.474	0.521	59.36%
Structural	0.379	0.429	50.47%
User	0.392	0.471	59.35%
Sentiment	0.360	0.359	45.30%
All	0.705	0.712	72.69%
NYC data set			
Content	0.560	0.584	65.50%
Structural	0.417	0.469	57.92%
User	0.431	0.505	61.99%
Sentiment	0.470	0.518	60.86%
All	0.726	0.724	75.11%

expect the content of a post to be a very strong indicator of the nature of the post and hence, the high performance of content-based features that take into account the relationship of the post content with the remaining thread. We also note that although the performance of sentiment based features is worst (in terms of F-1) among the four feature types for the Ubuntu data set, they are the second best performing feature type for the NYC data set (in terms of F-1). This difference could be attributed to the different nature of the two forums. A majority of discussions in the Ubuntu data set are about technical problems faced by different Ubuntu users. Hence, the problem description and related discussions tend to be more *objective* in nature. Further, although we expect the content-based, user-based, and structural features to be helpful in identifying posts belonging to all the classes, sentiment features are expected to be most useful in identifying posts belonging to the two feedback classes. In addition, the small number of posts belonging to the two feedback classes (refer to Table 4) may also be responsible for the low performance of the sentiment-based features. On the other hand, a majority of threads in the NYC forums contain discussions about users' travel plans and trip and vacation experiences. Hence, the discussions tend to be more *subjective* and there are more *sentiment-based clues* for the classifier to learn. Thus, for NYC data set, we expect the sentiment features to be useful for classes other than the two feedback classes.

Next, we study the importance of each individual feature for the post classification task. We evaluate all the features individually by measuring the chi-squared statistic with respect to the class label and rank all the features by their chi-square values. Table 10 lists the top 10 features for the post classification task for the two data sets. We note that no sentiment based features appear in the list of top ten features for the Ubuntu data set whereas features belonging to all feature types are present in top ten features for NYC data set, in accord with the previous observations.

Retrieval experiments—does post class information help improve thread retrieval? In this section, we describe

TABLE 10. Top 10 features for the data sets ranked by chi-square values.

Ubuntu	NYC
InitPostSim	AbsPosition
AbsPosition	InitPostSim
ThreadSim	IsStarter
IsStarter	NormPosition
PostLengthUnique	QuestionMark
PostLengthUniqueStemmed	Thank
PostLength	UserAuthority
UserAuthority	UserPostCount
QuestionMark	TitleSim
TitleSim	PostLengthUnique

TABLE 11. Effect of combining various priors with the Inference Network based (INet) thread retrieval model for Ubuntu and NYC data sets.

Method	MRR	P@5	P@10	NDCG@10	MAP
Ubuntu data set					
INet	0.7300	0.4880	0.4560	0.6788	0.6003
INet + S	**0.8533**	**0.5840**[a]	0.4840	**0.7692**[a]	**0.7001**[a]
INet + P	0.7480	0.5200	0.4880	0.7076	0.6190
INet + N	0.7300	0.4880	0.4560	0.6790	0.6003
INet + S + P	0.8200	0.5680	**0.5080**	0.7612a	0.6909a
INet + S + N	**0.8533**	**0.5840**[a]	0.4840	**0.7692**[a]	**0.7001**[a]
INet + P + N	0.7480	0.5200	0.4880	0.7074	0.6180
INet + S + P + N	0.8200	0.5680	**0.5080**	0.7612a	0.6909a
NYC data set					
INet	0.7587	0.5520	0.5360	0.7065	0.6572
INet + S	0.7924	0.6000	0.5440	0.7348	0.7027a
INet + P	0.7847	0.6080	0.5520	**0.7385**	0.6743
INet + N	0.7587	0.5520	0.5400	0.7083	0.6575
INet + S + P	**0.8100**	**0.6080**	**0.5680**	0.7344	0.6767
INet + S + N	0.7924	0.6000	0.5400	0.7350	**0.7044**[a]
INet + P + N	0.7847	**0.6080**	0.5480	0.7342	0.6742
INet + S + P + N	**0.8100**	**0.6080**	**0.5680**	0.7348	0.6774

Note. Symbols S, P, and N represent Solution, Positive Feedback, and Negative Feedback priors, respectively. Statistically significant improvements over the baseline model using Wilcoxon's signed rank test with 95% confidence interval (p-value < 0.05) are denoted by [a]. Figures in bold denote the best performing settings.

results of incorporating post class label information in the baseline retrieval model as discussed in the Using Post Class Labels for Improving Thread Retrieval section. To obtain the post class labels, we used the classifier trained on the labeled data sets of 100 threads to classify posts in all the other remaining threads for the Ubuntu and NYC data sets, respectively. Table 11 reports the retrieval results for the two data sets using different retrieval models. INet in the table refers to the baseline retrieval model based on inference networks; S, P, and N denote the solution, positive feedback, and negative feedback priors, respectively. We experiment by adding each prior one at a time and then by adding different prior combinations. From the table, we note that for both the data sets, incorporating different priors results in increased retrieval performance for many settings when compared to the baseline model. We also observe that

DOI: 10.1002/asi

incorporating negative feedback prior did not change the performance much when compared to the baseline model. One reason for this behavior could be the low count of negative feedback posts. In the random sample of threads that were used for post classification, we note that the negative feedback posts account for less than 2% of all the posts for both the data sets (refer to Table 4). We expect the distribution to be similar in the whole corpus and, hence, the inability to improve retrieval performance. Addition of solution and positive feedback priors for both the data sets achieves improvements over the baseline methods across all the metrics. Further, for NYC data set, the combination of solution and positive feedback priors performs best in terms of MRR and precision at ranks 5 and 10. For further analysis, we perform a per-query comparison in terms of average precision value for each model for both the data sets. For

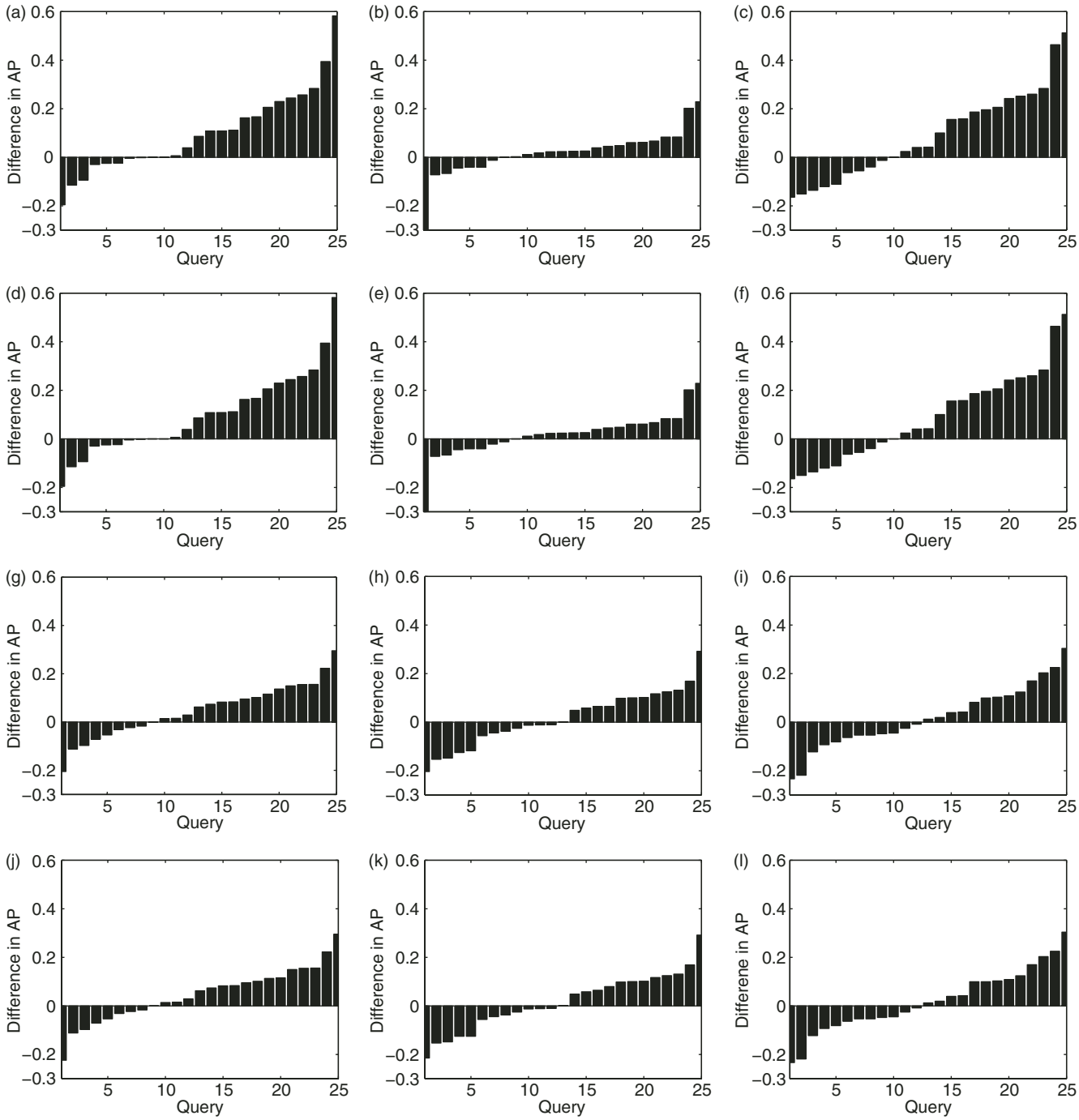

FIG. 2. Bar plots showing per query comparison between average precision values. Each bar represents the difference in average precision value achieved by the method using post class label information and the baseline method. S, P, and N correspond to solution, positive feedback, and negative feedback priors, respectively.

DOI: 10.1002/asi

each prior, we compute the difference in average precision achieved by the model utilizing that prior and the baseline model. Figure 2 summarizes this information and gives an idea of how many queries benefit from utilizing post class label information. Note that because of space constraints, we do not show the plots for the negative prior case for both data sets as the results for this model and the baseline model were almost similar for reasons discussed previously. From the figure, we note that by incorporating the additional prior information in the baseline retrieval model, on an average, the number of queries that gained in terms of average precision is more than the number for which average precision decreased. Further, for queries that gain in average precision, the magnitude of gain is more than the magnitude of loss for queries that do not gain in average precision. Thus, we see that on an average, models that incorporate post class information are able to achieve higher average precision values for a larger number of queries as compared to the baseline model that does not utilize post class label information.

Conclusions and Future Work

In this work, we investigated the problem of classifying individual user posts in an online discussion thread. For post classification, we experimented with a variety of features derived from the post's content, thread structure, user behavior, and sentiment analysis of the post's text. We experimented with two different data sets and achieved strong classification accuracy on both. We then incorporated the post class label information as prior probabilities in a state-of-the-art thread retrieval model and found that post classification can help improve thread retrieval performance. Our future work will focus on utilizing post class label information for thread summarization.

References

Aikawa, N., Sakai, T., & Yamana, H. (2011). Community QA question classification: Is the asker looking for subjective answers or not? IPSJ Online Transactions, 4, 160–168.

Bhatia, S., & Mitra, P. (2010). Adopting inference networks for online thread retrieval. In Proceedings of the 24th AAAI Conference on Artificial Intelligence (pp. 1300–1305). Atlanta, GA: AAAI.

Bhatia, S., Biyani, P., & Mitra, P. (2012). Classifying user messages for managing web forum data. In Fifteenth International Workshop on the Web and Databases, WebDB (pp. 13–18). Scottsdale, AZ: ACM.

Bian, J., Liu, Y., Agichtein, E., & Zha, H. (2008). Finding the right facts in the crowd: Factoid question answering over social media. New York, NY: ACM.

Biyani, P., Bhatia, S., Caragea, C., & Mitra, P. (2012a). Thread specific features are helpful for identifying subjectivity orientation of online forum threads. In COLING (pp. 295–310). Mumbai, India: The COLING 2012 Organizing Committee.

Biyani, P., Caragea, C., Singh, A., & Mitra, P. (2012b). I want what I need! Analyzing subjectivity of online forum threads. In Proceedings of the 21st ACM International Conference on Information and Knowledge Management (pp. 2495–2498). New York, NY: ACM.

Biyani, P., Caragea, C., & Mitra, P. (2013a). Predicting subjectivity orientation of online forum threads. In Proceedings of the 14th International Conference on Intelligent Text Processing and Computational Linguistics (pp. 109–120). Berlin, Heidelberg: Springer-Verlag.

Biyani, P., Caragea, C., Mitra, P., Chong Z., Yen, J., Greer, G.E., & Portier K. (2013b). Co-training over domain-independent and domain-dependent features for sentiment analysis of an online cancer support community. In 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM) (pp. 413–417).

Biyani, P., Bhatia, S., Caragea, C., & Mitra, P. (2014). Using non-lexical features for identifying factual and opinionative threads in online forums. Knowledge-Based Systems, 69, 170–178.

Cohen, W.W., Carvalho, V.R., & Mitchell, T.M. (2004). Learning to classify email into "speech acts." In D. Lin & D. Wu (Eds.), Proceedings of EMNLP 2004 (pp. 309–316). Barcelona, Spain: Association for Computational Linguistics.

Cong, G., Wang, L., Lin, C.-Y., Song, Y.-I., & Sun, Y. (2008). Finding question-answer pairs from online forums. In Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '08) (pp. 467–474). New York, NY: ACM.

Ding, S., Cong, G., Lin, C.-Y., & Zhu, X. (2008). Using conditional random fields to extract contexts and answers of questions from online forums. In Proceedings of ACL08—HLT 2008 (pp. 710–718). Columbus, OH: Association for Computational Linguistics.

Duan, H., & Zhai, C. (2011). Exploiting thread structures to improve smoothing of language models for forum post retrieval. In Proceedings of the 33rd European Conference on Advances in Information Retrieval (pp. 350–361). Berlin, Heidelberg, Germany: Springer-Verlag.

Elsas, J.L., & Carbonell, J.G. (2009). It pays to be picky: An evaluation of thread retrieval in online forums. In Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 714–715). New York, NY: ACM.

Feng, D., Shaw, E., Kim, J., & Hovy, E. (2006). Learning to detect conversation focus of threaded discussions. In Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics, HLT-NAACL '06 (pp. 208–215). Stroudsburg, PA: Association for Computational Linguistics.

Fleiss, J.L. (1971). Measuring nominal scale agreement among many raters. Psychological Bulletin, 76(5), 378–382.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I.H. (2009). The WEKA data mining software: An update. SIGKDD Exploration Newsletter, 11(1), 10–18.

Hong, L., & Davison, B.D. (2009). A classification-based approach to question answering in discussion boards. In Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 171–178). New York, NY: ACM.

Huang, J., Zhou, M., & Yang, D. (2007). Extracting chatbot knowledge from online discussion forums. In Proceedings of the 20th International Joint Conference on Artifical Intelligence (pp. 423–428). San Francisco, CA: Morgan Kaufmann Publishers Inc.

Jeong, M., Lin, C.-Y., & Lee, G.G. (2009). Semi-supervised speech act recognition in emails and forums. In Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (pp. 1250–1259). Singapore: Association for Computational Linguistics.

Joty, S., Carenini, G., & Lin, C.-Y. (2011). Unsupervised modeling of dialog acts in asynchronous conversations. In Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence—Volume Three (pp. 1807–1813). AAAI Press.

Kim, J., Chern, G., Feng, D., Shaw, E., & Hovy, E. (2006). Mining and assessing discussions on the web through speech act analysis. In Proceedings of the ISWC'06 Workshop on Web Content Mining with Human Language Technologies.

Kim, S.N., Wang, L., & Baldwin, T. (2010). Tagging and linking web forum posts. In Proceedings of the 14th Conference on Computational Natural Language Learning, CoNLL '10 (pp. 192–202). Stroudsburg, PA: Association for Computational Linguistics.

Lampert, A., Dale, R., & Paris, C. (2008). The nature of requests and commitments in email messages. Informal workshop paper.

Li, B., Liu, Y., & Agichtein, E. (2008). Cocqa: Co-training over questions and answers with an application to predicting question subjectivity orientation. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP '08 (pp. 937–946). Stroudsburg, PA: Association for Computational Linguistics.

Lin, C., Yang, J.-M., Cai, R., Wang, X.-J., & Wang, W. (2009). Simultaneously modeling semantics and structure of threaded discussions: A sparse coding approach and its applications. In SIGIR '09: Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 131–138). New York, NY: ACM.

Murray, G., Renals, S., Carletta, J., & Moore, J.D. (2006). Incorporating speaker and discourse features into speech summarization. In Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics (pp. 367–374). Stroudsburg, PA: Association for Computational Linguistics.

Ponte, J.M., & Croft, W.B. (1998). A language modeling approach to information retrieval. In Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. New York, NY: ACM.

Porter, M.F. (1980). An algorithm for suffix stripping. Program, 14(3), 130–137.

Seo, J., Croft, W.B., & Smith, D.A. (2009). Online community search using thread structure. In Proceedings of the 18th ACM Conference on Information and Knowledge Management. New York, NY: ACM.

Stoicke, A., Ries, K., Coccaro, N., Shriberg, E., Bates, R., Jurafsky, D., . . . Meteer, M. (2000). Dialogue act modeling for automatic tagging and recognition of conversational speech. Computational Linguistics, 26(3), 339–373.

Thelwall, M., Buckley, K., Paltoglou, G., Cai, D., & Kappas, A. (2010). Sentiment in short strength detection informal text. Journal of the American Society for Information Science and Technology, 61(12), 2544–2558.

Wang, X.-J., Tu, X., Feng, D., & Zhang, L. (2009). Ranking community answers by modeling question-answer relationships via analogical reasoning. In Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval. New York, NY: ACM.

Wang, H., Wang, C., Zhai, C., & Han, J. (2011). Learning online discussion structures by conditional random fields. In Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 435–444). New York, NY: ACM.

Xu, G., & Ma, W.-Y. (2006). Building implicit links from content for forum search. In Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. New York, NY: ACM.

Xue, X., Jeon, J., & Croft, W.B. (2008). Retrieval models for question and answer archives. In SIGIR '08: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 475–482). New York, NY: ACM.

Yang, J.-M., Cai, R., Wang, Y., Zhu, J., Zhang, L., & Ma, W.-Y. (2009). Incorporating site-level knowledge to extract structured data from web forums. In Proceedings of the 18th International Conference on World Wide Web. New York, NY: ACM.

Yang, W.-Y., Cao, Y., & Lin, C.-Y. (2011). A structural support vector method for extracting contexts and answers of questions from online forums. Information Processing & Management, 47, 886–898.

Yu, H., & Hatzivassiloglou, V. (2003). Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing, EMNLP '03 (pp. 129–136). Stroudsburg, PA: Association for Computational Linguistics.

Zhai, C., & Lafferty, J. (2001). A study of smoothing methods for language models applied to ad hoc information retrieval. In Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 334–342). New York, NY: ACM.

Zhao, J., Bu, J., Chen, C., Guan, Z., Wang, C., & Zhang, C. (2010). Learning a user-thread alignment manifold for thread recommendation in online forum. In Proceedings of the 19th ACM International Conference on Information and Knowledge Management, CIKM '10 (pp. 559–568). New York, NY: ACM.

Zhou, L., & Hovy, E.H. (2006). On the summarization of dynamically introduced information: Online discussions and blogs. In AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs (p. 237). AAAI.

DOI: 10.1002/asi