PriSM: Discovering and Prioritizing Severe Technical Issues from Product Discussion Forums

Rashmi Gangadharaiah
IBM Research, India
rashgang@in.ibm.com

Rose Catherine
IBM Research, India
rosecatherinek@in.ibm.com

ABSTRACT

Online forums provide a channel for users to report and discuss problems related to products and troubleshooting, for faster resolution. These could garner negative publicity if left unattended by the companies. Manually monitoring these massive amounts of discussions is laborious. This paper makes the first attempt at collecting issues that require immediate action by the product supplier by analyzing the immense information on forums. Features that are specific to forum discussions, in conjunction with linguistic cues help in capturing and better prioritizing issues. Any attempt to collect training data for learning a classifier for this task will require enormous labeling effort. Hence, this paper adopts a co-training approach, which uses minimal manual labeling, coupled with linguistic features extracted using a set-expansion algorithm to discover severe problems. Further, most distinct and recent issues are obtained by incorporating a measure of 'centrality', 'diversity' and temporal aspect of the forum threads. We show that this helps in better prioritizing longstanding issues and identify issues that need to be addressed immediately.

Categories and Subject Descriptors

H.3.5 [**On-line Information Services**]: Web-based services

Keywords

Discovering and Prioritizing Severe Technical Issues

1. INTRODUCTION

Online social media sites and discussion forums have rapidly become the primary channel for users to share views on issues ranging from troubleshooting hardware to deciding on what car to buy. Forums encourage user participation and simultaneously reduce demand on the companies' service centers. Users today prefer to post product-related issues directly on forums as it often facilitates quicker responses.

CIKM'12, October 29–November 2, 2012, Maui, HI, USA.

In addition to providing benefits to users, these sites have become an excellent source of information that can be mined to estimate consumer sentiment. However, as a result of the rapid growth in the popularity and access to these social media sites, they are often unstructured and difficult to process automatically, requiring effective use of both natural language processing and data mining techniques.

One alternative to online forums are, product manuals which typically contain frequently asked questions (FAQs). Unfortunately, these often do not contain solutions to all problems that may arise. An example of this is the reception problems that were faced by consumers on the release of Apple's® iPhone4[1]. A workaround for this was posted on forums much before official FAQs or solutions were released.

Discussion threads on various issues related to a product grow rapidly when a product is launched or an update is released (Apple® discussions has ≈300k threads on iPhone related issues[2]), making it a formidable task to monitor them manually. Hence, a technique to automatically obtain some kind of "priority" assignment is required to help developers prioritize these fixes. To the best of the authors' knowledge, our work is the first attempt at both discovering and Prioritizing Severe issues by Mining (PriSM) discussion forums in order to extract actionable business intelligence.

Specifically, the contributions of this paper are as follows:

- This paper makes the first attempt at discovering and prioritizing issues that require immediate action by product suppliers, from the large amounts of semi-structured information available in discussion forums.
- We develop forum-specific features and linguistic cues (Section 3.1) to prioritize issues, using minimal amounts of labeled data using co-training [1] (Section 3.3).
- We show how to incorporate a measure of 'diversity' into the issues we identify and show that this increases the usefulness of the result (Table 3 vs. Table 5).
- Finally, we model the temporal aspect of forum threads and their issues by showing that appropriate discounting of older threads helps both prioritizing longstanding issues as well as identifying issues that need to be addressed immediately (Table 5 vs. Table 6).
- We demonstrate that we are able to identify important problems, often before they are officially recognized by showing that the top issues that we find, match issues posted by product researchers (Section 4.3).

9[1]http://www.pcworld.com/article/199807/iphone_4_antenna_problems_mar_apples_big_day.html

9[2]https://discussions.apple.com/community/iphone

Cluster	Sample Thread Titles
Apps	**-How to update apps ?** **-Is there a translation app for texting ?**
email	**-MMS through email** -image attachments to emails **-Roadrunner email question**
photos	-How to mail more than one photo ? **-Want to Batch-delete photos?** **-Delete Photo Library**

Table 1: Apple® Discussions: Threads clustered and ranked based on density of the clusters.

2. LIMITATIONS OF PRIOR WORK AND OUR METHODOLOGY

Typically companies manually assess the severity of an issue from bug reports. This can be tedious, expensive and time-consuming. A few attempts have been made in the past towards mining and prioritizing bug fixes from bug reports [5]. However, no attempts have been made at extracting and prioritizing problems faced by users on forums.

An obvious approach to prioritizing issues is to use a measure of 'frequency' [3]. We argue that frequency alone is not a good indicator for this task; in fact, many frequently asked problems have solutions in product manuals or in the official FAQ lists and these problems are clearly not a high priority. To demonstrate this, we clustered the threads in our crawl of iPhone forum data to group similar threads together. Sample threads from the biggest clusters, indicating highest frequency, are shown in Table 1. Clearly, many of these problems (shown in bold) are already in the FAQ list provided by the iPhone manufacturer.

[5] used just lexical features to train supervised classifiers. We show that the performance of classifiers trained on lexical features can be boosted by incorporating forum-specific features (Section 4.1) (which are not present in bug reporting tools). These simple experiments highlight the fact that, in order to extract severe issues that need the manufacturer's attention, other cues from the discussion threads are crucial.

We propose using forum-specific features and other linguistic cues to automatically classify issues from forum discussions. However, the amount of labeled data available to learn these classifiers was small. We use Co-training [1], which is a learning method that has been shown to work well when the amount of labeled data is insufficient.

We then rank these severe threads to find distinct issues, using a measure of 'centrality' and 'diversity' to prevent redundant (or very similar) issues from receiving a high rank. At the same time, the issues that are ranked the highest are the ones that form a good representation of all related issues. We use GRASSHOPPER (Graph Random-walk with Absorbing StateS that HOPs among PEaks for Ranking) [8] that ranked items in a way that highly ranked items formed good representatives of their local groups (items similar to the highly ranked item) and the top ranked items were diverse items. In this paper, we extend this approach by incorporating the 'temporal aspect' of the issues by discounting older threads to better prioritize longstanding issues as well as identify issues that need immediate attention by appropriately modifying the GRASSHOPPER algorithm.

3. PROPOSED APPROACH

This paper proposes a bottom-up approach to discover problems that require immediate attention, from social media discussions. The first step is to decide at a discussion (thread) level, if that thread is discussing an issue that could be severe. Once such potentially severe issues have been identified, they are aggregated, to discover and rank issues globally, according to their severity. In the rest of the paper, 'severity of an issue' refers to the degree of impact the issue has on the usability of a product. For e.g., a bug that crashes an iPhone frequently can be said to have high severity. Severity level of different issues vary with product, functionality and the domain.

Several cues to determine the severity of an issue exist in the posts. To decide the severity at a thread level, the key is to identify these 'severity indicators'. The indicators are grouped into two distinct groups: Lexical (LEX), which takes cues from words or phrases used in the text of the discussion, and Structural (STRUCT), which uses the structure and forum-specific features of the discussion.

3.1 Features Used

Consider the thread (predicted as severe) in Figure 1:
LEX: The following features are obtained using cues from the phrases mentioned in the discussions:

- **Usability (L_1):** if the discussion thread includes terms indicating that the product is no longer usable or has limited functionality, then this indicates that the problem could be severe. Eg. "freeze", "broken", etc.
- **Negative Emotion (L_2):** terms indicating outbursts of anger, such as "annoy", "frustrating", etc. show that the user is not satisfied with the product.
- **Frequency of occurrence (L_3):** problems that appear frequently can lead to frustration and need immediate attention. Indicators for this feature include terms such as, "frequently" and "always".
- **Urgency (L_4):** these are terms indicating urgency from the user. Eg. "urgently", "immediately", etc.
- **Number of people affected (L_5):** When a problem is severe, many users reply indicating that they too faced a similar problem. Eg. "same issue", "same problem", in subsequent replies posted in the thread.

STRUCT: The following features are obtained using structural and forum-specific features of the discussion thread:

- **Rating of Problem Author (S_1):** Novice users post queries which are usually mentioned in FAQs or product manuals, while expert users tend to post real or difficult problems that require more attention. Ratings of posts' authors were already available in the discussion site that was crawled to perform the experiments.
- **Length of thread (S_2):** When an issue is severe, discussions tend to grow longer. Severe issues involve exploring different suggestions until the issue is resolved or the thread initiator gives up.
- **Rating of Post Authors (S_3):** When more authors of high rating are involved in the discussion, it suggests that the problem being discussed is highly severe. The counts are grouped by the ratings.
- **Time Duration (S_4):** Time elapsed (in minutes) starting from when the question was posted to the time when the last post was entered. The longer the discussion lasts, more severe is the problem.

The fact that severity indicators can be grouped into independent sets of features, paves way for using the co-training methodology [1] with one classifier C_{lex} trained on LEX fea-

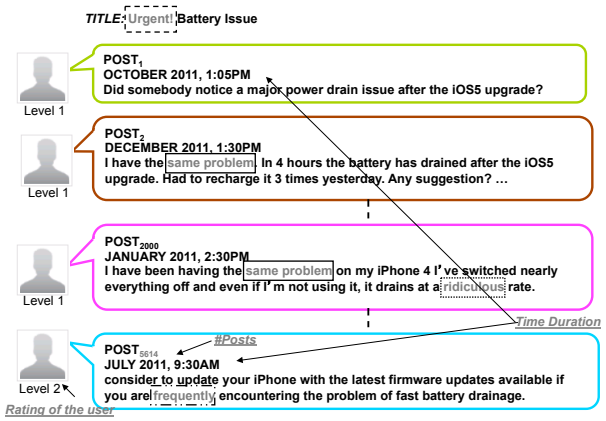

Figure 1: LEX and STRUCT cues in a sample Discussion Thread indicating a high severity level.

L_i	Seed [size]	Expansion [size]
L_1	crash, freeze [7]	useless, repair, mess, break [770]
L_2	annoy, frustrating [6]	ridiculous, horrible [87]
L_3	periodically, randomly [5]	frequently, daily [229]
L_4	urgently, immediately [5]	asap, deperately, sooner [31]

Table 2: Sample seeds and expanded sets

tures and another C_{struct}, trained on STRUCT features. Each of L_1 thru L_4 of LEX are employed as dictionaries, the building of which is discussed in Section 3.2. The features to C_{lex} are the frequency counts for terms/phrases that matched in these dictionaries. Implementation of C_{lex} and C_{struct} classifiers used LIBSVM[3].

3.2 Seed Set Expansion for Dictionaries

Given a small set of seed entities, set expansion discovers other entities that may belong to the same concept set. The dynamic thresholding algorithm, SEISA [4], was implemented for this. SEISA has been shown to outperform Random walk-based strategies [4] such as, SEAL, especially, when used on inherently noisy general-purpose web data and when the size of the seed set is small.

Terms from discussion threads obtained from a crawl of Apple's® iPhone forums (more details in Section 4) were used for this purpose. For example, with a seed set containing, "crashing" and "freezing", the algorithm retrieves other entities such as, "useless", "mess", etc. Terms supporting features, L_1, L_2, L_3 and L_4, were obtained with small seed sets. The candidate terms for L_1 and L_2 were restricted to verbs, adjectives and adverbs[4]. Candidate terms for L_3 and L_4 were limited to adverbs and adjectives. Additionally, whenever a candidate term was preceded by the adverb, "not" (as in the phrase: "*not* useful'), the "not" was also considered as part of the candidate term. Each stemmed term was modeled as a graph node on the left and the context (prefix and suffix of the term, up to 3 tokens) on the right. Sample output of set expansion for LEX is in Table 2, along with the sizes of the seed and the expanded sets.

[3] www.csie.ntu.edu.tw/∼cjlin/libsvm/

[4] http://nlp.stanford.edu/software/tagger.shtml

3.3 Co-training for Classification

The co-training methodology of [1] allows using inexpensive unlabeled data to augment a much smaller labeled set for achieving better accuracies. Here, each example has two distinct views and the assumption is that either of the two views is sufficient for learning if enough data were available. Both these views are used together for achieving better performance when a much smaller labeled set is available. Two learning algorithms are trained separately on the two different views of the examples, and each algorithm predicts possible labels for the examples in the unlabeled set. The prediction of one algorithm is used to augment the training data of the other learning algorithm and vice versa. A PAC-style analysis of this setting is given in [1] and has been shown to provide significant improvements by adopting this approach on the unlabeled examples.

In this paper, C_{lex} and C_{struct} are the conditionally independent views of the data. Classifiers trained in the final iteration are used to retrieve the final list of 'High' severity threads. To obtain a global ranking for issues from these threads, all threads on which the classifiers disagreed on the label are filtered out. Then the confidence of prediction on the rest of the threads is calculated as $P(label|C_{lex}) \times P(label|C_{struct})$ using the assumption of conditional independence, and those with confidence lower than the threshold, t_{conf}, are filtered out. The remaining threads are then clustered on their title and ranked based on the cluster sizes.

3.4 Diversity Ranking for 'Distinct' Issues

The previous step classified examples as having "High" or "Low" severity. To find the most severe issues related to the product, one could sort the issues based on the classifier's score ($P(label|C_{lex}) \times P(label|C_{struct})$). Table 3 shows the top ranked issues sorted based on the scores obtained from the classifier. The first column indicates the entity to which the issue is related to, the second column corresponds to the Topic (mentioned in the forums) and the third column summarizes the issue posted by the thread initiator. We see that many of the issues are related to each other. For example, the first and the second issue are both related to the 'battery draining' issue. Since it is uninformative to view such similar items, there is a need to rerank these issues such that the list only contains 'diverse' issues covering many distinct problems. At the same time, a highly ranked issue should be a representative of a group of related issues ('centrality').

For this, an adjacency matrix A of size $n \times n$ (where n is the number of issues obtained from the previous section) is built with each entry $A_{i,j}$ representing the cosine similarity between, $issue_i$ and $issue_j$. W is the score obtained from the classifiers from the previous section ($P(label|C_{lex}) \times P(label|C_{struct})$). A and W are discounted by λ_2^Y, where, y_i represents the difference in the year in which the thread i was posted and the present year. For example, if the present year is 2012 and the thread was posted in 2007, the difference $y_i = 5$. When calculating the similarity between node i and node j, each of their time differences, i.e., y_i and y_j are added to obtain y. This gives a higher score to issues posted more recently, as issues posted much earlier may already have been resolved by the producers of the product.

One could use only the most recent issues to perform diversity-based ranking under the assumption that issues posted recently are more severe than older issues as they may have already been solved. However, this fails to iden-

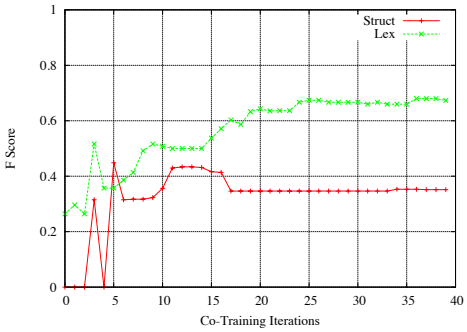

Figure 2: F-scores obtained with the two classifiers after each iteration of co-training.

tify longstanding issues. Considering old as well as new issues boosts the scores of recurring issues due to their large number of occurrences. Ideally, we would want to have a balance between the two. The parameter, λ_2 can be set by the product producers. In this paper, λ_2 was set to 0.4.

$$\tilde{P} = [\lambda_1 A + (1 - \lambda_1)1W^{\intercal}]. * \lambda_2^Y \quad (1)$$

$\tilde{P}$ is normalized to obtain the probability transition matrix P. '.*' in Equation 1 represents element-wise multiplication. P can also be viewed as a graph with n nodes and each entry in P, $P_{i,j}$ corresponds to the edge between $node_i$ and $node_j$. We perform a random walk on this graph. Ranking issues just based on the stationary distribution obtained at the end of the walk, will have the same problem of lack of diversity. We use the algorithm proposed in [8] to obtain distinct issues. The topmost node (g_1) obtained from the stationary distribution is converted into an absorbing state.[2] suggests a way to calculate the number of visits to each node before absorption. The node with the highest expected number of visits is ranked as the next prominent issue g_2. The process is repeated until the top K nodes are selected.

4. RESULTS AND DISCUSSION

Since there were no publicly available data sets with discussions, the dataset was crawled from Apple® Discussions with threads created during the period, 2007-2011. Out of about 147,000 discussion threads crawled, 270 threads were randomly chosen and manually labeled as 'High' or 'Low' according to their severity level by annotators who were iPhone users. Of these, 20 were used as the training set (initial labeled data for co-training) and the remaining 250 were used as the test set. The remaining threads were used as the unlabeled data for our co-training experiments.

The threads were processed to remove common stopwords and stemmed using Porter Stemmer [6]. In each iteration of Co-training, 5 most confident predictions from each of C_{lex} and C_{struct} were added to the set of training examples. This process was repeated for 40 iterations. t_{conf} was set to 0.7 which resulted in 512 threads which were then ranked based on centrality and diversity (as explained in Section 3.4).

4.1 Co-training for Severity Classification

To demonstrate the benefits of co-training, we show the accuracy of each of the classifiers for a few iterations on the 250 threads test set in Figure 2. We use the F-Score [7] as our accuracy measure. The results clearly show that each of the classifiers learn to improve their performance and correct their mistakes using the predictions of the other classifier. For example, the STRUCT based classifier initially had an F-Score of 0, but using the additional (originally unlabeled) data labeled by the LEX classifier, it's accuracy improves to 43.3. Similarly, the LEX classifier benefits substantially from the STRUCT classifier's predictions.

	STRUCT	LEX	Combined
Supervised	0	26.4	49.3
Co-Training	43.3	53.7	70.1

Table 4: First row: F-score obtained on the LEX, STRUCT and Combined classifiers with supervised learning, using only the labeled examples. Second row: F-score obtained with these classifiers with the co-training approach.

We also analyzed the improvements that could be obtained over standard separate training of supervised classifiers. The labeled set of 20 examples was used to train two classifiers- one trained only on LEX features, and the other trained on the STRUCT features. A combined classifier was also used by multiplying the probabilities of the predictions of the two classifiers ($P(label|C_{combined}) = P(label|C_{lex}) * P(label|C_{struct})$) and the class ('High' or 'Low') with the highest score was used. The resulting accuracies are given in the first row of Table 4. Accuracies of the individual classifiers obtained in the final iteration of co-training *with the same initial labeled set* are given in the second row of Table 4. As seen, a substantial gain in performance is observed with the use of the unlabeled data highlighting the effectiveness of co-training using our features.

4.2 Diversity Ranking and Time Discounting

Next, we study the value of time discounting and diversity ranking. Table 5 shows the top ranked results without the time discounting in Equation 1, and Table 6 shows the results with time discounting. If very recent discussions need to be given higher importance, then λ_2 needs to be smaller. λ_2 in Equation 1 was set to 0.4. λ_1 was set to 0.6 to give a higher weight to the classifiers' predictions. Variation of λ_1 (> 0.5) only resulted in variations of the ranks of the top few severe issues. However, the top issues shown in Table 6 remained within the Top 15 ranks. In general, this parameter may need to be controlled by the product producers.

To understand the usefulness of time discounting, we further analyzed the results in Table 5 and Table 6. A few of the issues in Table 5 correspond to issues with older versions of updates, also implied by the year in which the threads were created. Issues ranked high in Table 6 correspond to more recent issues. Considering other examples- the first version of iPhone did not support AVRCP[5], a feature that allows remote control of playback functions on the iOS device from Bluetooth devices. However, since the newer versions supported this capability, issues regarding the AVRCP disappeared in Table 6 where time discounting is involved, giving higher preference to recently posted issues. Other issues, such as "battery draining", "iPhone freezing" have been recurring issues, and they remain even with time discounting.

4.3 Comparing with Commonly Noted Issues

After the release of iPhone and subsequent long discussions on different social media about the issues that surfaced,

9[5]http://support.apple.com/kb/HT3647

Issue related to	Topic	First Post
battery draining	Battery drain since upgrading to 2.2	Last night I upgraded to 2.2. Today after being off the charger for 4 hours it is down to 20%...
battery draining	Battery Drain!	It is fully charged in morning, by lunch its down to 70%
Wifi	no WiFi signal on iPhone V2	Updated to V2, now my Wifi reception is poor ...
Wifi	iPhone wifi not working	my iphone has had major difficulties finding wifi networks..
Data outage	Widespread AT and T data outage	There are multiple reports of a widespread AT and T data outage

Table 3: Top 5 ranked issues (based on the classifiers' scores)

Issue related to	Year	Topic	First Post
battery draining	2009	iPhone 3GS with fw 3.1, battery life gets even worse	I've made the so long attended upgrade to iPhone OS 3.1 and strangely battery life has got even worse...
GPS	2009	GPS	Anyone having GPS problems with 3.0
freezing	2011	iOS 4.3 update to iPhone 3Gs freezes phone	I did the update to the phone and I have not been able to boot it up since then ....
AVRCP	2009	No AVRCP?	Why no AVRCP? Arrrggggg ...
Data outage	2009	Widespread AT and T data outage	There are multiple reports of a widespread AT and T data outage affecting both 3G and EDGE networks.

Table 5: Top ranked issues without time discounting

Issue related to	Year	Topic	First Post
battery draining	2011	battery drain	iOS 4.3 is causing poor battery performance.
freezing	2011	iOS 4.3 update freezes phone	I did the update and I have not been able to boot ...
slowness	2011	4.2.1's lag issue on iPhone 4	I just updated to 4.2.1 few days ago ...
Wifi	2011	2.2.1 Killed my wifi	did the 2.2.1 update, but i'm unable to connect.
heat issue	2011	iPhone 4 heat issue	My iPhone 4 has started to get very hot ....

Table 6: Top ranked issues with time discounting

several bloggers and market researchers have attempted to collate the most-talked about issues; for example, these webpages[6] [7] lists the most annoying problems and it can be noted that, Table 6 closely matches many of them.

5. CONCLUSION AND FUTURE WORK

This paper proposed an approach to automatically discover severe issues from discussions with very minimal supervision from large amounts of semi-structured information available in forums. Linguistic cues and other forum specific features with minimal amounts of training data were used to train severity predictors. To prevent similar issues from all obtaining a high rank, a measure of 'diversity' and 'centrality' was adopted. Temporal aspect of forum threads was used to further give higher preference to recently created threads which helped in prioritizing longstanding issues as well as identifying issues that need immediate attention.

6. REFERENCES

[1] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *COLT 1998*, pages 92–100.

[2] P. G. Doyle and J. L. Snell. Random walks and electric networks. In *Mathematical Association of America*, 1984.

[3] A. Fourney, R. Mann, and M. Terry. Characterizing the usability of interactive applications through query log analysis. In *CHI 2011*, pages 1817–1826.

[4] Y. He and D. Xin. Seisa: set expansion by iterative similarity aggregation. In *WWW 2011*, pages 427–436.

[5] A. Lamkanfi, S. Demeyer, Q. D. Soetens, and T. Verdonck. Comparing mining algorithms for predicting the severity of a reported bug. In *CSMR 2011*, pages 249–258. IEEE Computer Society.

[6] M. F. Porter. Readings in information retrieval. chapter An algorithm for suffix stripping, pages 313–316. Morgan Kaufmann Publishers Inc., 1997.

[7] C. J. van Rijsbergen. *Information Retrieval (2nd ed.)*. Butterworth, 1979.

[8] X. Zhu, A. B. Goldberg, J. Van, and G. D. Andrzejewski. Improving diversity in ranking using absorbing random walks. In *Physics Laboratory University of Washington*, pages 97–104, 2007.

[6] http://iphoneappcafe.com/10-most-annoying-iphone-problems-and-how-to-solve-them/ (accessed Feb 3, 2012)

[7] http://www.macnn.com/articles/10/07/28/ios.said.to.be.unusable.on.the.iphone.3g/