# Thread-level Analysis over Technical User Forum Data

**Li Wang, Su Nam Kim and Timothy Baldwin**

NICTA VRL
Department of Computer Science and Software Engineering
University of Melbourne
VIC 3010 Australia

li.wang.d@gmail.com, sunamkim@gmail.com, tb@ldwin.net

## Abstract

This research focuses on improving information access over troubleshooting-oriented technical user forums via thread-level analysis. We describe a modular task formulation and novel dataset, and go on to describe a series of preliminary classification experiments over the data. We find that a class composition strategy achieves the best results, surpassing multiclass classification approaches.

## 1 Introduction

Online forums and discussion boards are online platforms for people to hold discussions in particular domains. They are widely used in various areas such as customer support, community development, interactive reporting and online education. Forums provide one of the primary avenues for online users to share information on the Internet. Users post their questions or problems onto online forums and get possible solutions from other users. Through this simple mechanism, great volumes of data with customised answers to highly specialised domain-specific questions are created on a daily basis. However, it is not an easy job to extract the information latent in the threads.

The aim of our research is to help users to more easily access existing information in forums which relate to their questions, by text mining troubleshooting-oriented, computer-related technical user forum data (Baldwin et al., 2010). An example thread from a real-world forum is shown in Figure 1, which is made up of 4 posts with 3 distinct participants.

Our proposed strategy is to model the "content structure" of forum threads by analysing requests for information and provision of solutions in the thread data. We devise an ontology of *problem sources* and *solution types* with which to analyse individual threads, paving the way for users to spell out the general nature of their support need in their queries. The main contributions of this paper are: (1) designing a modular thread-level class set; (2) constructing and publishing an annotated dataset; and (3) performing preliminary thread-level experiments over the dataset.

**HTML Input Code - CNET Coding & scripting Forums**

| | |
|---|---|
| User A<br>Post 1 | **HTML Input Code**<br>...Please can someone tell me how to create an input box that asks the user to enter their ID, and then allows them to press go. ... |
| User B<br>Post 2 | **Re: html input code**<br>Part 1: create a form with a text field. ... |
| User C<br>Post 3 | **asp.net c# video**<br>I've prepared for you video.link click ... |
| User A<br>Post 4 | **Thank You!**<br>I have Microsoft Visual Studio 6, what program should I do this in? ... |
| User D<br>Post 5 | **A little more help**<br>...You would simply do it this way: ... |

Figure 1: An extract from a real-world thread

## 2 Related Work

There is very little work that is specifically targeted at the thread-level analysis of web user forum data. The most closely-related work is that performed by Baldwin et al. (2007), and our thread class set was created based on this original work.

Another research line that relates to the thread classification is discussion summarisation. For example, technical online IRC (Internet Relay Chat) discussions are summarised and segmented in Zhou and Hovy (2005)'s research. The message segments are then clustered to find the most relevant information to users using machine learning models. There has also been work on email summarisation, concentrating primarily on summarising and organising email archives by extracting overview sentences to help users find the most useful email threads (Nenkova and Bagga, 2004;

Li Wang, Su Nam Kim and Timothy Baldwin. 2010. Thread-level Analysis over Technical User Forum Data. In *Proceedings of Australasian Language Technology Association Workshop*, pages 27−31

Rambow et al., 2004; Wan and McKeown, 2004).

## 3 Class Definition

The proposed thread class set is made up of two orthogonal Basic Class sets (BASIC), and a Miscellaneous Class set (MISC). The first BASIC class set is Problem Source (PROBLEM), which concerns the targets and sources of the problem described in threads. It contains 6 basic classes: *Operating System* (*OS*), *Hardware*, *Software*, *Media*, *Network* and *Programming*. The second BASIC class set is Solution Type (SOLUTION), which describes the types of the solution presented in threads in the form of 4 classes: *Documentation*, *Install*, *Search* and *Support*. The MISC class set includes two classes: *Other* and *Spam*. A detailed description of each class in the thread class set is presented in Table 1.

A given thread is labelled either with one class label from each of the two BASIC class sets (i.e. two class labels in total), or alternatively one class label from the MISC class set. For example, the thread from Figure 1 would be labelled as *Programming*/*Documentation*. Therefore, when doing the actual annotation, we used the ALLCLASS class set containing 26 classes in total, i.e. the cross product of the two BASIC class sets plus *Other* and *Spam*.

It should be noted that while the design of our class set is specific to computer-related technical user forum threads, the idea of the two orthogonal BASIC class sets, namely PROBLEM and SOLUTION, can be applied to troubleshooting-oriented forum threads from other domains. This is because most troubleshooting-oriented forum threads present one or more problems (i.e. PROBLEM), and imply possible solution types (i.e. SOLUTION), even if the thread is unresolved.

## 4 Data Collection

This research focuses exclusively on data from CNET forums.[1] Firstly, 1000 threads were crawled from CNET forums using SiteScraper.[2] We only collected threads that contained 2 to 16 posts, as threads containing only 1 post have no answers and cannot provide solutions, and long threads tend to be more discussion-oriented and/or contain multiple sub-threads.

[1] http://forums.cnet.com

[2] http://sitescraper.googlecode.com/

The crawled threads were then preprocessed. Only the title and sub-forum information of each thread, and the body, title, and author information of each post were preserved. Finally, we randomly selected 500 threads from 4 sub-forums of the CNET forums: Operating Systems, Software, Hardware, and Web Development.

Two annotators performed a pilot annotation using a seed set of 150 threads and a dedicated web annotation tool. The $\kappa$ value for the pilot annotation (indicating the relative agreement between the two annotators) was 0.43. The annotators sat down together to go over every thread where there were disagreements, and discussed the disagreements based on the class descriptions. Then, the two annotators annotated 327 new threads, achieving a more respectable $\kappa$ value of 0.74. The annotators furthermore met again to resolve any disagreements in the labelling of the 327 threads. Most of the disagreements arose from confusion between Hardware and Media in the PROBLEM set, and Documentation and Support in the SOLUTION set.

## 5 Experimental Methodology

We carried out preliminary experiments over the annotated data, focusing on the implications of the modular class design for thread classification.

As our feature representation, we firstly removed all punctuation in the threads and normalised the threads to lower case. Then, we lemmatised the threads using the GENIA Tagger (Tsuruoka et al., 2005), and removed stopwords.[3] Based on the preprocessed threads, we used a bag-of-words term frequency representation, concatenating all posts in the thread into a single meta-document and thereby treating the task as a document categorisation task.

All of our experiments were carried out using Hydrat (Lui and Baldwin, 2009), a classifier comparison framework. Hydrat integrates several machine learning software packages including BSVM (Hsu and Lin, 2006), weka (Hall et al., 2009) and MALLET (McCallum, 2002), in addition to native implementations of a number of more basic learners. In our experiments, we tried a range of machine learning models including Support Vector Machines (SVM), multinominal Naive Bayes (NB), and instance-based learners

[3] Using the stop word list from InfoMap (http://infomap-nlp.sourceforge.net/).

| Class Category | | Description |
|---|---|---|
| PROBLEM: | *OS* | Operating system |
| | *Hardware* | Core computer components, including core external components (e.g. a keyboard) |
| | *Software* | Software-related issues, including applications and programming tools |
| | *Media* | Hardware which is either a non-standard external component or peripheral device |
| | *Network* | Network issues (e.g. connection speed, and installing a physical network) |
| | *Programming* | Coding and design issues relating to programming |
| SOLUTION: | *Documentation* | How to use a certain function, select a computer/component, or perform a task |
| | *Install* | How to install a component |
| | *Search* | Search for a particular component (e.g. a software package) |
| | *Support* | How to fix a problem with a computer or component |
| MISC: | *Other* | Troubleshooting-related, but the problem source is not included in the PROBLEM set |
| | *Spam* | The thread is not troubleshooting-related |

Table 1: The components of the thread class set

(NN). A majority-class model (ZEROR) was used as the baseline.

The class set was represented in three ways, based on its two orthogonal components: (1) all 26 multiclasses (ALLCLASS); (2) only the PROBLEM class sub-set, the *Other* class and the *Spam* class, comprising 8 classes in total (PROBLEM); and (3) only the SOLUTION class sub-set, the *Other* class and the *Spam* class, comprising 6 classes in total (SOLUTION). By combining the outputs of classifiers based on the PROBLEM and SOLUTION class sub-sets (i.e. class composition), it is possible to construct full ALLCLASS classes, and we additionally compare the single-pass multiclass classification strategy with multi-pass class composition.

All experiments were carried out based on stratified 10-fold cross-validation. The results were evaluated via both micro-statistics and macro-statistics. Micro-statistics describe average performance *per instance* (i.e. thread), as represented in the micro-averaged precision ($\mathcal{P}_\mu$), recall ($\mathcal{R}_\mu$) and F-score ($\mathcal{F}_\mu$). Macro-statistics, on the other hand, describe average performance *per class*, as represented in the macro-averaged precision ($\mathcal{P}_M$), recall ($\mathcal{R}_M$) and F-score ($\mathcal{F}_M$). It should be noted that the $\mathcal{P}_\mu$, $\mathcal{R}_\mu$ and $\mathcal{F}_\mu$ are always the same, as the prediction per document is always unique. Moreover, because cross-validation is used, the averaged $\mathcal{F}_M$ is not necessarily the harmonic mean of the averaged $\mathcal{P}_M$ and $\mathcal{R}_M$.

Because we were more interested in the classification effectiveness per thread, the micro-averaged F-score ($\mathcal{F}_\mu$) was used as our primary evaluation method. We also tested the statistical significance of the results using randomised estimation with $p < 0.05$ (Yeh, 2000).

| Class Space | Learner | $\mathcal{P}_M$ | $\mathcal{R}_M$ | $\mathcal{F}_M$ | $\mathcal{P}_\mu/\mathcal{R}_\mu/\mathcal{F}_\mu$ |
|---|---|---|---|---|---|
| ALLCLASS | ZEROR | .006 | .018 | .009 | .038 |
| | SVM | .268 | **.248** | **.246** | **.382** |
| | NB | **.306** | .211 | .182 | .333 |
| PROBLEM | ZEROR | .038 | .142 | .060 | .266 |
| | SVM | .564 | **.485** | **.500** | .661 |
| | NB | **.574** | .483 | .481 | **.691** |
| SOLUTION | ZEROR | .122 | .168 | .140 | .304 |
| | SVM | .500 | **.387** | **.413** | **.575** |
| | NB | **.513** | .270 | .246 | .520 |

Table 2: The performance of different learners over ALLCLASS, PROBLEM and SOLUTION

## 6 Results and Evaluation

We performed a series of experiments by applying the learners described in Section 5 over the three class sets (i.e. ALLCLASS, PROBLEM and SOLUTION). Because NN performed significantly below the other two learners in all experiments, we only present results from SVM and NB (along with baseline ZEROR). The performance of different learners over ALLCLASS, PROBLEM and SOLUTION is shown in Table 2. For each class space, the best result for each column is presented in **boldface**.

There are several things to note in the results presented in Table 2. First, we can see that the majority class (ZEROR) results are quite poor, especially for ALLCLASS. This is due to the effects of cross-validation, in learning the majority class from the training data in each fold, but due to relative class uniformity, often finding that this is not the majority class in the test data. Second, SVM has relatively strong performance over all three tasks, especially in ALLCLASS and SOLUTION with the best $\mathcal{F}_\mu$ scores. This is not surprising, because it is often reported that SVMs have superior performance in document categorisation tasks (Yang and Liu, 1999; Joachims, 1998). However,

| PROBLEM | SOLUTION | ALLCLASS Results | | | |
|---|---|---|---|---|---|
| Learner | Learner | $\mathcal{P}_M$ | $\mathcal{R}_M$ | $\mathcal{F}_M$ | $\mathcal{P}_\mu/\mathcal{R}_\mu/\mathcal{F}_\mu$ |
| SVM | SVM | .345 | **.313** | .314 | .434 |
| NB | SVM | **.379** | .310 | **.316** | **.443** |
| SVM | NB | .278 | .259 | .229 | .398 |
| NB | NB | .268 | .247 | .206 | .398 |

Table 3: Results for class composition of the separate predictions from the PROBLEM and SOLUTION classifiers

it is interesting to note that NB produces the best $\mathcal{F}_\mu$ (i.e. 0.691) in the PROBLEM task. Although this figure is not significantly better than the $\mathcal{F}_\mu$ (i.e. 0.661) from SVM, it still may imply that we should optimise our methodology over each subtask.

As is explained in Section 5, the main purpose of the experiments is to examine whether the modular class set (i.e. PROBLEM and SOLUTION as two orthogonal components of the overall ALLCLASS) has the potential to benefit the classification task for ALLCLASS. One simple way is to explore class composition. To be specific, as PROBLEM and SOLUTION represent orthogonal components of ALLCLASS, it is possible to perform classification separately over PROBLEM and SOLUTION, and compose predictions via a combined class set, to form the ALLCLASS class set. For example, if a given thread is predicted to have a PROBLEM class of *Hardware* and a SOLUTION class of *Documentation*, we can compose the two predictions into the *Hardware/Documentation* class. In order to map the results back onto the ALLCLASS class set cleanly, we used the combined class set, where the combination of *Other* from either PROBLEM or SOLUTION with any other class from the second class set produces an overall classification of *Other* in the ALLCLASS set, and the combination of *Spam/Spam* is treated as *Spam*.

The combined results for the ALLCLASS class set are presented in Table 3, with the best outcome for each column once again indicated in **boldface**. From the results we can see that the composition of NB for PROBLEM and SVM for SOLUTION yields the best $\mathcal{F}_M$ (i.e. 0.316) and $\mathcal{F}_\mu$ (i.e. 0.443), significantly improving over the best ALLCLASS results from Table 2 (0.246 and 0.382 respectively). It would therefore appear to be the case that class composition is effective in boosting overall classification performance.

## 7 Conclusion

This research is aimed towards improving information access over troubleshooting-oriented technical user forum data, focusing on automated thread-level analysis of the problem sources and solution types. As first steps in this direction, we designed a modular thread-level class set, annotated 327 threads, and performed thread classification over the data. We proposed a class composition strategy by first performing classification separately over the PROBLEM and SOLUTION class sets, and composing the predictions into an overall thread classification. This approach gives us the best classification performance overall, with an $\mathcal{F}_\mu$ of 0.443, well above the best result from doing the ALLCLASS classification directly.

Much more work could be done in terms of feature engineering. This could include new features such as author name/profile and the number of posts in the thread. We also speculate that noise in the threads, such as typos and incorrect casing/punctuation, reduced overall performance, suggest that text normalisation may help boost our classifiers. Additionally, because of the promising results produced by the class composition strategy and the innate structure of our thread class set, we could consider more sophisticated hierarchical classification methods (Dekel et al., 2004; Tsochantaridis et al., 2005). We leave these for future work.

### Acknowledgements

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

## References

Timothy Baldwin, David Martinez, and Richard B. Penman. 2007. Automatic thread classification for Linux user forum information access. In *Proceedings of the 12th Australasian Document Computing Symposium (ADCS 2007)*, pages 72–79, Melbourne, Australia.

Timothy Baldwin, David Martinez, Richard Penman, Su Nam Kim, Marco Lui, Li Wang, and Andrew MacKinlay. 2010. Intelligent Linux information access by data mining: the ILIAD project. In *Proceedings of the NAACL 2010 Workshop on Computational Linguistics in a World of Social Media: #SocialMedia*, pages 15–16, Los Angeles, USA.

Ofer Dekel, Joseph Keshet, and Yoram Singer. 2004. Large margin hierarchical classification. In *Pro-*

*ceedings of the 21st International Conference on Machine Learning (ICML 2004)*, Banff, Canada.

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

Chih-Wei Hsu and Chih-Jen Lin. 2006. BSVM. `http://www.csie.ntu.edu.tw/~cjlin/bsvm/`.

Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning (ECML'98)*, pages 137–142, Chemnitz, Germany.

Marco Lui and Timothy Baldwin. 2009. hydrat. `http://hydrat.googlecode.com`. Retrieved on 25/10/2010.

Andrew Kachites McCallum. 2002. MALLET: A machine learning for language toolkit. `http://mallet.cs.umass.edu/`.

Ani Nenkova and Amit Bagga. 2004. Facilitating email thread access by extractive summary generation. In Nicolas Nicolov, Kalina Bontcheva, Galia Angelova, and Ruslan Mitkov, editors, *Recent Advances in Natural Language Processing III: Selected Papers from RANLP 2003*, pages 287–294. John Benjamins, Amsterdam, Netherlands.

Owen Rambow, Lokesh Shrestha, John Chen, and Chirsty Lauridsen. 2004. Summarizing email threads. In *Proceedings of the 4th International Conference on Human Language Technology Research and 5th Annual Meeting of the NAACL (HLT-NAACL 2004)*, pages 105–108, Boston, USA.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. 2005. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6(Sep):1453–1484.

Yoshimasa Tsuruoka, Yuka Tateishi, Jin-Dong Kim, Tomoko Ohta, John McNaught, Sophia Ananiadou, and Jun'ichi Tsujii. 2005. Developing a robust part-of-speech tagger for biomedical text. In *Proceedings of the Advances in Informatics - 10th Panhellenic Conference on Informatics, LNCS 3746*, pages 382–392, Volos, Greece.

Stephen Wan and Kathy McKeown. 2004. Generating overview summaries of ongoing email thread discussions. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*, pages 549–555, Geneva, Switzerland.

Yiming Yang and Xin Liu. 1999. A re-examination of text categorization methods. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99)*, pages 42–49, Berkeley, USA.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*, pages 947–953, Saarbrücken, Germany.

Liang Zhou and Eduard Hovy. 2005. Digesting Virtual "Geek" Culture: The Summarization of Technical Internet Relay Chats. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*, pages 298–305, Ann Arbor, USA.