Automatic Thread Classification for Linux User Forum Information Access

Timothy Baldwin, David Martinez, Richard B. Penman
CSSE
University of Melbourne
VIC 3010 Australia
`{tim,davidm,rbp}@csse.unimelb.edu.au`

Abstract *We experiment with text classification of threads from Linux web user forums, in the context of improving information access to the problems and solutions described in the threads. We specifically focus on classifying threads according to: (1) them describing a specific problem vs. containing a more general discussion; (2) the completeness of the initial post in the thread; and (3) whether problem(s) in the initial post are resolved in the thread or not. We approach these tasks in both classification and regression frameworks using a range of machine learners and evaluation metrics.*

Keywords Web Documents, Document Management

1 Introduction

Due to the sheer scale of web data, simple keyword matching is an effective means of information access for many informational web queries. There still remain significant clusters of information access needs, however, where keyword matching is less successful, due to a combination of factors including: overly-specific information needs (e.g. as specified in technical queries such as `kswapd0 hogs cpu "Slackware 10.0" "windows XP" vmware "no swapping"`); low density of relevant documents (e.g. for monolingual queries in low-density languages such as Uighur, or over domains with little web presence); the inability of keyword queries to handle data ranges (e.g. `Dell laptop $1000-1500`); and information streams spanning multiple documents, with no complete description of the contained information in any one of the documents (e.g. as occurs in logs of mailing lists). While query expansion and document normalisation can go some way towards ameliorating the first two effects [8, 2], the third question of querying and reasoning over data ranges tends to point to the need for some form of information extraction and semantic normalisation of the document content, and the final question of document segmentation requires some form of meta-document identification or threading.

Proceedings of the 12th Australasian Document Computing Symposium, Melbourne, Australia, December 10, 2007. Copyright for this article remains with the authors.

In this paper, we are concerned with information access in the domain of Linux troubleshooting, based on Linux web user forum data. Consider the following scenario:

> *Kim, a Debian GNU/Linux user, notices that as a result of the latest upgrade on her laptop, she can no longer start up the GNOME desktop environment. She goes to Google to troubleshoot the problem and tries inputting the version details of various X packages and the hardware particulars of her laptop, along with different combinations of keywords such as `broken`, `not working` and `won't start`; all of the top-ranking hits are either outdated and inapplicable to the latest version packages, or irrelevant to the task at hand. She checks the archives of a selection of relevant-sounding Debian mailing lists without luck. Finally after searching the web for 2 hours she stumbles across a series of pages describing an `error` with gtk and the method for correcting the problem.*

This example (based on real-world experience) is intended to illustrate the fact that, while web search engines such as Google are remarkably successful at locating individual documents/sites typifying a well-defined information type, they have shortcomings in: (1) tracking data streams spanning multiple documents as found, e.g., in mailing list archives; (2) identifying documents associated with particular (ranges of) distribution or package versions, or identifying minor vs. major version changes in a given package; and (3) predicting acceptable lexical variance between a query and a document (e.g. between `broken` and `error`).

Our proposed alternative to conventional (web-based) information retrieval over Linux data is the ILIAD (Improved Linux Information Access by Data Mining) system which trawls the main English-based Linux web forums throughout the world, analyses each thread to arrive at a conceptual representation specified for the package, version and system information, and pre-classified according to the particular problem type. Further, we aim to distill the evolution of the proposed solutions and diagnostics in a given thread into a single succinct list of factoids, and the various threads on the web pertaining to a particular problem into a single ranked list of possible solutions, with links to the original web data. Access to this information then takes the form of a web interface where the user selects the type of problem experienced (e.g. some component of a package is broken or the user wishes to configure a package in a particular way), the package or component type that is applied to (e.g. emacs or the X window system in general), and (optionally) the system and hardware configuration (e.g. Debian 3.1, or ALSA 1.1 on a ThinkPad X60), and the system returns information relevant to that query in a pre-distilled form for easy application. This paper describes the first tentative steps towards this goal, in performing thread-level estimation of the utility of a given thread for troubleshooting purposes.

2 Related work

Recently there has been growing interest in the automatic processing of discussion-based information sources, such as mailing lists or web forums. However, there is little work that focuses specifically on thread-level classification. Most related work in this area relies on speech acts to annotate utterances in the discussion, such as the approaches in [5] and [4]. Here, speech acts such as *question* and *elaboration* are detected and applied in different tasks, including identifying the roles of participants, finding unanswered threads, predicting what type of response would be appropriate in a given context, and question answering.

Another related line of research is on discussion summarisation, to provide only the most relevant information to the user. An example of this kind of work can be seen in [9], where a system is developed to summarise technical online IRC (Internet Rely Chat) discussions by clustering message segments and finding the most relevant parts using machine learning methods.

Regarding the classification of threads, [6] present an evaluation of automatic assessment of the post quality of online discussions of software. Their approach is related to our methods, and they rely on quality assessment from online users to train and test their classifiers. Using different types of features they are able to build systems with high performance for their task. Their motivation is also to deliver information to end-users in a more effective way.

3 Data description

There is a vast range of forums, newsgroups and mailing lists available on the web that cover different aspects of the Linux domain, from general discussions to very specific applications. For our purposes, we chose to target well-known, active forums, including sites that are specifically devoted to Linux (e.g. LINUXQUESTIONS and FEDORAFORUM) and general-domain resources that contain Linux-specific sections (e.g. EXPERTS EXCHANGE and GOOGLE GROUPS). In the interests of maximising the proportion of troubleshooting-related threads, we focused primarily on two main resources: Linuxquestions,[1] and a subset of the Debian mailing lists.[2] From those we further selected three specific subforums intended to include a representative range of both general-purpose and hardware/package-specific forums, namely Linuxquestions-software, Debian-amd64, and Debian-apache as our data sources.

We crawled the data from these forums using dedicated software tailored to the VBulletin[3] forum manager (for Linuxquestions) and the pipermail web log (for the Debian mailing lists). In order to extract the relevant information from the documents we analysed the table structure of the VBulletin forums and the header structure of the pipermail web logs. We required specific rules to identify quotations in the posts, which were represented with special codes as formatted text in XML. This software has also been used to crawl data from other resources (e.g. FEDORAFORUM).

Each of the forums was then stored in a MySQL relational database. We designed a common representation for the discussion lists, based on forums, threads, and posts:

Forum: the source forum, namely Linuxquestions-software, Debian-amd64 or Debian-apache.

Thread: each thread is linked to a given forum, and stored with a title, author, date, and flag (e.g. *sticky post*).

Post: each post is related to a thread and forum. We also register its title, position in the thread, date, which post it follows (for nested threads), raw text, and formatted text.

We filtered out threads containing only one post and those containing more than 14 answers to the original post. Obviously threads with no answers cannot provide problem solutions, and long threads tend to be more discussion related (e.g. discussing the relative merits of different distributions or packages) and/or contain too much information for accurate processing. For Linuxquestions we also removed those threads flagged as "sticky" because they usually contain polls and discussions.

After filtering, our data collection contains more than 380,000 posts spanning 90,000 threads. The detailed statistics of the crawled data from each of the three forums are given in Table 1. We can see that most of the data comes from the Linuxquestions forum,

[1]`http://www.linuxquestions.org`
[2]`http://lists.debian.org/completeindex.html`
[3]`http://www.vbulletin.com`

	Linuxquestions	Debian-apache	Debian-amd64	Overall
Years	2000-2007	2001-2006	2003-2006	
Threads	80,814	5,251	4,364	90,429
Posts	352,677	8,063	19,670	380,410
Users	42,622	3,484	2,305	48,411
Average posts per thread	4.36	1.54	4.51	4.21
Average posts per user	8.27	2.31	8.53	7.86

Table 1: Details of the forum data

which is the one with the most users. Also, we can notice a clear difference in the length of threads for the two Debian subforums.

4 Task descriptions

The first step in extracting information from the threads is to identify which threads are more relevant in the context of troubleshooting. There are different characteristics of a thread that can make it less useful, including: it may not refer to a specific problem; the thread may be unsolved; or the description of the problem may be unclear. For this work we detected 5 characteristics that are important in profiling the predicted troubleshooting utility of a given thread:

Task orientation: is the thread focused on solving a specific problem or devoted to a more general discussion on some topic?

Completeness: does the initial post include a sufficiently detailed specification of the problem for a third party to be able to realistically provide a solution?

Solvedness: is there a documented solution to the original problem described by the thread initiator in the thread (including the possibility of URLs pointing off to solutions elsewhere on that same forum or generally on the web)?

Spam: is the thread spam?

Problem type: free text keyword description of the type of problem described (e.g. *software installation*).

We hand-annotated a subset of the thread collection in order to get a better understanding of the data and train thread-level classifiers over. 250 threads were annotated independently by three annotators, in terms of the 5 thread-level features described above. The threads were chosen randomly from across the three forums. For the "Problem type", we allowed the annotators to either choose from a pull-down list of tags from previously-annotated threads, or alternatively to enter a free-text description for the current thread. This information is not used in the experiments described in this paper. Similarly, our simple filtering of threads based on the number of posts removed all spam from our 250 thread sample, such that the "Spam" task is also not a relevant task for the current dataset.

The annotators were asked to rate the "Task orientation", "Completeness" and "Solvedness" of each thread based on a five point scale, with a score of 1 indicating a high degree of fit with the given designation, and a score of 5 indicating a low degree of fit. The mean numeric value across the 3 annotators was used to derive the gold-standard value for each of the three tasks. For the majority of our experiments, we convert the numeric score into a binary value, using 2.5 as our breakpoint. The annotation process took between 5.7 and 8.7 hours depending on the annotator.

We measured the inter-tagger agreement for the 3 different tasks using the Spearman rank correlation and the Kappa statistic. Rank correlation was calculated between each annotator and the mean value, and for the Kappa statistic we converted the 5-way scores into 3-way values (positive, neuter and negative) and computed the agreement between the 3 annotators. The reason for calculating the rank correlation in addition to the Kappa statistic is that Kappa is incapable of capturing the fact that the rating scale is ordinal, and treats a 4 vs. 5 rating difference equally to a 1 vs. 5 rating difference, where there is clearly higher agreement in the first instance; rank correlation is better equipped to model this difference, by viewing the rating according to its relative position in the ranking rather than the raw numbers.

Table 2 shows the inter-tagger agreement for the three tasks.

We can see that the Kappa values are low, indicating that there is low agreement relative to the label bias of the task, and that the classifiers may have problems with the three-way classification task. However, the high rank correlation shows that, while the individual annotators may have interpreted the raw numbers on the 5-point scale differently, there is general consistency in terms of what they considered to be more or less task oriented, complete or solved. We approach the task in both classification and regression terms in order to explore both the "direct hit" discrete view of the data as performed by the annotators, and the ordinal view of the data represented in the ranking (mapping the real-valued outputs of the regression model onto a ranking).

Test	Task Orientation	Completeness	Solvedness
Rank Correlation A1/mean	0.94	0.86	0.71
Rank Correlation A2/mean	0.94	0.81	0.82
Rank Correlation A3/mean	0.90	0.85	0.79
3-way kappa	0.64	0.21	0.38

Table 2: Agreement statistics: rank correlation between each annotator and the mean, and the 3-way Kappa statistic

5 Features

In order to represent the threads for our classifiers and regression models, we defined two common sets of features to use for all three tasks. As the baseline feature set we constructed simple bag of words (BoW) features, consisting of all the words occurring in each of the threads. The second feature set (THREAD) relies on the thread structure and specific contextual features for the representation. We identified four subparts of the thread that appear to contain different types of information, and are expected to impact differently on each of three thread classification tasks:

Initial Post (INITIALPOST): the first post of the thread, which contains the original problem description; we also include the immediately proceeding posts (based on the chronological order of the posts) if they were authored by the same user, as they usually constitute clarifications or supplementary information. This part of the thread is expected to be particularly relevant to both the “Task Orientation” and “Completeness” tasks.

First Response (FIRSTRESPONSE): the first post in the thread belonging to a user other than the initiator; expected to be relevant for all three tasks.

All Responses (ALLRESPONSES): the remaining posts in the thread after removing the two sets above; expected to be relevant for the “Task Orientation” and “Solvedness” tasks.

Final Post from the Initiator (FINALPOSTINIT): the final (sequence of) post(s) from the thread initiator after the initial post, i.e. in response to a post from someone else; expected to be particularly relevant for the “Solvedness” task.

From each of these sets we extracted a group of 18 lexical and contextual features, as defined in Table 3. Below, we denote each of these 4 threads subparts with the indicated acronyms (i.e. INITIALPOST, FIRSTRESPONSE, ALLRESPONSES and FINALPOSTINIT), and the union of these groups as THREAD.

Apart from the intra-thread features, there are other forum-level features that are indicative of the rating of a given thread according to the three classification tasks, such as the particular users that initiate/respond to the threads, or the number of external links pointing to the target thread. We plan to analyse these feature types in the future.

6 Machine Learners

In our experiments, we used three different machine learning software packages to build our classifiers and regression models:

SVMLIB [1]: an integrated toolkit for support vector machine classification, regression, and distribution estimation (one-class SVM). For our experiments we used the classification and regression modules, relying on an RBF kernel in each case.

TIMBL [3]: a fast implementation of a number of variants of the k-nearest neighbour algorithm, out of which we use an extended version of IB1.

WEKA [7]: a suite of machine learning and regression algorithms for data mining tasks. From the suite of algorithms, in our experiments we applied the following classification algorithms: JRIP, J4.8 decision trees, support vector machine with a linear kernel (SVM), naive Bayes (NB), ADABOOSTM1, BAGGING and STACKING; and the following regression algorithms: linear regression (LM) and PERCEPTRON.

7 Evaluation

In this section, we describe our experiments on the three different thread classification tasks. First we present our experimental setting, and then we analyse the results of the different feature sets for two classifiers. Next, we report on the classification accuracy of the different models in binary classification terms, and also rank correlation for the regression models. Finally, we present learning curves to shed light on the relationship between the amount of training data and our relative results.

7.1 Experimental setting

For all our experiments on the 250 thread dataset, we performed stratified 10-fold cross-validation. For the main classification tasks we binarised the 1-5 values given by the three annotators into positive and negative instances relying on the mean of the scores. We decided to remove the few cases where the mean of the three annotators’ ratings was exactly at the midpoint of the 5-point scale (i.e. 3). This left us with 244 threads for the

Feature Description	Feature Type
Number of posts	Integer
Number of Linux distribution mentions (e.g. *redhat*)	Integer
Number of beginner keywords mentioned (e.g. *noob*)	Integer
Number of emoticons detected (e.g. *:-)*)	Integer
Number of version numbers mentioned (e.g. *version 5.1*)	Integer
Number of URLs detected (e.g. *www.ubuntu.org*)	Integer
Proportion of words relative to full thread	Real
Proportion of sentences relative to full thread	Real
Proportion of question sentences in set	Real
Proportion of exclamation sentences in set	Real
Proportion of simple declarative sentences in set	Real
Proportion of code sentences in set	Real
Proportion of other sentences in set	Real
Average sentence length	Real
Average word length	Real
Proportion of sentences to first question	Real
Proportion of thread posts to first class post	Real
Proportion of thread posts to last class post	Real

Table 3: Lexical and contextual features extracted for each of the 4 thread subparts

"Task Orientation" task, 232 for the "Completeness" task, and 222 for the "Solvedness" task. For the rank-correlation evaluation we used all 250 threads for each task.

We measured accuracy values for the classification experiments. As our baseline we used a majority class (MC) classifier that assigns the majority class in the training data to all test instances.

7.2 Feature sets

For our first experiment we analysed the performance of binary classifiers over the different feature sets. This gives us an indication of whether the THREAD features provide an enhancement over the classic BOW model. We further contrast this with the combination of the two feature sets (All) to determine whether they are complementary. At this stage of our experiments, we focused exclusively on two learners: TIMBL and SVMLIB.

The results for TIMBL are shown in Table 4. We can see that the contribution of the feature sets differs across the three tasks, but that the performance is lower than our MC baseline in most cases. The most significant drop occurs for BOW in the "Solvedness" task, while the other feature sets are more stable across different tasks.

The results for SVMLIB are given in Table 5. Here the system performs at the same level as the MC baseline in most cases. There is a slight increase is the "Solvedness" task performance for some of the feature sets, and a small decrease for the THREAD feature set, but overall, the classifiers mirror the performance of the baseline.

Looking over the results for the three tasks across the different feature sets, we find that with TIMBL there is partial agreement with our expectations of which thread subparts would be most predictive of the three classification tasks. We also can see that particularly for the "Solvedness" task, the THREAD features provide useful information not present in the simple BOW features, or at the very least that the performance with the THREAD features is more consistent than with the BOW or All features. For the remainder of our experiments, therefore, we use the THREAD features exclusively.

7.3 Binary Classification and Rank Correlation

For our next experiment we applied the extra learners from the WEKA toolkit to the THREAD feature set to compare their performance with that of TIMBL and SVMLIB. The results of these experiments are given in Table 6. We can see that the overall results are low, and that in most cases the simple MC baseline obtains the highest result. Only for the "Solvedness" task do we achieve above-baseline results, using JRIP and STACKING.

We analysed the cause of the unexpectedly low performance of our classifiers, looking first at the binarisation of the tasks. In the annotation phase we observed low Kappa scores among the annotators, but considerably higher rank correlation. This suggests that the relative ordering of the ratings for the annotators is consistent but there is disagreement in the absolute numbers they use. Thus, the binarisation of the task could be one of the causes of the low performance of the classifiers, as the break point between classes may be hard to determine.

Another way to process the threads is to establish a ranking using regression models or classification weights. This allows us to measure the rank correlation

Features	Task Orientation	Completeness	Solvedness
MC	0.816	**0.948**	**0.765**
BoW	**0.820 (0.032)**	0.935 (0.021)	0.279 (0.171)
InitialPost	0.713 (0.072)	0.918 (0.039)	0.703 (0.077)
FirstResponse	0.746 (0.106)	0.905 (0.037)	0.685 (0.069)
AllResponses	0.734 (0.080)	0.909 (0.030)	0.734 (0.072)
FinalPostInit	0.668 (0.104)	0.879 (0.045)	0.667 (0.073)
Thread	0.762 (0.064)	0.914 (0.033)	0.653 (0.058)
All	**0.820 (0.031)**	0.935 (0.028)	0.401 (0.147)

Table 4: Accuracy for TiMBL using the different feature sets (standard deviation given in parentheses; best results are shown in **bold**)

Features	Task Orientation	Completeness	Solvedness
MC	**0.816**	**0.948**	0.765
BoW	**0.816** (0.027)	**0.948** (0.016)	0.779 (0.026)
InitialPost	**0.816** (0.018)	**0.948** (0.016)	0.779 (0.026)
FirstResponse	**0.816** (0.025)	**0.948** (0.016)	0.779 (0.036)
AllResponses	**0.816** (0.026)	**0.948** (0.016)	0.779 (0.031)
FinalPostInit	**0.816** (0.024)	**0.948** (0.016)	**0.788** (0.031)
Thread	0.807 (0.026)	**0.948** (0.016)	0.757 (0.041)
All	**0.816** (0.024)	**0.948** (0.016)	0.779 (0.031)

Table 5: Accuracy for SVMLib using the different feature sets (standard deviation given in parentheses; the best results are shown in **bold**)

between the gold standard values and the predictions. We performed these experiments by applying two regression models (linear regression and SVM) and two classifiers with weighted predictions. We tested different kernel types for SVM with small differences in performance, and report here the results for linear kernels. We measured the Spearman rank correlation between the gold standard weights and the predicted values, and present the results in Table 7.

The results are still lower than the simple MC baseline. Linear regression performs better than the other approaches, but still significantly below the baseline. Therefore the low correlation suggests that determining the boundary is not the only problem with this dataset.

7.4 Learning Curves

In this analysis we explored whether the hand-annotated data was sufficient in quantity to successfully train the classifiers. Thus, we tested the effect of using different amounts of training data, ranging from 20% to 100% of the total training set. We present the results for the SVMLib classifier and the linear regression model from Weka. The performance for SVMLib is given in Figure 1. We see that there is not a clear improvement in the results as the quantity of training data increases. This suggests that even with more data we do not expect much success from SVMLib for the different tasks.

The linear regression results are given in Figure 2. In this case the introduction of more training data in-

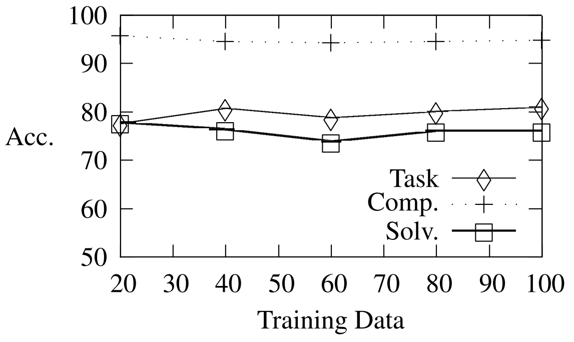

Figure 1: Learning curve for SVMLib

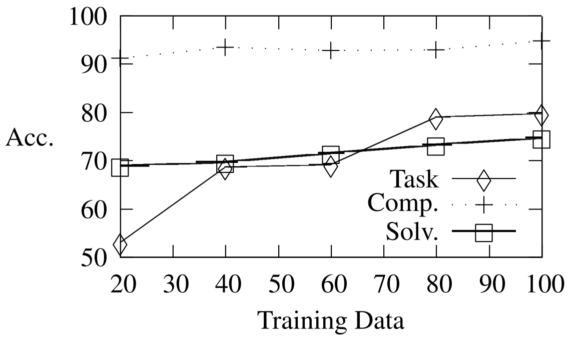

Figure 2: Learning curve for linear regression

creases the accuracy of the system. This indicates that the regression model could benefit from extra annotated data.

System	Task Orientation	Completeness	Solvedness
MC	**0.816**	**0.948**	0.765
TiMBL	0.762	0.914	0.653
SVMlib	0.807	**0.948**	0.757
JRip	0.807	**0.948**	0.770
J4.8	0.717	**0.948**	0.685
SVM	0.783	0.940	0.765
Perceptron	0.729	0.909	0.721
NB	0.734	0.741	0.641
LM	0.791	0.944	0.739
AdaBoostM1	0.799	0.931	0.730
Bagging	0.811	**0.948**	0.766
Stacking	**0.816**	**0.948**	**0.779**

Table 6: Accuracy for all classifiers using Thread feature-set (best result per column shown in **bold**)

System	Task Orientation	Completeness	Solvedness
MC	**0.54**	**0.53**	**0.51**
LM	0.14	0.40	0.37
SVM-regression	0.12	0.32	0.25
SVM-classification	0.18	0.10	0.18
JRip	0.08	0.22	0.04

Table 7: Spearman rank correlation between goldstandard and different models (best result per column shown in **bold**)

8 Discussion

The empirical results showed the problems in both the classification and regression frameworks. We observed that our annotation scheme produced low Kappa agreement, and this reflected on the performance of the classifiers, causing them to perform at or below the baseline. We had higher hopes in the regression experiments, because the rank correlation between annotators was higher, but the regression models did not show much improvement.

In order to test the classifiers in the same setting as the original annotation, we also explored the results of classifiers on 5-way classification. We observed that in this case the performance was clearly above the MC baseline for the "Completeness" and "Solvedness" tasks. This classification would be potentially useful to help rank threads to present to the user. This 5-way annotation task highlighted the problem of the boundary (i.e. mid-range) cases, which were difficult to detect for the classifier and contributed most of the errors.

Another problem that was not addressed in the experiments was the relationship between the different tasks. For instance, if a thread is considered discussion oriented (in terms of the "Task Orientation" task), the annotation of "Solvedness" should be different from a specific-task oriented thread. We performed an experiment to measure the results using different subsets of data from the gold standard annotation (e.g. measure "Solvedness" and "Completeness" in task-oriented threads only). The results show only slight increases over the MC baseline, suggesting that partitioning the data in a hierarchical approach would not completely solve the problem.

All in all the experiments show the complications in automatising these tasks. The results indicate that automatic methods can be developed to identify the most clear-cut cases and provide rankings of threads, but for most cases the results are below the MC baseline. It is worth mentioning that the baseline is very high (94.8%) in the case of "Completeness", and this makes it difficult to improve over. The majority class is also high for the other two tasks (77.9% and 81.6%).

9 Conclusion

In this work we present the ILIAD project for information delivery in the Linux domain, and the first exploration of utility-based classification of threads from different sources. For our main goal of providing better information-access tools, we developed software to crawl and normalise data from different mailing lists and forums. We used these tools to create a collection of more than 90,000 threads addressing different problems.

After analysis of a sample of threads, we identified some relevant characteristics that could be useful to deliver the information. We defined the "Task orientation", "Completeness", and "Solvedness" as the object of study for this paper, and annotated a subset of threads on their having these characteristics. We then developed automatic methods to classify threads according to these dimensions.

Our experiments highlighted some of the problems to carry out these tasks automatically. The low Kappa agreement between annotators lead to difficulties to train systems for binary classification. However, there was higher rank correlation, and this suggests that we can develop systems to rank threads according to their characteristics.

We are currently working on extending our feature set to include other types of features, such as the authors of the posts and the external links pointing to target threads. We are also applying and evaluating these techniques in an IR framework. Our aim in this task is to measure the contribution of devoted tools over general keyword-based search. For future work we plan to develop tools to process the content of the threads, and extract the relevant information for delivery.

Acknowledgements This research was carried out with support from Australian Research Council grant no. DP0663879.

References

[1] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[2] Hang Cui, Ji-Rong Wen, Jian-Yun Nie and Wei-Ying Ma. Probabilistic query expansion using query logs. In *Proceedings of the 11th International Conference on the World Wide Web*, pages 325–332, Honolulu, USA, 2002.

[3] Walter Daelemans, Jakub Zavrel, Ko van der Sloot and Antal van den Bosch. Timbl: Tilburg memory based learner, version 5.1, reference guide. Technical report, ILK University of Tilburg, 2004.

[4] Edward Ivanovic. Dialogue act tagging for instant messaging chat sessions. In *Proceedings of the ACL Student Research Workshop*, pages 79–84, Ann Arbor, USA, 2005.

[5] Jihie Kim, Grace Chern, Donghui Feng, Erin Shaw and Eduard Hovy. 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*, Athens, USA, 2006.

[6] Markus Weimer, Iryna Gurevych and Max Mühlhäuser. Automatically assessing the post quality in online discussions on software. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 125–128, Prague, Czech Republic, 2007.

[7] Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. 2nd Edition, Morgan Kaufmann, San Francisco, USA, 2005.

[8] Jinxi Xu and W. Bruce Croft. Query expansion using local and global document analysis. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 4–11, Zurich, Switzerland, 1996.

[9] Liang Zhou and Eduard Hovy. Digesting virtual geek culture: The summarization of technical internet relay chat. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 298–305, Ann Arbor, USA, 2005.