Detecting Bot-Answerable Questions in Ubuntu Chat

David C. Uthus
NRC/NRL Postdoctoral Fellow
Washington, DC 20375
duthus@google.com

David W. Aha
Navy Center for Applied Research in AI
Naval Research Laboratory (Code 5514)
Washington, DC 20375
david.aha@nrl.navy.mil

Abstract

Ubuntu's Internet Relay Chat technical support channel has bots that output specific messages in response to command words from other channel users. These messages can be used to answer frequently-asked questions instead of requiring an expert to (repeatedly) type a lengthy reply. We describe an approach to automatically distinguish bot-answerable questions, which would mitigate this problem. To the best of our knowledge, this is the first work on investigating question answering in a multiparticipant chat domain. Our results indicate that for some types of questions, supervised learning algorithms perform well on this task and, in addition, that character n-grams are a better representation than traditional bag-of-words for this task and domain.

1 Introduction

Ubuntu (a Linux-based operating system) maintains multiple Internet Relay Chat (IRC) channels for technical support. Some of these channels contain bots, which are automated agents pre-programmed to perform certain tasks. One of the bots can output pre-written messages, called *factoids*, in response to command words. For example, if a user types "!flash", then the bot would output "To install Flash see https://help.ubuntu....mats/Flash - See also !Restricted and !Gnash". These factoids are used to answer common questions, enforce channel guidelines, direct non-English speakers (in their native tongue) to Ubuntu's foreign language support channels, and query Ubuntu's repository of packages. While useful, this bot must be manually invoked. Automating the bot to self-detect and answer questions that it can answer could reduce the workload for knowledgeable experts trying to help other users. This is applicable to not only Ubuntu's technical support channels, but to other IRC channels providing technical support (e.g., Debian's support channels) and to channels that use similar bots (e.g., Eggdrop and Infobot) for other purposes.

We describe initial steps on a self-invoking bot. We begin by investigating the multi-classification task of which questions are bot-answerable questions (BAQ) and which are human-answerable questions (HAQ), which requires a human to answer due to the question's complexity. We implemented a baseline non-learning approach and supervised support vector machines (SVM) and k-nearest neighbor (k-NN) algorithms. Our results show that a bot can answer with confidence some types of questions, especially those directing users to more appropriate channels for help on certain topics.

Our contributions are as follows:

- Problem: We identified a real-world problem that has not been investigated, despite bots having been around for years on IRC channels.
- Data: We created an annotated multiparticipant chat corpus that is publicly available.
- Empirical study: We report on our investigation of applying supervised learning algorithms and leveraging different feature representations, whose results will be used as a foundation for a larger case-based reasoning approach.
- Discussion: We describe how some types of automatically-answerable questions can be easy or difficult to classify.

2 Related Work

Chat is a difficult medium to analyze: its characteristics make it difficult to apply traditional natural language processing techniques. It has uncommon features such as frequent use of abbreviations, acronyms, missing subject pronouns, emoticons, abbreviated nicknames, words stripped of vowels to reduce number of keystrokes, and entangled conversation threads (Uthus and Aha, 2013a).

In the multiparticipant chat domain, there has been some work in creating a bot that can answer questions with Cobot (Isbell et al., 2006). This bot was limited in capability – it could only respond to questions directed at it. Another recent work resulted in a bot which could respond to utterances through word matching and used templates for output (Shaikh et al., 2010).

Also related in this domain are a few military research efforts that have focused on classifying chat messages. One examined profile-driven information extraction from chat using regular expressions and entity classes (Berube et al., 2007). Another examined identifying uncertainty and urgency within a chat message using rule-based approaches and statistical analysis (Budlong et al., 2009). A third, whose work is most similar to ours, compared several supervised algorithms for classifying chat utterances (Dela Rosa and Ellen, 2009). Using an artificial chat log, they classified messages as either non-important filler messages or as messages containing Navy ship updates. Their results showed k-NN and SVM performed best for this task. Our task differs from these previous investigations in that we are applying supervised learning algorithms to a multi-labeled corpus composed of real chat messages.

This problem is also related to the larger field of question answering, such as pertaining to discussion boards (Hong and Davison, 2009; Kim et al., 2007), frequently asked question files (Burke et al., 1997), and community-based question answering (Zhou et al., 2012). An important difference between this body of related work and what we are investigating is the medium. Multiparticipant chat is more difficult to work with compared to other mediums due to entangled conversation threads – a researcher cannot easily automatically analyze the messages of a single conversation. In prior work, researchers could usually isolate individual conversations automatically, making it possible to identify (to some extent) the question and the answer. Another important difference is the temporal scale – chat is in real-time, and a chat user expects to receive an answer quickly. A chat user can only see messages while they are logged in (in the case where there is no archive being stored offline). Both of these differences results in new challenges not seen in other mediums for question answering.

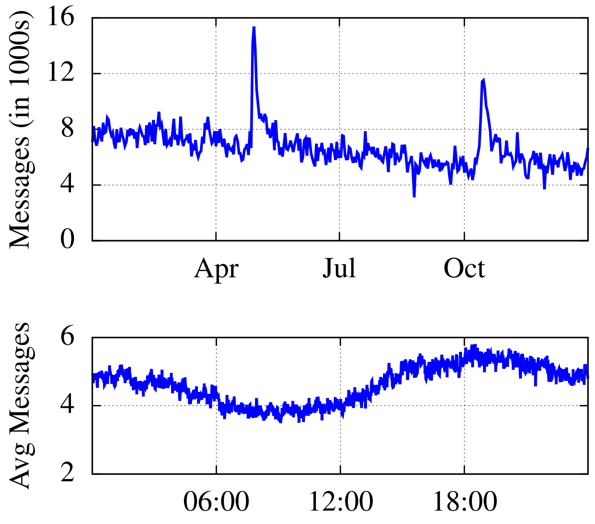

Figure 1: The volume of messages in the IRC channel `#ubuntu` during 2011.

3 Ubuntu's IRC Channels

The IRC channel `#ubuntu` is Ubuntu's primary technical support channel. It provides support for those who have problems using Ubuntu; it is not used for socializing or for receiving help with other Linux distributions (e.g., Debian, Linux Mint, Fedora) or software.

The channel's traffic level varies throughout the year and day (see Figure 1). During the year, it experiences heavy traffic during Ubuntu's semiannual new releases in April and October and generally experiences heavy traffic during the North American and European evening hours. Heavy traffic creates difficulties for users trying to get answers to their questions.

`ubottu`[1] is the bot that can access 1234 factoids, corresponding to 2324 commands (some factoids are mapped to multiple commands). It can also provide information about any software package found in Ubuntu's software repository. A channel user (oftentimes an expert) can task `ubottu` to answer another user's question (see

[1]`http://ubottu.com/`

```
[13:19] <p5yx> is the netbook
remix not available anymore?
[13:20] <histo> !unr | p5yx
[13:20] <ubottu> p5yx:  Starting
with Ubuntu 11.04, the Ubuntu
Netbook Edition is no longer
being offered as a separate
install as Unity is now standard
for all Ubuntu desktop installs.
```

Figure 2: Example of `ubottu` being invoked with a command word (in this case “!unr”) to answer a question.

Figure 2). Automating `ubottu` would allow experts to focus their valuable time on responding to more challenging requests.

4 Corpus

We created an annotated corpus by pulling questions from the Ubuntu Chat Corpus, specifically from the `#ubuntu` channel logs (Uthus and Aha, 2013b). This corpus has 4577 messages, including 2002 HAQs and 2575 BAQs from 68 factoid categories. These messages were taken from chat logs from 28 April 2011 (the day Ubuntu 11.04 was released) to 13 October 2011 (the day before Ubuntu 11.10 was released).

We looked for messages in which a question is answered with a factoid, or a question required a human to answer. To judge between these two types, we relied on the expertise of users and how they answered the questions. To reduce noise, we limited HAQs to conversations in which the first reply came from a user who invoked `ubottu` frequently (i.e., experts). These `ubottu` invokers are considered a better judge of what is a BAQ or HAQ compared to someone who rarely invokes `ubottu` to answer questions. For the BAQs, we restricted questions to those with at least ten examples mapped to a factoid. Figure 3 shows the distribution of BAQs to factoids in our corpus.

Some of the corpus’ messages are not in English. In such cases, users will be directed to one of Ubuntu’s foreign-language support channels (though a user can re-ask their question in English). Some languages present in the corpus include Spanish, French, Chinese, Russian, German, Polish, and Portuguese. Additionally, some of these messages are written with non-Latin characters, such as Chinese and Russian.

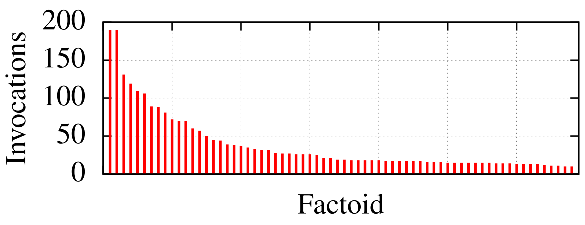

Figure 3: Distribution of the sixty-eight factoid invocations in our corpus.

5 Approach

We are using k-NN and SVM algorithms for classifying messages. This builds on results by Dela Rosa and Ellen (2009), who had found these two supervised learning algorithms to work best on chat messages. Implementation of these algorithms were obtained from Scikit-learn (Pedregosa et al., 2011).

For preprocessing, text was normalized by lowering the case for each term.

We examined different representations for encoding the questions. These include bag-of-words, bigrams, and character n-grams. With the character n-grams, we examined n-grams which overlap words and n-grams which are restricted to within word boundaries. We used $tf - idf$ to weigh the features and χ^2 for feature selection.

6 Empirical Study

We have two hypothesis we are testing:

H1: Supervised learning algorithms will outperform a non-learning baseline approach for classifying BAQs.

H2: Using character n-grams for this domain will allow for better precision and recall when compared to more traditional representations of bag-of-words and bigrams.

Our intuition for H2 is that we believe that character n-grams will allow for better representation of misspelled words commonly seen in chat messages when compared to bag-of-words and bigrams.

6.1 Baseline

Our non-learning baseline algorithm maps questions to factoids by checking if the question contains the factoid command as a word token. As a reminder, multiple factoids can map to the same response. If a question contains multiple factoids, then the most frequently invoked factoid is applied (ties are broken by alphabetical ordering). If a

Representation	χ^2 Feature Size	Precision	Recall	$F_{0.5}$ Score
Non-learning baseline				
–	–	0.44	0.24	0.37
SVM				
Character 3-grams, WB	4000	0.67	0.42	0.60
Character 3-grams	3200	0.67	0.38	0.59
Character 4-grams, WB	3600	0.63	0.40	0.57
Bag-of-words	1600	0.62	0.40	0.56
Character 4-grams	4000	0.58	0.35	0.51
Bigrams	1200	0.51	0.20	0.39
$k-$NN				
Character 4-grams, WB	800	0.55	0.34	0.49
Character 3-grams, WB	800	0.55	0.32	0.48
Character 4-grams	1200	0.57	0.30	0.48
Character 3-grams	800	0.54	0.41	0.47
Bag-of-words	400	0.54	0.31	0.47
Bigrams	400	0.44	0.15	0.32

Table 1: Results for the baseline, SVM and k-NN algorithms. WB means the character n-grams were bounded within word boundaries.

question contains no factoids, then it is considered a HAQ.

6.2 Metrics

We used a 10-fold cross evaluation protocol and precision, recall, and the $F_{0.5}$ score as our evaluation metrics. For this work, we consider precision to be more important than recall because a bot that frequently answers questions incorrectly could anger chat users and cause them to ignore the bots. Therefore, $F_{0.5}$ is more appropriate here than the standard F score because it places more emphasis on precision. Additionally, as these are multi-classification problems, we used the macro version of these metrics to average over the different labels.

When calculating precision, recall, and $F_{0.5}$, we omit the HAQ scores when calculating the macro scores for this multi-classification problem. We also omitted any questions that are incorrectly labeled as HAQ for calculating precision and recall scores because a HAQ can be answered by a human expert. Essentially, we do not penalize for erring on the side of caution.

6.3 Results

Table 1 summarizes the results of the application of our baseline and learning algorithms. We applied all variations of the two learning algorithms, testing on all combinations of representations and χ^2 feature size limits. For the feature size limits, we tried values between [400:4000] in steps of 400. The results display the best configuration for each representation.

As shown, the learning algorithms outperformed the baseline for all three metrics, supporting hypothesis H1. This shows that some questions cannot be easily distinguished by simply looking for factoid commands within the questions.

In regards to the second hypothesis, both learning algorithms performed best when using character n-grams, especially when restricted by word boundaries, thus supporting H2. We believe this is due to the character n-grams being able to better handle noisy nature of chat, especially with the misspellings and abbreviations.

We next examine what *type* of questions do these learning algorithms perform well on, especially when compared to our baseline. For this, we focus on the results of applying SVMs. Figure 4 compares the difference of $F_{0.5}$ scores between SVM and the baseline. For most factoids, SVMs performed better or had similar performance to the baseline. The small number of factoids it performed worse on were generally factoids both

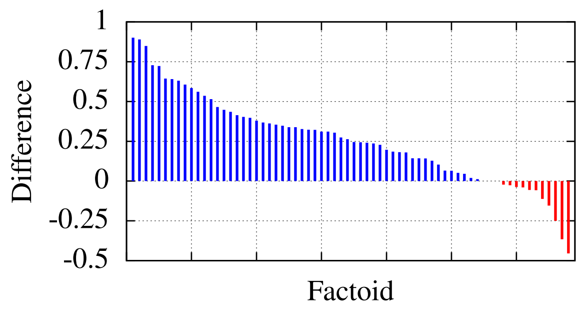

Figure 4: Comparison of $F_{0.5}$ scores between SVM and baseline.

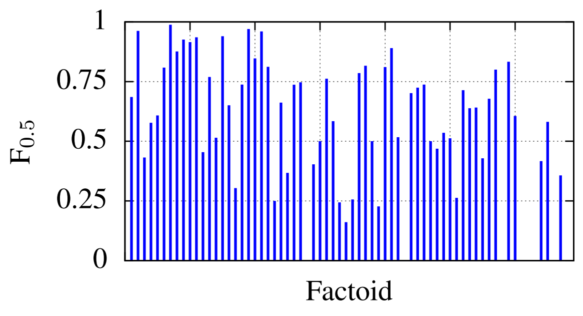

Figure 5: $F_{0.5}$ scores for each factoid when applying SVM, ordered by distribution.

SVM and the baseline performed poorly, finding low $F_{0.5}$ scores.

Figure 5 shows the $F_{0.5}$ scores achieved by SVM for each individual factoid category. These are ordered by their distribution in the corpus (see Figure 3). One fact shown by this figure is that there is not a strong correlation between the distribution size and the result achieved by the SVM. While having more examples within a category does help, there are plenty of factoids where SVM performed poorly. This shows that it is the difficulty of the questions themselves, and not the amount of examples, which causes difficulty for the SVMs, let alone learning algorithms for this domain.

One set of questions SVMs performs well on are questions where users are subsequently directed to another channel. This includes Ubuntu's non-English channels and channels that provide support for other Linux distributions. SVMs did well on all the non-English factoids, with $F_{0.5}$ scores ranging between 0.88 (for Chinese) and 0.99 (for Russian). This is probably due to these questions having uncommon features, such as non-English words or software that is not supposed to be discussed in `#ubuntu`.

One similar pair of questions, which are addressed by two factoids, caused some confusion for the learners – asking for permission to ask a question (e.g., "Can I ask a question?") or asking if anyone can help without stating their problem (e.g., "Can anyone help me?"). This commonly happens with first time visitors to the channel, as they do not know the channel guidelines and will then ask for permission to ask a question or if someone could help them. The channel operators try to encourage users to just ask their question – this happens frequently enough that there are two factoids (labels *ask* and *anyone* in the corpus) to answer these similar questions. Unfortunately, there is a lack of consistency in invoking these two factoids, and as such the learning algorithms we tested had difficulty with these questions.

Some other types of questions that SVMs struggle with are those that cover a wide-range of possible questions. For example, *#ubuntu* can be used either in cases to explain what the channel topic is (for those asking), or to get users on topic (with the possible off-topic message types being wide-ranging); *details*, which can be used whenever someone asks a question or for help without providing enough details for anyone to begin to help; and *wine* to help users with problems running any type of Windows program under Linux. These types of questions may require a human to aid in answering, as it would be difficult to learn all possible types of questions that are covered by these factoids.

7 Conclusions

We have investigated applying supervised learning algorithms to classify questions as HAQ or BAQ, and our results show that these algorithms can outperform a non-learning baseline approach. We also show that character n-grams are a better representation than traditional bag-of-words for our task. More importantly, the learning algorithms can answer some types of questions well, indicating that a self-invoking bot can be created that can answer common questions with confidence.

Future work to extend this is to apply unsupervised methods for finding additional questions to match with the factoids. This would greatly extend what we have presented, as we were restricted to the manually-labeled messages to match questions with answers. We plan on applying a case-based reasoning framework (Richter and Weber, 2013) to achieve such a goal.

A final area to investigate is an extension of `ubottu` that can learn to update its knowledge. Currently, only a few users are allowed to edit or add new factoids to `ubottu`. It would be advantageous if it could add new commands and factoids itself by summarizing common answers, or update outdated factoids should it see a common pattern of answers conflicting with its knowledge.

Acknowledgments

Thanks to NRL for funding this research. David Uthus performed this work while an NRC postdoctoral fellow located at the Naval Research Laboratory. The views and opinions contained in this paper are those of the authors and should not be interpreted as representing the official views or policies, either expressed or implied, of NRL or the DoD.

References

Christopher D. Berube, Janet M. Hitzeman, Roderick J. Holland, Robert L. Anapol, and Stephen R. Moore. 2007. Supporting chat exploitation in DoD enterprises. In *Proceedings of the International Command and Control Research and Technology Symposium*. CCRP.

Emily R. Budlong, Sharon M. Walter, and Ozgur Yilmazel. 2009. Recognizing connotative meaning in military chat communications. In *Proceedings of Evolutionary and Bio-Inspired Computation: Theory and Applications III*. SPIE.

Robin D. Burke, Kristian J. Hammond, Vladimir Kulyukin, Steven L. Lytinen, Noriko Tomuro, and Scott Schoenberg. 1997. Question answering from frequently asked question files: Experiences with the `FAQ FINDER` system. *AI Magazine*, 18(2).

Kevin Dela Rosa and Jeffrey Ellen. 2009. Text classification methodologies applied to micro-text in military chat. In *Proceedings of the International Conference on Machine Learning and Applications*, pages 710–714. IEEE Computer Society.

Liangjie Hong and Brian D. Davison. 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*, pages 171–178. ACM.

Charles Lee Isbell, Michael Kearns, Satinder Singh, Christian R. Shelton, Peter Stone, and Dave Kormann. 2006. Cobot in LambdaMOO: An adaptive social statistics agent. *Autonomous Agents and Multi-Agent Systems*, 13(3):327–354.

Jihie Kim, Erin Shaw, Grace Chern, and Donghui Feng. 2007. An intelligent discussion-bot for guiding student interactions in threaded discussions. In *Proceedings of the AAAI Spring Symposium on Interaction Challenges for Intelligent Assistants*. AAAI.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Michael M. Richter and Rosina O. Weber. 2013. *Case-Based Reasoning: A Textbook*. Springer Berlin.

Samira Shaikh, Tomek Strzalkowski, Aaron Broadwell, Jennifer Stromer-Galley, Sarah Taylor, and Nick Webb. 2010. MPC: A multi-party chat corpus for modeling social phenomena in discourse. In *Proceedings of the Seventh conference on International Language Resources and Evaluation*, pages 2007–2013. European Language Resources Association.

David C. Uthus and David W. Aha. 2013a. Multiparticipant chat analysis: A survey. *Artificial Intelligence*, 199-200:106–121.

David C. Uthus and David W. Aha. 2013b. The Ubuntu Chat Corpus for multiparticipant chat analysis. In *Proceedings of the AAAI Spring Symposium on Analyzing Microtext*, pages 99–102. AAAI.

Tom Chao Zhou, Michael R. Lyu, and Irwin King. 2012. A classification-based approach to question routing in community question answering. In *Proceedings of the 21st International Conference Companion on World Wide Web*, pages 783–790. ACM.