6.863 Project Report: Newsgroup Analysis

Ethan Perry
drayside@mit.edu
ethanLP@media.mit.edu

Robert Seater
rseater@mit.edu

May 15, 2003

Contents

1 Introduction ........................................... 5
  1.1 Objectives ........................................... 5

2 Methodology Overview ................................ 6
  2.1 Message categorization ............................... 7
    2.1.1 Requests ...................................... 7
    2.1.2 Non-requests .................................. 8
  2.2 Statistical Bag-of-Words Approach .................. 8
  2.3 Patterns ............................................. 9
  2.4 Learning Algorithms ................................. 9
  2.5 Training Data ....................................... 10

3 Results .................................................. 11

4 Assessment .............................................. 14
  4.1 Analyzing the Bow results ............................ 14
  4.2 Analyzing the Performance of Our Rule-based System 15
    4.2.1 Effects of Different Grouping .................. 15
    4.2.2 Systematic Errors ............................... 15

5 Future Work ............................................ 19
  5.1 ambiguity in message classification ................ 19
5.2 Further optimization of our patterns ........................................... 20
5.3 technical improvements ........................................................ 21
5.4 improve and diversify how we test the approaches ..................... 22

A Kinds of Messages 24
   A.1 Requests ................................................................. 24
   A.2 other - Non-requests ............................................... 25
   A.3 False positives ....................................................... 25

B Sample Output 26

C Our Rules 28

List of Figures

   1 Performance of our rule-based system ................................. 11
   2 Performance of BOW (vs our system, bottom right; cf Figure 1) ... 13
   3 Grammar for specifying rules ........................................ 28

List of Tables

   1 Confusion matrices for stage1alt (entire dataset) .................. 26
   2 Confusion matrices for stage1alt (training portion of dataset) ... 26
   3 Confusion matrices for stage1alt (testing portion of dataset) .... 26
   5 Details of individual rule performance for stage1alt (entire dataset) 27

Source code listings

   1 rob.rules .............................................................. 29
   2 anthypophora.awk ................................................... 30
   3 adv.awk ................................................................. 31
   4 ans.awk ................................................................. 31
   5 req:arbi.awk .......................................................... 31
   6 adv:buy.awk ........................................................... 32
<table>
<thead>
<tr>
<th></th>
<th>File Name</th>
<th>Lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>cancant.awk</td>
<td>32</td>
</tr>
<tr>
<td>8</td>
<td>req:clar.awk</td>
<td>32</td>
</tr>
<tr>
<td>9</td>
<td>colon-q.awk</td>
<td>33</td>
</tr>
<tr>
<td>10</td>
<td>confused.awk</td>
<td>33</td>
</tr>
<tr>
<td>11</td>
<td>dumb.awk</td>
<td>34</td>
</tr>
<tr>
<td>12</td>
<td>faq.awk</td>
<td>35</td>
</tr>
<tr>
<td>13</td>
<td>geeviz.awk</td>
<td>36</td>
</tr>
<tr>
<td>14</td>
<td>just.awk</td>
<td>36</td>
</tr>
<tr>
<td>15</td>
<td>lets.awk</td>
<td>36</td>
</tr>
<tr>
<td>16</td>
<td>maybe.awk</td>
<td>37</td>
</tr>
<tr>
<td>17</td>
<td>mus.awk</td>
<td>37</td>
</tr>
<tr>
<td>18</td>
<td>need-help.awk</td>
<td>38</td>
</tr>
<tr>
<td>19</td>
<td>req:opin.awk</td>
<td>38</td>
</tr>
<tr>
<td>20</td>
<td>outright.awk</td>
<td>39</td>
</tr>
<tr>
<td>21</td>
<td>please.awk</td>
<td>39</td>
</tr>
<tr>
<td>22</td>
<td>qa.awk</td>
<td>39</td>
</tr>
<tr>
<td>23</td>
<td>false-pos.awk</td>
<td>40</td>
</tr>
<tr>
<td>24</td>
<td>general-req.awk</td>
<td>40</td>
</tr>
<tr>
<td>25</td>
<td>rhe.awk</td>
<td>41</td>
</tr>
<tr>
<td>26</td>
<td>seeking.awk</td>
<td>42</td>
</tr>
<tr>
<td>27</td>
<td>req:meta.awk</td>
<td>42</td>
</tr>
<tr>
<td>28</td>
<td>adv:sell.awk</td>
<td>43</td>
</tr>
<tr>
<td>29</td>
<td>series.awk</td>
<td>43</td>
</tr>
<tr>
<td>30</td>
<td>skeptical.awk</td>
<td>44</td>
</tr>
<tr>
<td>31</td>
<td>someone.awk</td>
<td>44</td>
</tr>
<tr>
<td>32</td>
<td>req:spec.awk</td>
<td>44</td>
</tr>
<tr>
<td>33</td>
<td>req:tech.awk</td>
<td>45</td>
</tr>
<tr>
<td>34</td>
<td>thank.awk</td>
<td>46</td>
</tr>
<tr>
<td>35</td>
<td>wonder.awk</td>
<td>46</td>
</tr>
<tr>
<td>36</td>
<td>I.awk</td>
<td>47</td>
</tr>
<tr>
<td>37</td>
<td>it.awk</td>
<td>47</td>
</tr>
<tr>
<td>38</td>
<td>you.awk</td>
<td>47</td>
</tr>
</tbody>
</table>
### Dramatis Personæ

<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ethan</td>
<td>visionary leader</td>
</tr>
<tr>
<td>Rob</td>
<td>unemployed British housewife</td>
</tr>
<tr>
<td>Derek</td>
<td>grease monkey</td>
</tr>
</tbody>
</table>
1 Introduction

Internet discussion groups such as Usenet newsgroups contain a wide variety of messages. While some groups enable the exchange of information, others are more social in nature, and some are just a dumping ground for spam. Using a traditional newsreader application makes it difficult to get a sense of character of each group without reading a series of messages one by one. Natural language processing and related techniques offer the potential to enable quick summaries of the behavior within discussion groups. One way to put such techniques to use is to identify messages containing requests and classify those requests.

Identifying and classifying requests can aid both existing and potential members of a newsgroup. Existing members can locate questions that need answering, and quickly find out whether their previous postings need clarification. Potential members can get a sense of the flavor of the group by the frequency and distribution of requests: A newsgroup with many open ended requests for opinions will appeal to different users than one which contains mostly requests for details about specific topics. Analysis of requests can augment and complement other dimensions of classification such as subject matter (from the newsgroup name), identifying flames (Spertus, 1997), and determining degree of subjectivity (Wiebe et al., 2002). Such information helps extract a representation of the identity of participants, which is useful from a sociological perspective (Chewar et al., 2003; Erickson et al., 2002; Smith, 1999).

1.1 Objectives

We present a rule-based method, using regular expression patterns, to identify messages containing requests, and to classify those requests into meaningful categories.

We use the term “request” to refer to any expression of need for a particular kind of information or service.

Rule of Thumb:

- Not all requests are questions.
- Not all questions are requests.

“Request” is an interesting type of message from a natural language perspective because they are not trivially detectably by, say, scanning for question marks. That naive approach results in both in many false positives (such are rhetorical questions which are not actually requests), and many false negatives (such as statements of need, which are requests but not questions). We compare our rule-based approach to existing bag-of-word statistical approaches, and discuss areas for potential improvement.
2 Methodology Overview

Our system relies on a corpus of Usenet messages from the 20 newsgroup data set compiled at Carnegie Mellon in conjunction with the Bow package (McCallum, 1996). We hand-classified 1211 messages for the presence of requests and according to the type of that request. Those results were used as training data for a learning algorithm. We have used an approach similar to the Smokey project, which used a combination of hand-crafted patterns and machine learning to identify which newsgroup messages were flames (Spertus, 1997). Our patterns use regular expressions to identify patterns of language which are likely to identify request types and rule out false positives. For each message, we generate a bit vector recording which patterns matched that message. These bit vectors are fed as attributes to the C4.5 machine learning application, which then generates a decision tree for classifying additional messages (Quinlan, 1996).

The key components of our system are:

- **Patterns**: A set of awk files containing regular expression rules. See appendix C.

- **Corpus**: A set of hand-categorized newsgroups messages; a subset of the BOW corpus McCallum (1996).

- **Message Pre-processor**: This program strips out extraneous text and puts every sentence on a separate line. Extraneous information includes: headers (except for the subject line), footers and signatures, and quoted text from an earlier posting. While this is not intellectually interesting, it requires a degree of effort to do well, and is quite important to the overall performance of the system.

- **Directory creator**: This component reads the tags file and builds a new directory tree, with messages placed into subdirectories according to which category they belong in. Consider a message \( M \) which human tagging classified as categories \( c_1 \) through \( c_n \) (a message may be in more than one category, although most fall into only one). Copies of \( M \) are placed in the subdirectories named \( c_1 \) through \( c_n \). This tree structure is used by the bow package.

- **Data file creator**: This program fires the patterns on each message and creates a data file recording which patterns matched that message. There is a row for every message containing a binary string indicating which patterns matched the message.

- **C4.5**: (Machine learning application) We feed our data file and message names to C4.5, which generates a decision tree for classifying additional messages.
• **High Precision Rule Optimizer:** This program attempts to improve the decision tree by looking for high precision patterns which are not being taken full advantage of by the decision tree.

• **Report Generators:** These programs take the results of the classification and generate the graphs in Figures 1 and 2 and the tables in appendix B, as well as a few others that we have not included in this report.

The implementation of the above comprises approximately 900 lines of shell script and 1100 lines of awk script, not counting the rules (which are listed in appendix C).

### 2.1 Message categorization

There are two stages to the task of identifying different kinds of requests:

1. identifying whether or not a message is a request
2. deciding what kind of request a message is, given that it is a request

We used the following schema for classifying messages:

#### 2.1.1 Requests

All message tags beginning with `req` are requests of some form.

- **req:opin** A request for an opinion, or an open ended question inviting discussion. The author expects a reply, possibly many replies.

- **req:tech** A request for a technique or general approach to a problem. This includes requests for operational guidelines for devices and technology.

- **req:spec** A request for details (specifications, symptoms, stats, etc) of a specific product, technology, or disease. It does not cover requests for how to use, find, or deal with a specific product; those fall under “req:tech”

- **req:meta** A request concerning the operation of the newsgroup itself, email addresses of the members, questions about whether or not a post is off-topic, and surveying the newsgroup to find out if anyone is a specialist on a particular topic.

- **req:fact** A request for a fact or an answer to a factual question. These are historical or general facts, not facts about a specific item (req:spec) or a technique (req:tech).

- **req:expr** A request for experience or empirical information others may have about a topic.
req:call A call for participation (in a conference or event) or submission (of papers or answers to a survey).

req:clar A request for clarification about a particular posting by another member. This includes only serious questions, and not rhetorical questions phrased as a need for clarification.

req:arbi A request for arbitration by someone else about two conflicting points of view — “which one of us is right”.

req:adv-sell An announcement of willingness to sell — an advertisement.

req:adv-buy An announcement of willingness to buy — a “wanted” ad.

2.1.2 Non-requests

rhe Rhetorical questions — sentences which are questions in form but intended as statements.

faq These are any message which has a list of questions and their answers, intended to answer those questions before they are asked. Examples of this are FAQ’s (frequently asked questions) and QA’s (question-answer interview summaries)

ans A question intended as an answer to a request or as an instruction. For example, “Why don’t you try doing such-and-such?”.

mus A musing the author makes to him or herself, or a question the author asked of himself and then immediately answers. For example, “I tried that five times (or was it six?)” or “Is this justice? Of course not!”.

- All other non-requests. We refer to this category as ‘other’.

Examples of each category can be found in the Appendix A.

2.2 Statistical Bag-of-Words Approach

We initially tried using a statistical bag of words tool as a baseline to see how a purely statistical approach to classify messages would perform. Other semantic classification tasks have used statistical methods based on the presence of specific adjectives and verbs as well as co-located sequences of words. An example is Janyce Wiebe’s work with her colleagues to indicate the presence of subjective language in a message (Wiebe et al., 2002). The statistical tool we used, the Bow package from Carnegie Mellon University (McCallum, 1996), uses keywords to determine how similar documents are (McCallum and Nigam, 1998). Not surprisingly, the presence of keywords was an ineffective predictor of whether a message was a request or not as requests
have a variety of formats and subject matter that do not consistently use the same types of words. However, using a significantly larger training corpus might improve the performance of this statistical approach.

2.3 Patterns

Our patterns took the form of regular expressions written in awk, and are listed in appendix C. They fell into two main categories:

1. **General counts**: counted a word or symbol which is related to questions or requests, but does not offhand indicate a particular kind of request. Examples are dollars signs, question marks, and various WH words. These patterns usually returned a count, rather than a simple boolean value.

2. **Category targeted**: for each category, one or more patterns were written with more complex structure, which attempted to be high precision. Oddly, often a rule attempting to accurately identify one category ended up being used by the decision tree to identify a very different one. For example, anthypophora (posing a question and then immediately answering it) is a pattern written to identify rhetorical questions, but the decision tree used it to identify non-requests (the ‘other’ category). It is not clear if this is merely unexpected or a mistake by the decision tree.

2.4 Learning Algorithms

**c4.5 decision tree**

The c4.5 application takes a data file (described in the previous section) and a names file that lists the classes, attributes and attributes values. It then generates a decision tree based on our data file and names file.

The decision tree generated by c4.5 tended to ignore patterns which have high precision but low recall. This observation means that a set of high precision patterns might be useful if combined (using disjunction) into a single pattern, but are ignored when separate – an undesirable result. We attempted to correct for this as follows: for each pattern P and each category C and a decision tree DT, if P has higher precision for C than DT does for C, then whenever P matches a message M, we put M in C even if DT would not have done so.

This produced a big win in some toy examples we tried, but produced a small, but negative, effect on our larger results. We beleive that this technique has potential for improving results in the general case, but failed to do so in our case because of how we wrote our patterns. Our experiments on small examples and our intuitions both suggest that high-precision correction is effective when there are several patterns with
very high precision (.9-1.0) and moderate (.25-.33) recall. Our patterns did not have high enough recall, and often did not have high enough precision. We speculate that there would be an improvement had we used a pattern set which

1. has much more specific (and thus precise) patterns, and
2. was much larger (thus having a high combined recall).

2.5 Training Data

We tried training on the messages grouped in different ways, representing different tasks one might wish to perform

- **sorted0** Contains all messages, which are divided into all sub-categories (including different types of requests and different types of false positives).

- **sorted0alt** Contains all the messages, but collapses together categories that are semantically close, and which people have difficulty distinguishing:
  - other: other, faq, req-call
  - rhetorical: rhetorical, req-clar, mus, ans
  - fact: req-fact, req-spec, req-tech, req-meta, req-expr
  - adv: req-adv-sell, req-adv-buy
  - opin: req-opin, req-arbi

- **stage1** Contains all messages, which are divided into “request” and “non-request”. False positives (e.g., rhetoricals), are considered “non-requests”.

- **stage1alt** Same as stage1, except false positives are removed.

- **stage2** Contains only those messages manually designated as requests, which are divided into the different sub-categories of requests.

- **stage2alt** Like stage2, but with req:spec, req:tech, and req:fact all merged into req:fact.

- **stages-1-and-2** First and second stages combined. Contains all messages, divided into sub-categories of request and “non-request” (which includes false positives, such as rhetoricals).

We tried training on half the data and testing on the other half, as well as training on 80% and testing on the remaining 20%. These are identified as ‘split2’ and ‘split5’ on Figure 1. The values on Figure 1 are arithmetic averages of ten runs (where each run involves a new random partitioning of the dataset).
3 Results

Figure 1 illustrates the results of our experiments with our system. Precision is on the y-axis and Kappa is on the x-axis. While these two values do not represent a trade-off — usually high kappa entails high precision — these are the values that we are interested in optimizing. The best points are near the top right, and the worst points are towards the bottom left. The same kind of graph is used to show BOW’s performance in Figure 2.

Figure 1 contains data points for each dataset, such as sorted0 or stage1alt, evaluated with both a 50/50 and an 80/20 training split, and for both straight c4.5 and c4.5 with our high-precision rule filter. So, since there are seven datasets, that means the graph has $7 \times 2 \times 2 = 28$ datapoints. As mentioned previously, each datapoint is an average of ten experiments. However, since the results for the 50/50 and 80/20 splits are about the same, and our high-precision rule filter does not seem to effect the overall result much, there are seven clearly identifiable clusters of four points each on the graph: one cluster for each dataset.

Appendix B shows some example confusion matrices and other reports for one of our experiment runs. More such details are available on request.

**Figure 1** Performance of our rule-based system

![Performance of Rule-based System](image)
Figure 2 contains seven graphs from our experiments with BOW: one for each dataset. The eighth graph in the bottom right corner is a miniature version Figure 1, which shows the results for our system.

Each BOW graph in Figure 2 contains 16 datapoints, which are always clustered fairly closely together. We ran BOW with all four of its classification schemas: Naive Bayes, K-Nearest-Neighbor, TFIDF, and probabilistic indexing. We also ran it with and without so-called ‘stop’ words such as ‘the’ and ‘of’. Each run produces two datapoints, so \( 4 \times 2 \times 2 = 16 \).

Each of the seven BOW graphs correspond to one of the seven clusters on our graph. For example, the best cluster on our graph is for the dataset stage1alt, which should be compared with the fourth BOW graph (ie, the one for stage1alt).
Figure 2 Performance of BOW (vs our system, bottom right; cf Figure 1)
4 Assessment

Our system was intended to classify newsgroup messages as requests, potentially classifying the messages into subcategories. Initially we wanted to compare a purely statistical approach to a rule-based approach. A key factor was the pre-processing we used to strip out irrelevant content from the messages. When we initially ran the Bow package, our best precision rate was 55%. After stripping out the occurrence of references to previous messages,

Bow is able to achieve an average precision rate of 78% for distinguishing between requests and non-requests. However, while it classifies an average of 95% of non-requests correctly, it achieves a much lower rate of 31% precision for requests. Our rule based system, in contrast, has a precision rate of 80% for non-requests, and 60% for requests. By using a decision tree with rules targeted at identifying requests, we were able to greatly increase our capability to do so, with a cost of a higher error rate in identifying non-requests due to false positives. This may be more useful, however, for users interested in finding requests (rather than non-requests). We believe with additional refinement of the rules we would be able to improve our overall precision, since we had limited time to experiment with different permutations of rule sets.

When we classified messages into all subcategories of requests and non-requests, Bow achieved an average precision rate of 49%, while our rule-based approach achieved a slightly better rate of 58%. Bow succeeded at classifying messages into the largest categories of other and rhetorical, but had poor performance on all request categories, with req-adv-sell and req-call faring slightly better than the other request categories. Our rule-based system had more success with req-adv-buy, req-adv-sell, req-spec and also with the faq category. Our targeted rules for these categories were more likely to identify messages accurately than solely looking for common keywords.

4.1 Analyzing the Bow results

Bow first generates a model in which it identifies the frequency of word occurrence, and then looks for the distribution of those words in messages of each category. For most of the categories, the presence of these words does not correspond to reliable evidence that of membership in the category. Therefore, most of the Bow output indicates successful classification of frequently occurring categories, but a kappa value in the range of 0.5. The categories that are the exceptions to this rule are

- req-adv-sell This category has a high frequency of the word “sale” so Bow successfully identifies a portion of the messages.

- req-call This category has a high frequency of the word “call” so Bow successfully identifies a portion of these messages. There were also some duplicates messages in our data set advertising a particular Libertarian conference with
words like “hotel” and “Marriott” in these messages. Bow has a higher success rate classifying these messages, because the identical messages enabled it to more reliably correlate word frequency with category.

4.2 Analyzing the Performance of Our Rule-based System

Since we used hand-crafted regular expressions to identify particular types of requests, our success was partly driven by our ability to manually discern patterns of language that indicated a particular request type.

4.2.1 Effects of Different Grouping

A set of categories with an even distribution of messages tends to produce a more accurate decision tree, since the learning mechanism looks for mappings in which large numbers of messages are identified successfully, and statistically this is more likely for the larger categories. Also, categories with a small number of messages may be less likely to have a sufficient number of examples to generate the appropriate rule profile. Groupings of categories with one or two categories with many messages and several additional categories with a small number of messages are likely to perform poorly. Therefore, the groupings we selected which have a more even distribution of messages, such as stage 1 — all requests vs. all other messages will be more likely to generate a successful decision tree.

4.2.2 Systematic Errors

In the confusion matrix resulting from the test in which we classified messages across all subcategories, the following errors are notable:

“Advertisement of Item for Sale” misclassified as “Other” Many of these advertisements that are misdiagnosed consist of a product name, description and a price, with no additional information other than an email address. This leaves few cues or patterns to discover to identify that it is an advertisement without having semantic understanding that the item discussed is a product that could be offered for sale. The price (searching for dollar signs) is not sufficient, because discussions of money or budgets appear in political newsgroups, prices for products sometimes appear in FAQ type listings of books or resources, and sports groups often mention players’ salaries. Therefore the learning algorithm does not conclude that the presence of a dollar sign indicates a request, and no other patterns are sufficient. Improving the performance on these messages might require a corpus of product-related terms accompanied by the price indicator, which was out of scope for this research. It also may help to incorporate information about the name of the newsgroup; in our case,
most advertisements were in “misc.forsale”, and almost all dollar signs indicated an advertisement.

“Request for Specific Product or Technology” misclassified as “Request for Fact”  Both requests tend to be short messages often with the word “thanks”. Additional distinctions beyond these two types of messages are often semantic, such as an understanding that the question “Can anyone help me?” refers to a prior explanation in the message about a particular problem that requires a technique, as is the case in comp.os.ms-windows.misc.10051. In comp.os.ms-windows.misc.10007 “Does anyone have the documentation for the MS Mouse Driver 8.2?” implies a need for a technique that would be explained in the documentation. Because of the subtlety that separates these categories, our human taggers often disagreed about which tag was appropriate, perhaps decreasing the effectiveness of the training corpus and indicating the difficulties in classifying messages between these two categories.

“Request for Clarification” misclassified as “Rhetorical”  Since rhetorical questions often rely (often very subtle) on semantic knowledge that is not discernible from simple syntactic patterns (such as regular expressions), our system is not as successful at classifying these types of messages. Less than half of the rhetorical questions encountered were deemed “clear independent of context” by the human taggers. Almost all of the rest were in the form of requests for clarification or requests for facts (see below), but were clearly rhetorical given context and content.

In rec.sport.hockey-52645, we have an example where semantic understanding of the question is needed, but context is not necessary for a human to be confident. The form of the question is not enough.

These new rule changes are great! However, I think that your rules are MUCH too complicated. How will the normal average fan be able to count how many fouls a player has? And then we would even have to remember the names of the players, in order to determine who drew the foul! And, of

In soc.religion.christian-20507, we have an example where context is required to determine if the question is rhetorical (as it is) or an honest question.

The canon of Scripture is complete.
Does this mean that God no longer speaks?
I have heard his voice -- not audibly (though some have), but clearly nonetheless.

One might judge that the question is clearly phrased to sound absurd, but consider the following (fabricated) context, and renamed propper noun:
You referred to Bob as dumb.
Does this mean that Bob no longer speaks?

distinguishing these cases would require quite subtle understanding of the meaning of the surrounding context.
Later in soc.religion.christian-20507, there is a prime example of context information which regular expressions can identify (and ours do).

Is what I heard equivalent to Scripture? No.

A question followed by a one-word statement is a sure sign of a rhetorical statement or musing.

**Confusion between “Other” and “Rhetorical”** The decision tree generated from our rules using very intricate branching structures based on the best match it could find with our training corpus. Because many messages fall into these two categories, they are more likely to be confused than other categories. There are also several different pre-processing issues that may result in questions remaining in the message when it is irrelevant to the discussion at hand. These problems will bias the decision tree to classify messages based on somewhat “noisy”, less accurate training data.

- **Subject headers** A specific area of confusion could be that our decision to include the Subject line of the header in the messages may be introducing misleading content. The subject line provides useful information, including the “Re:” that indicates a message is a response. Our intention is to develop rules that test subject line content for specific types of request and non-request categories but this has not been implemented yet. The presence of the “Re:” header lines followed by the initial poster’s subject line sometimes retains a question after the message has been pre-processed, as in message alt.atheism-51163-other.msg, which has subject line “Re: ¡Political Atheists?” but has no other questions, but has been classified as rhetorical, perhaps based on the presence of a question mark.

- **Removal of signature files** While we detect several different indicators of signatures such as the use of a blank line followed by a line with only “—”, there are many variations on the conventions people use to indicate signature files. In sci.crypt-15179, the author did not have a blank line before the “—” line, and his signature contained the question, “Do you know where your wallet is?”, leading this message to be erroneously classified as a rhetorical. It is difficult to use pattern matching to identify when a signature starts if a non-conventional indicator is being used, or if the indicator might potentially be used in some other context, such as “—” as a hyphen.
• Commented lines While we remove lines that begin with certain characters that commonly indicate commented reproductions of earlier messages (e.g. “>”), some newsreaders indicate these lines by just indenting the prior message, and our pre-processor is not sophisticated enough to be able to tell whether a line that begins with this kind of indent is a commented prior message or not. This may result in certain questions from prior messages being included the analysis of a subsequent message, misleading the decision tree.

Confusion between “Request for Fact” and “Rhetorical” Many rhetoricals are phrases as requests for facts.

In alt.atheism-51149, we have a message fragment which look both like a request for a fact (or even an opinion) and a rhetorical statement.

>As for rape, surely there the burden of guilt is solely on the rapist?

Not so. If you are thrown into a cage with a tiger and get mauled, do you blame the tiger?

Without the ability to recognize the question as absurd, there is little hope of knowing it to be rhetorical. Absurdity is a rather subtle semantic property, and not at all syntactic.

Poor results for “Request referring to the newsgroup itself” Detecting “meta” requests requires at least some semantic understanding; you have to be able to differentiate based on whether the subject matter is the newsgroup, a member’s email address, or something else. A pattern which simply searches for the the keywords “newsgroup” and “email” produced too many false positives to be effective.
5 Future Work

While our results are promising, there are many directions in which we can readily improve. At almost all stages of the process, there are opportunities to improve by being more automatic in our selection of rules, and better handling ambiguity that is legitimately present.

5.1 ambiguity in message classification

In reality, requests do not always fall cleanly and clearly into one category or another. Even when there is only one request in the entire message, which is almost always the case, that request may play several roles at once: it may both be requesting experience and requesting specific details, or both requesting a fact and clarification.

We experimented with completely merging those categories which were hard to differentiate among by hand, and found that it produced a vast improvement (reported in Section 3). While those results are not scientifically grounded, they indicated that one of the major obstacles is the need for a better treatment of ambiguity.

messages falling into more than one category A fair number of messages were classified into two or more categories - in such cases, a copy of the message is placed into each of the corresponding directories. Since our approaches only put a message into one category, it would always register as making a mistake on all but one of the messages. In order to not bias the analysis, our accuracy measure ignored such messages, but that is only a temporary solution. There are two more satisfactory solutions to this problem, depending on how you interpret what multiple tags mean:

- Investigate learning methods which determine the best $n$ categories for each message. Under this interpretation, the analysis makes one mistake per category it fails to put the message into; we take the tags to mean “this message should be in all of these categories.”. This will increase errors, but may not hurt kappa, since human taggers will probably also show much higher discrepancy when they can choose more than one category.

- Count a message as being correctly categorized if it fall into any of the indicated categories. Under this interpretation, the tags are taken to mean “I would be happy if this message were classified into any of the following categories.”.

Either method will surely improve the accuracy of our approach.
more rigorous manual tagging, human error  While we calculated kappa based on the confusion matrix, comparing the results to our initial tags as if they were a perfect gold standard. It would be preferable to have a kappa for the comparison of multiple human taggers, who will inevitably show discrepancy in their tags. Our experience was that errors in our tags were not a rare occurrence – there were both cases which were obviously mistakes once they were noticed, and cases which were difficult judgement calls (where the human taggers disagreed even after re-examining the message).

category boundaries  Another approach to resolving legitimate ambiguity between categories is to simply note which categories are similar. Rather than have all categories be equally different, build a network of categories:

- the nodes in the network are categories
- the network is completely connected
- the weight on the edge between two categories is the number of times two humans disagreed as to which of those categories a message should be in

When analysing accuracy, errors are punished an amount inversely proportional to the weight of the corresponding edge.

For Example: Suppose that there are 4 messages for which Mr.A chose category $C_1$ but Mr.B chose $C_2$, and 2 times when Mr.A chose $C_2$ but Mr.B whose $C_1$. The edge between nodes $C_1$ and $C_2$ in the category network will get a weight of 6. Suppose, also, that the weight between $C_1$ and $C_2$ is 2 (calculated in a similar manner). When an analysis confuses $C_1$ and $C_3$, its accuracy would be be punished three times as much as if it had confused $C_1$ and $C_2$.

5.2 Further optimization of our patterns

Each pattern can be either binary or continuous: binary patterns either match or don’t, continuos patterns return the number of matches. This choice, along with the nature of the pattern itself, was chosen manually based on the authors’ experience and by examining a subset of the test data. The way the patterns were chosen tended to produce high precision, low recall rules, but such rules are not used very well by c4.5’s decision tree. It is possible that a better set of rules would provide dramatic improvement. A central problem with human generated rules, is that small mistakes in regular expressions can cause a large number of errors; it is easy to inadvertently allow combinations of words without realizing it. Hopefully automation of rule generation would help avoid such inadvertent mistakes. Automatically generating rules via machine learning, perhaps with some limited guidance, would provide a much clearer picture of how effective the rule-based approach is independent of one’s ability to
hand write patterns. Alternatively, humans might provide a set of seed rules which would be altered and selected via a genetic algorithm.

5.3 technical improvements

There are two main opportunities for technical improvement, each of which would serve to improve performance by removing noise from the corpus.

**preprocessor** Preprocessing message to remove extraneous text turned out to be a highly non-trivial task. There is opportunity for additional rules for eliminating footers—many messages contained large quotes or signatures which may cause patterns to misfire. For example, many footers contained rhetorical questions, which when not stripped not only lead to misclassification of that message, but may also disrupt the construction of the decision tree (causing even more damage). Attempts to automatically recognize footers were To the human eye, it was always clear what was a footer and what was part of the main message, suggesting that the right set of rules could successfully automate the process (thus making it feasible). A similar problem arose in several cases where quoted text was indicated by leading whitespace on a line, something our preprocessor was not equipped to handle.

In some cases, there were duplicate messages with a newsgroup, or the same message was posted to several newsgroups in our data set. The worst case of this was an announcement of a Libertarian conference which was posted 3 times in each of 3 different newsgroups, making “Libertarian” and “Mariott” two of the most common words! We removed most duplicates by hand, but the process should be automated to reduce human error and oversights.

**More Sophisticated NLP techniques** Performing a hierarchical parse of the sentences would allow us to remove nested phrases. Our experience was that when a pattern matches a nested *spar* expression, it was almost always a false positive. However, such cases were fairly rare, so it is not likely to make a significant improvement.

**normalization** Another improvement would be to normalize word choice. That is, replace each word with a canonical representative from an equivalence class of its synonyms. Such equivalence classes are available on Wordnet, or we might need to construct a thesaurus specialized for requests. For example, some differences might be:

- WH words should be in separate equivalence separate, although in other contexts (like flames) they could be lumped together
• Most nouns could be compressed into a single class, although certain nouns such as “sale”, “advice”, and “opinion” would need to be separated out from that class.

Normalization would require part of speech tagging. Our preliminary experience with P.O.S. tagging newsgroup messages was that it was adequate, but tended to have many errors. This is surely due to the poor grammar, lack of punctuation, and slang common to newsgroup emails.

Normalization probably won’t help rule-based approach as much as bag-of-words approaches, since many synonyms were hard coded into the patterns. However, there are still four major benefits

(a) it is likely to provide some benefit to both techniques.

(b) it would be more rigorous than ad-hoc guessing of likely synonyms,

(c) we will be able to test the merit of the core of each technique, rather than its brittleness.

5.4 improve and diversify how we test the approaches

Currently we select a random subset of our data to be training data, and use the rest as testing data. An alternative would be to choose training data from one set of newsgroups (chosen randomly) and the testing data from another set. This experiment would determine how well the training carried over between different newsgroups. That is, how domain specific is the decision tree that is built?

A larger corpus of human classified messages would not only improve the effectiveness of the training, but would also give us a more statistically significant number of some of the categories. For example, currently there are only 2 examples of requests for arbitration (req:arbi), only one of which is a strong example.
References


A  Kinds of Messages

The following section provides example excerpts of each type of message that are requests and non-requests.

A.1 Requests

req:opin - Request for an opinion  “What would Montreal give San Jose if the Sharks got first pick and took Daigle?”

req:tech - Request for a technique or approach  “How do you take off the driver side door panel from the inside on an ’87 Honda Prelude?”

req:spec - Request for a specific thing (product, technology, information)  “Do you know about a mouse odometer for windows?”

req:fact - Request for a (historical) fact  “How many years were involved in the mechanization of cotton farming? When did this first appear?”

req:expr - Request for experience of empirical information  “I’m thinking of buying a new Dodge Intrepid - Has anyone had any experiences that they’d like to share?”

req:adv-sell - Advertisement of item for sale  “Panasonic KX-T3000H, Combo black cordless & speaker phone all in one. ...If you are interested in either of the above mail me at radley@gibbs.out.unc.edu.”

req:adv-buy - Advertisement of intent to buy item  “I’m looking for a used/inexpensive audio mixer.”

req:call - Call for participation, call for papers  “The HumBio Project: Call for Data and Visualizations ...If you’re interested in sharing your data for the purpose of education and supporting the introductions of CD-ROM and scientific visualization technologies into our schools, please e-mail phughes@igc.org.”

req:clar - Request for clarification from author  “Where did you read this? I don’t think this is true.”
req:arbi - Request for arbitration between posters  "Which one of us is right?"

req:meta - Request referring to the newsgroup itself  "I would be grateful if someone could repost the rules and instructions for the playoff pool sometime next week, for the benefit of those who missed the first two postings."

A.2  other - Non-requests

Messages that are not requests nor false positives have been labeled as non-requests: "I’m surprised nobody mentioned that twitching of the eyelid can be a symptom of an infection, especially if it also itches or stings." “Maine beat LSSU 5-4.” “THANKS TO ALL OF YOU WHO RESPONDED TO MY POSTING. THE PROBLEM WITH MY TRUCK’S HEADLIGHTS LOW BEAM PROBLEM WAS A ‘LOOSE WIRE CONNECTION’. IT WAS NOT THE ‘FUSE’ AS A MINORITY OF YOU SUGGESTED.” “The space and astronomy discussion groups actually are composed of several mechanisms with (mostly) transparent connections between them.”

A.3  False positives

These messages either have questions as part of an FAQ document, or are some sort of rhetorical question.

faq - FAQs, QA  "WHY DOES THE SHUTTLE ROLL JUST AFTER LIFTOFF? The following answer and translation are provided by Ken Jenkins (kjenks@gothamcity.jsc.nasa.gov)."

"So what is the philosophical justification or basis for atheism?" There are many philosophical justifications for atheism. To find out why a particular person chooses to be an atheist, it’s best to ask her.”

rhe - Rhetorical question  “Sure, an original together with Id card of sender and receiver would be fine. So what’s that supposed to say? Am I missing something?"

“Can you say ‘neutron and other radiation flux due to radioactive decay’, boys and girls?

ans - Rhetorical question as an answer to a request  “Was your system.ini file erased for some reason? That would be the only reason that I would think that Windoes is giving you the error now.”

mus - Rhetorical question as musing comment  “I often wonder just what Castro would have done if the Cubans presently in Miami would have been forced to remain in Cuba. Would they have revolted and killed him off, or been killed?”
## B Sample Output

Our system produces a variety of reports, including confusion matrices and a table of individual rule performance (which we use for improving the rules). Some samples of these reports are given in this section for the stage1alt dataset.

### Table 1 Confusion matrices for stage1alt (entire dataset)

<table>
<thead>
<tr>
<th>Confusion</th>
<th>other</th>
<th>req</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>other</td>
<td>606</td>
<td>20</td>
<td>0.968</td>
</tr>
<tr>
<td>req</td>
<td>43</td>
<td>280</td>
<td>0.867</td>
</tr>
<tr>
<td>Totals</td>
<td>649</td>
<td>300</td>
<td></td>
</tr>
</tbody>
</table>

P(E) = 0.559; P(A) = 0.934;  
Kappa = 0.850

<table>
<thead>
<tr>
<th>Confusion</th>
<th>other</th>
<th>req</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>other</td>
<td>602</td>
<td>24</td>
<td>0.962</td>
</tr>
<tr>
<td>req</td>
<td>40</td>
<td>283</td>
<td>0.876</td>
</tr>
<tr>
<td>Totals</td>
<td>642</td>
<td>307</td>
<td></td>
</tr>
</tbody>
</table>

P(E) = 0.556; P(A) = 0.933;  
Kappa = 0.848

### Table 2 Confusion matrices for stage1alt (training portion of dataset)

<table>
<thead>
<tr>
<th>Confusion</th>
<th>other</th>
<th>req</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>other</td>
<td>469</td>
<td>24</td>
<td>0.951</td>
</tr>
<tr>
<td>req</td>
<td>29</td>
<td>229</td>
<td>0.888</td>
</tr>
<tr>
<td>Totals</td>
<td>498</td>
<td>253</td>
<td></td>
</tr>
</tbody>
</table>

P(E) = 0.551; P(A) = 0.929;  
Kappa = 0.843

<table>
<thead>
<tr>
<th>Confusion</th>
<th>other</th>
<th>req</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>other</td>
<td>468</td>
<td>25</td>
<td>0.949</td>
</tr>
<tr>
<td>req</td>
<td>27</td>
<td>231</td>
<td>0.895</td>
</tr>
<tr>
<td>Totals</td>
<td>495</td>
<td>256</td>
<td></td>
</tr>
</tbody>
</table>

P(E) = 0.550; P(A) = 0.931;  
Kappa = 0.846

### Table 3 Confusion matrices for stage1alt (testing portion of dataset)

<table>
<thead>
<tr>
<th>Confusion</th>
<th>other</th>
<th>req</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>other</td>
<td>114</td>
<td>19</td>
<td>0.857</td>
</tr>
<tr>
<td>req</td>
<td>16</td>
<td>49</td>
<td>0.754</td>
</tr>
<tr>
<td>Totals</td>
<td>130</td>
<td>68</td>
<td></td>
</tr>
</tbody>
</table>

P(E) = 0.554; P(A) = 0.823;  
Kappa = 0.604

<table>
<thead>
<tr>
<th>Confusion</th>
<th>other</th>
<th>req</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>other</td>
<td>114</td>
<td>19</td>
<td>0.857</td>
</tr>
<tr>
<td>req</td>
<td>15</td>
<td>50</td>
<td>0.769</td>
</tr>
<tr>
<td>Totals</td>
<td>129</td>
<td>69</td>
<td></td>
</tr>
</tbody>
</table>

P(E) = 0.552; P(A) = 0.828;  
Kappa = 0.617
Table 5 Details of individual rule performance for stage1alt (entire dataset)

This page shows only the top part of the table. Numbers in parenthesis indicate total number of matches. ‘==’ indicates number of matches; ‘!=' indicates number of non-matches; ‘P’ indicates precision; ‘R’ indicates recall. Precision values greater than 0.5 are visually highlighted with a box.

<table>
<thead>
<tr>
<th>Rule</th>
<th>other (626)</th>
<th>(323) box</th>
</tr>
</thead>
<tbody>
<tr>
<td>I (92)</td>
<td>== 35</td>
<td>57</td>
</tr>
<tr>
<td></td>
<td>!= 591</td>
<td>266</td>
</tr>
<tr>
<td></td>
<td>P 0.380</td>
<td>[0.620]</td>
</tr>
<tr>
<td></td>
<td>R 0.056</td>
<td>0.176</td>
</tr>
<tr>
<td>advertising-words (2)</td>
<td>== 0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>!= 626</td>
<td>321</td>
</tr>
<tr>
<td></td>
<td>P 0.000</td>
<td>[1.000]</td>
</tr>
<tr>
<td></td>
<td>R 0.000</td>
<td>0.006</td>
</tr>
<tr>
<td>ans-formations (10)</td>
<td>== 3</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>!= 623</td>
<td>316</td>
</tr>
<tr>
<td></td>
<td>P 0.300</td>
<td>[0.700]</td>
</tr>
<tr>
<td></td>
<td>R 0.005</td>
<td>0.022</td>
</tr>
<tr>
<td>anthyphthora (187)</td>
<td>== 123</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td>!= 503</td>
<td>259</td>
</tr>
<tr>
<td></td>
<td>P [0.658]</td>
<td>0.342</td>
</tr>
<tr>
<td></td>
<td>R 0.196</td>
<td>0.198</td>
</tr>
<tr>
<td>arbitration (1)</td>
<td>== 0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>!= 626</td>
<td>322</td>
</tr>
<tr>
<td></td>
<td>P 0.000</td>
<td>[1.000]</td>
</tr>
<tr>
<td></td>
<td>R 0.000</td>
<td>0.003</td>
</tr>
<tr>
<td>bible-ratio (70)</td>
<td>== 53</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>!= 573</td>
<td>306</td>
</tr>
<tr>
<td></td>
<td>P [0.757]</td>
<td>0.243</td>
</tr>
<tr>
<td></td>
<td>R 0.085</td>
<td>0.053</td>
</tr>
<tr>
<td>buying-words (68)</td>
<td>== 20</td>
<td>48</td>
</tr>
<tr>
<td></td>
<td>!= 606</td>
<td>275</td>
</tr>
<tr>
<td></td>
<td>P 0.294</td>
<td>[0.706]</td>
</tr>
<tr>
<td></td>
<td>R 0.032</td>
<td>0.149</td>
</tr>
<tr>
<td>cancant (0)</td>
<td>== 0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>!= 626</td>
<td>323</td>
</tr>
<tr>
<td></td>
<td>P 0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>clarification (103)</td>
<td>== 49</td>
<td>54</td>
</tr>
<tr>
<td></td>
<td>!= 577</td>
<td>269</td>
</tr>
<tr>
<td></td>
<td>P 0.476</td>
<td>[0.524]</td>
</tr>
<tr>
<td></td>
<td>R 0.078</td>
<td>0.167</td>
</tr>
</tbody>
</table>
C Our Rules

In our system, rules are specified in a .rules file, which is written according to the grammar in Figure 3. Lines that begin with a hash character (#) are considered to be comments. A rule specification has three parts: the name of the rule, the action to evaluate, and the names of the results.

There are two kinds of rule actions: egrep and awk. An egrep action optionally specifies some egrep parameters (eg, -i to be case insensitive). An awk action specifies the absolute path of the awk script to run (the name of which must end in `.awk`). An awk script used for this purpose must produce only one word of output, and so consequently the output must be produced from the END block.

An action may return two kinds of results: either match/no-match or a count of the number of matches. The latter case is indicated by the c4.5 reserved word ‘continuous’. If the names of the results are not specified, the system defaults to ‘+ruleName’ to indicate a match and ‘-ruleName’ to indicate a non-match.

Figure 3 Grammar for specifying rules

\[
\begin{align*}
\text{line} & \quad ::= \quad \text{ruleName ruleAction resultNames} \\
\text{ruleAction} & \quad ::= \quad [\text{egrepSwitches}] \text{regexp} | \text{awkFileName} \\
\text{resultNames} & \quad ::= \quad \text{‘continuous’} | \text{matchString} \text{nonMatchString}
\end{align*}
\]

Note that rule names cannot include spaces (obviously), and also cannot be the reserved word ‘category’.

The rules file that we use for our experiments, which has come to be known as rob.rules, is shown in Listing 1. The individual rules in awk files that it refers to are listed on the subsequent pages.
Listing 1 rob.rules

# rule files to run
# formats:
# name /path/awkfile.awk

# general rules
anthypophora /usr/local/ngat/rules/anthypophora.awk
advertising-words /usr/local/ngat/rules/adv.awk
ans-formations /usr/local/ngat/rules/ans.awk
arbitration /usr/local/ngat/rules/req:arbi.awk
buying-words /usr/local/ngat/rules/adv:buy.awk
cancant /usr/local/ngat/rules/cancant.awk
clarification /usr/local/ngat/rules/req:clar.awk
colon-question /usr/local/ngat/rules/colon-q.awk
confused /usr/local/ngat/rules/confused.awk
# do dot dot dot
# dumb question
# explain
faq-format /usr/local/ngat/rules/faq.awk
geewiz /usr/local/ngat/rules/geewiz.awk
just /usr/local/ngat/rules/just.awk
lets /usr/local/ngat/rules/lets.awk
maybe /usr/local/ngat/rules/maybe.awk
musings /usr/local/ngat/rules/mus.awk
need-help /usr/local/ngat/rules/need-help.awk
opin /usr/local/ngat/rules/req:opin.awk
say-will-be-question /usr/local/ngat/rules/outright.awk
please /usr/local/ngat/rules/please.awk
qa-format /usr/local/ngat/rules/qa.awk
question-after-colon /usr/local/ngat/rules/false-pos.awk
question-formation /usr/local/ngat/rules/general-req.awk
rhetorical-formations /usr/local/ngat/rules/rhe.awk
seeking /usr/local/ngat/rules/seeking.awk
self-referential /usr/local/ngat/rules/req:meta.awk
selling-words /usr/local/ngat/rules/adv:sell.awk
sequence-of-questions /usr/local/ngat/rules/series.awk
skeptical /usr/local/ngat/rules/skeptical.awk
some-any-one-body /usr/local/ngat/rules/someone.awk
some-any-one-body-q /usr/local/ngat/rules/someone.awk
specific-product /usr/local/ngat/rules/req:spec.awk
technique /usr/local/ngat/rules/req:tech.awk
thank /usr/local/ngat/rules/thank.awk
wondering /usr/local/ngat/rules/wonder.awk

# pronouns followed by a questionmark
I /usr/local/ngat/rules/I.awk
it /usr/local/ngat/rules/it.awk
you /usr/local/ngat/rules/you.awk

# domain specific / augmenting preprocessing
# libertarian /usr/local/ngat/rules/libertarian.awk
bible-ratio /usr/local/ngat/rules/bible.awk

# simple counts
dollar-sign /usr/local/ngat/rules/dollars.awk  continuous
question-mark /usr/local/ngat/rules/qmark.awk  continuous
Listing 2 anthypophora.awk
#anthypophora
# a self posed question, but not one of musing nor an FAQ

BEGIN{
    count=0;
    IGNORECASE=1
    stagesA=0;
}

# pattern A
# question followed by a key word, short sentence, or exclamation
#  
#  What’s wrong with that?
#  Nothing at all.
#  
#  Is that crazy?
#  Perhaps.
#  
#  What are we doing?
#  Being losers!
#  
#  FIRST
# a question mark
/
\/? { 
    stagesA=1;
}

# THEN
# keyword
/"(nothing|yes|no|perhaps|maybe|absolutely|every ?thing|some ?one|some ?body|any ?one|any ?body|no ?
   if (stagesA==1) {count++;}
}

# OR THEN
# one or two or three word statement
/"[a-zA-Z]+( [a-zA-Z]+)?( [a-zZ-Z]+)? (?\.|\!|:\);)? / { 
    if (stagesA==1) {count++;}
}

# OR THEN
# exclamation
/!/ { 
    if (stagesA==1) {count++;}
}

# FIXUP
/\/? { 
    stagesA=0;
}

END{print count;}


Listing 3 adv.awk

```
# sale-or-trade
# some kind of adv (buy or sell)

BEGIN{count=0;IGNORECASE=1;}

# listing two transactional options
# sale or trade
# buy and sell
# trade / sell
# trade/sale
/\(sale|sell|trade|buy)\( /\ | or | and \)\(sell|sale|trade|buy)\)/ {count++}

END{print count;}
```

Listing 4 ans.awk

```
# answering-with-a-question
# identify ans and maybe rhe and perhaps even mus

BEGIN{count=0;IGNORECASE=1;}

# counterexample
# How about the south american tree monkey?
# What about me?
/\(how|what\) about / {count++;}

"have you" formations
# have you tried rebooting?
/\(Have you \)/ {count++}

END{print count;}
```

Listing 5 req:arbi.awk

```
# req:arbi
# arbitration

BEGIN{count=0;IGNORECASE=1;}

# am I right? Who's right?
/\wh.*[^~,\.] right ?\/? {count++;}

#/right ?\/? {count++;}

END{print count;}
```
Listing 6 adv:buy.awk

```bash
#adv:buy

BEGIN{count=0;IGNORECASE=1;}

# key words and phrases indicating a wish to buy a certain product
# I'm looking for a
# I am trying to find a
# I want to locate a
# I am interested in buying
/(^[a-zA-Z]+$)|(am|I'm) looking for|(?want|trying) to (buy|get|locate|find|trade for|locate)|interested in buying

# request for a seller to respond
# does anyone have a...
# I am looking for anybody who has a...
/(^[a-zA-Z]+$)|(any|some)?(one|body)) (have|who has) .*/ {count++;

END{print count;}
```

Listing 7 cancant.awk

```bash
#cancant

BEGIN{count=0;IGNORECASE=1;}

# pronoun positive ... negative pronoun
# I can FOO, can't I?
# They wouldn't do that to us, would they?
/(^[a-zA-Z]+$)|(he|she|it|they|I|you|we|there|one|any)?(one|any)?(body|some)?(one|some)?(body|no)?(one|no)?

END{print count;}
```

Listing 8 req:clar.awk

```bash
#clarification

BEGIN{count=0;IGNORECASE=1;}

/(^[a-zA-Z]+$)you (seem|appear)?.*\?/ {count++;

/(^[a-zA-Z]+$)(explain|explanation|elaborate|elaboration|clarify|clarification) /{count++;

/(why|what|which\?) (do|would) you (think|mean|want)/{count++;

END{print count;}
```
Listing 9 colon-q.awk

```awk
#colon-question

BEGIN{
    count=0;
    IGNORECASE=0;  # NOT A MISTAKE
    stages=0;
}

#colon then question
# here is a good question:
# Is there a god?
#
# a good analogy is the following:
# which came first, the chicken or the egg?
/:/{
    stages=1;
}
#THEN
/\?/{
    if (stages==1) {count++;}
}
#FIXUP
!/:/ {
    stages=0;
}

END{print count;}
```

Listing 10 confused.awk

```awk
#confused

BEGIN{count=0; IGNORECASE=1;}

#confused, baffled
/confused|baffled/ {count++;}

END{print count;}
```
Listing 11 dumb.awk

#dumb

BEGIN{count=0;IGNORECASE=1;}

#self depreciating statement
#note: the insult word can be any word at all
#this might be a really dumb question?
#this might be an extremely crazy thing to say?
#this might be a smart thing to say.

/([^a-zA-Z]|^)((might be|is this) an? (really( ?,)? |very( ?,)? |extremely( ?,)? |rather( ?,)? |part

END{print count;}

Listing 12 \texttt{faq.awk}
\begin{verbatim}
#faq

BEGIN{
    count=0;
    IGNORECASE=1;
    stagesA=0;
    stagesC1=0;
    stagesC2=0;
}

# all caps question
#/^[a-z]* \?$/ {count++;}

# titles that give it all away
# FAQ
# frequently asked question
/([Ff][Aa][Qq])/ {count++;}

# pattern A
# question followed by a "well, ..."
/\?/ {
    stagesA=1;
}
# THEN
/"Well \?\?/ {
    if (stagesA==1) {count++;}
}
# FIXUP
!/\?/ {
    stagesA=0;
}

# pattern C
# blank then question then blank
/^$/ {
    stagesC1=1;
}
# THEN
/\?/ {
    if (stagesC1==1) {stagesC2=1;}
}
# THEN
/^[0-9]* \?\. / {count++;}

END{print count;}
\end{verbatim}
Listing 13 geewiz.awk

```
# geewiz

BEGIN{count=0;IGNORECASE=1;}

# geewiz exclamations followed by a questionmark
# gosh, is he that good?
/(gosh|god|man|wow|gees|gee|huh|well then) .*/?/ {count++;}
/(gosh|god|man|wow|gees|gee|huh|well then) ?!/ {count++;}

END{print count;}
```

Listing 14 just.awk

```
# just
# often indicates erotema

BEGIN{count=0;IGNORECASE=1;}

# just as the first or second word of a sentence
# just why are you so stupid.
# you just want my attention
/^just / {count++;}
/^[a-zA-Z]+ just / {count++;}

END{print count;}
```

Listing 15 lets.awk

```
# lets

BEGIN{count=0;IGNORECASE=1;}

# lets, let's, let 's
/^Let ?'s / {count++;}

END{print count;}
```
Listing 16 maybe.awk

```awk
#maybe

BEGIN{count=0;IGNORECASE=1;}

#maybe at the start of a sentence
# maybe you should...
/"(maybe|perhaps)/ {count++}

END{print count;}
```

Listing 17 mns.awk

```awk
#musing

BEGIN{count=0;IGNORECASE=1;}

#’or’ beginning a sentence
# or was is a mastodon?
/([^a-zA-Z]|^)or( ,)? (was it|perhaps) .*/ {count++;}

#question in parnes
# and then I killed 5 (6?) dinosaurs
/\(.\.*\? \?\)/ {count++;}

END{print count;}
```
Listing 18 need-help.awk

```awk
#need-help

BEGIN{count=0;IGNORECASE=1;}

#help me
/0help me0/ {count++;}

#something unexpected
# I am having something very unusual happening
/([^a-zA-Z]+)(something|this ([a-zA-Z ])?thing) (unusual|unexpected|wierd|weird|bizarre|strange)0.*

#my question is...
/([^a-zA-Z]+)((my|your) (questions?|problems?))/ {count++;}
/\^I (had|have)\ am\ having)\ (this|the|a) ([a-zA-Z ])?problem)/ {count++;}

#desperate
/(would\ be|are|is|will\ be)\ ((much( ,)? )*(very( ,)? )*(extremely( ,)? )*(really( ,)? )*)(appreciated|appreciate|need)0.*
/([^a-zA-Z]+)any\ info\ (rmation)/ {count++;}

END{print count;}
```

Listing 19 req:opin.awk

```awk
#opinions

BEGIN{count=0;IGNORECASE=1;}

/^\w.* do you think/ {count++;}
/^do.* have an opinion/ {count++;}
/\comment on / {count++;}
/ give [a-zA-Z ]+ \(opinion|thoughts|ideas\)/ {count++;}

END{print count;}
```
Listing 20 outright.awk

```awk
BEGIN{count=0;IGNORECASE=1;}

# uses the actual word "question" or "request"
# I have a question...
# I have a strange request...
/
```

Listing 21 please.awk

```awk
BEGIN{count=0;}

# please
/
```

Listing 22 qa.awk

```awk
BEGIN{count=0;IGNORECASE=0;}  # intentional

/
```
**Listing 23** false-pos.awk

```awk
# non-question-questionmark
# question after a colon

BEGIN{
    count=0;
    IGNORECASE=1;
    stages=0;
}

# question in form but not in content -- not really a questions
# that's just a way of saying this: am I god?
# a good analogy is: am I god?
# that's like saying: am I god?
/([^a-zA-Z])*(way of saying|analogy|metaphor|example|like saying).*/ {  
    stages=1;
}
# THEN
/
    if(stages==1) {count++;}
}

# FIXUP
!/(^[a-zA-Z]* (way of saying|analogy|metaphor|example).*) / {  
    stages=0;
}

END{print count;}
```

**Listing 24** general-req.awk

```awk
# general-formation
# general ways to start a question

BEGIN{count=0;IGNORECASE=1;}

/
    (are(n '?')|do(n '?')|have(n '?')|will|won '?t|can(n '?')|would(n '?')|did(n '?')|should(n '?')|was(n '?')|www(n '?')|will?|would?|should?|www?|www?/ {count++;}

/
    (am|do(n '?')|have(n '?')|will|won '?t|can(n '?')|would(n '?')|did(n '?')|should(n '?')|was(n '?')|www(n '?')|www?|www?|www?|www?|www?|www?|www?/ {count++;}

/
    ((do)? you (mean|think|saying|trying|want|thought))([^a-zA-Z]*/ {count++;}

/
    ([^a-zA-Z]*|I.* (really(,)? )*(trying|want|interested|would like to|appreciate|wondering|looking|interested))([^a-zA-Z]*/ {count++;}

/
    (who|what|where|when|how|whom|whose|which|whence|wherefore|why)(ever)? / {count++;}

END{print count;}
```
Listing 25 rhe.awk

```
#rhetorical

BEGIN{
    count=0;
    IGNORECASE=1;
    stagesB=0;
}

do you still want to waste my time?
/mind (me\|my\|if I) ask(ing)?.*$/ {count++;}

can you say FUBAR boys and girls?
/\"can you say .*/ {count ++;}

who would have thought/guessed
/\[^a-zA-Z]*\"who would have / {count++;}

closing parenthetical expressions
/, anyway ?\?/ {count++;}
/, [a-zA-Z]+ see ?\?/ {count++;}

I think ?\. / {count++;}

\"are you\' \ formations
/\[^a-zA-Z]*\"are you (attempting\|trying\|actually\|really) / {count++;}
/\[^a-zA-Z]*\"are you (as|so|such) .* that / {count++;}
/\[^a-zA-Z]*\"or are you [a-zA-Z]+ / {count++;}
/\[^a-zA-Z]*\"(what are you thinking)/ {count++;}

should I? (two words)
/\"(should|can|could|am|are|is|were|was|have|has|would|will|wo|did|does)(n \?'t)? (he|she|it|they|I|you)/ {count++;}

\[^a-zA-Z]*\"what you are saying[^?]*$/ {count++;}
/\"(now(\ ,)? isn\t that[^a-zA-Z]* / {count++;}

\[^a-zA-Z]*\"what is\s wrong with / {count++;}

\[^a-zA-Z]*\"since when
/\[^a-zA-Z]*\"since when (should|can|could|am|are|is|were|was|have|has|would|will|wo|did|does)(n \?'t)? (he|she|it|they|I|you)/ {count++;}

\[^a-zA-Z]*\"I assume.*$/ {count++;}

\[^a-zA-Z]*\"isn\t that ADJ/ADV ?!?!?
/\"Isn \t that [a-zA-Z]+ ?(\?!+)/ {count++;}

\[^a-zA-Z]*\"annoyed
\[^a-zA-Z]*\"not this again
/\[^a-zA-Z]*\"not (going through |going to have to go through )?this ([a-zA-Z]+ )?again/ {count++;}
/\"(are we)|(am I)) really going to / {count++;}

\[^a-zA-Z]*\"oh no! this again?
/\"(oh no)|(uh oh)[^a-zA-Z]/ {count++;}
```

Listing 26 seeking.awk

```awk
#seeking-survey-of-population

BEGIN{count=0;IGNORECASE=1;}

/([^a-zA-Z]|~)any( ?one)? .* (out there|reading this|in this (news)?group|who can answer)/ {count++;}

/is any ?one a / {count++;}

END{print count;}
```

Listing 27 req:meta.awk

```awk
#mention-self

BEGIN{count=0;IGNORECASE=1;}

#"meta"-questions
"other possible words to look for: email, post

/([^a-zA-Z]|~)(newsgroup|thread) / {count++;}

END{print count;}
```
Listing 28 adv: sell.awk

```
#adv: sell

BEGIN {count=0; IGNORECASE=1;}

#key words and phrase formats indicated a sale
# wife for sale, any price
# I am willing to sell my wife.
# I have all these things trade.
# I want to get rid of my wife.
# my brother is trying to trade his wife, too.
/(for|will|willing to|will|things to|stuff to|want to|must|trying to) (sale|sell|trade|get rid of)/ {count++;

#more specific phrases that didn’t fit into the above format
# $20 or best offer
# everything must go!
# trade or sale
# sale/trade
/(best offer|priced to sell|asking for|brand new|((sale|sell) (?or|\|\-|\&\&|and) ?trade) |(trade (?or|\|\-|\&\&|and) ?sell)))/ {count++;

#statement of availability
# I am offering 5 golden rights.
# I’m selling 3 turtle doves.
/((am|’m) (offering|selling))/ {count++;

END {print count;}
```

Listing 29 series.awk

```
#series-of-questions

BEGIN{
   count=0;
   IGNORECASE=1;
   stages=0;
}

#series of questions
/
\?
/ {  
   if(stages==1) {count++;
   stages=1;
   
   #FIXUP
   !/\?
   / {  
   stages=0;
   }

END{print count;}
```
Listing 30  skeptical.awk

```
#skeptical
#questions which might indicate skepticism

BEGIN{count=0;IGNORECASE=1;}

#question made as an afterthought
# then what
# -- whatever
/(\*|\d:\d|--|\d\d\d\d\d\d/) (*)(who|whom|what|where|when|which|how|whence|whence|whose|whos)(ever)/

#conjunction then if word
# and, who goes there anyway?
/\(^\s*(And|But|Or),\s*(.*)(who|whom|what|where|when|which|how|whence|whence|whose|whos)(ever)\s*/{count++}

#helper verb beginning sentence
# won't there be a free lunch there?
# shouldn't we be having fun yet?
/\(^\s*(should|can|could|am|are|is|were|was|have|has|would|will|wo|did|does)\s*(n't)\s*/{count++}

END{print count;}
```

Listing 31  someone.awk

```
#someone

BEGIN{count=0;IGNORECASE=1;}

#("a-zA-Z"]"][^-](some|any|no)?(one|body))/ {count++}
/([^a-zA-Z]|^[^-](some|any)?(one|body))/ {count++}

END{print count;}
```

Listing 32  reqspec.awk

```
#specific (product) information

BEGIN{count=0;}

#req:spec
/([0-9]\.[0-9]([a-zA-Z][a-zA-Z]+[0-9]+[a-zA-Z])/ {count++}
/([^a-zA-Z]version([^a-zA-Z]drivers([^a-zA-Z]product info).*/ {count++}
/([^a-zA-Z]([have been|am] using|trying|working with|running)).*/ {count++}

END{print count;}
```
Listing 33 req:tech.awk

`req:tech

BEGIN{count=0;IGNORECASE=1;}

# what method should I use?
/\what.*\([^()]*\)\([^()]*\)? I do\|way to).*/\{count++;}

# how do I go about catching my own tail?
/\how.*\([^()]*\)\([^()]*\)? \{count++;}
/\how \(do I|should I|ought I|does one|can I\)/ \{count++;}
/\can anyone tell me how to/ \{count++;}

END{print count;}`
Listing 34 thank.awk

#thank

BEGIN{count=0;IGNORECASE=1;}

/"thank/ {count++;}

/([a-zA-Z]?)grateful/ {count++;}

END{print count;}

---

Listing 35 wonder.awk

#just-wondering

BEGIN{count=0;IGNORECASE=1;}

#/([a-zA-Z]?)just (asking|curious|wondering)(([a-zA-Z]?)$)/ {count++;}

/ (asking|curious|wondering)/ {count++;}

END{print count;}

---
Listing 36 l.awk

```awk
#you

BEGIN{count=0;IGNORECASE=1;}
#
../you../?
/you.*\?/ {count++}

END(print count;)
```

Listing 37 it.awk

```awk
#it

BEGIN{count=0;IGNORECASE=1;}

/it.*\?/ {count++}

END(print count;)
```

Listing 38 you.awk

```awk
#you

BEGIN{count=0;IGNORECASE=1;}

/you.*\?/ {count++}

END(print count;)
```

Listing 39 bible.awk

```awk
#bible
#bible citations and mathematical ratios

BEGIN{count=0;IGNORECASE=1;}

#number colon number
# 12 : 30
# 3:4
/
/[0-9]\.* *[0-9]/ {count++;}

END(print count;)
```
Listing 40 dollars.awk
# dollar-sign

BEGIN{count=0;IGNORECASE=1;}

# dollar sign
/\$/ {count++;}

END{print count;}

Listing 41 qmark.awk
# question marks

BEGIN{count=0;IGNORECASE=1;}

/\?/ {count++;}

END{print count;}

Listing 42 linecount.awk
# linecount

BEGIN{count=0;IGNORECASE=1;}

// {count++;}

END{print count;}

Listing 43 ifthenwhy.awk
# ifthenwhy

BEGIN{count=0;IGNORECASE=1;}

# if I'm god then why am I short?
/^If .* then (who|what|when|where|how|which|whom|whence|wherefore|why|whose)(ever)? / {count++;}

END{print count;}

Listing 44 short.awk
# short
BEGIN{count=0;IGNORECASE=1;}
# short, open ended questions
/([a-zA-Z][`])(\((any| ?one| ?body)? )?\(thoughts?|suggestion?s|ideas?|interest(ed)?|takers?|advice|comes\))/ {count++;}

END{print count;}
Listing 45 shouldI.awk

```bash
#shouldI

BEGIN{count=0;IGNORECASE=1}

#should I?
/
(should|can|could|am|are|is|were|was|have|has|would|will|wo|did|does)(n \?'t)? (he|she|it|they|I|you|we)? \n
END{print count;}
```

Listing 46 whydont.awk

```bash
#whydont

BEGIN{count=0;IGNORECASE=1;}

#why don't you try eating your own head
/
(Why don \?'t you )/ {count++;}

END{print count;}
```

Listing 47 how.awk

```bash
#how

BEGIN{count=0;IGNORECASE=1;}

/
(how \/) / {count++;}

END{print count;}
```

Listing 48 what.awk

```bash
#what

BEGIN{count=0;IGNORECASE=1;}

/
(what((so)?ever)? / {count++;}

END{print count;}
```
Listing 49 when.awk

#when

BEGIN{count=0;IGNORECASE=1;}

/\when(( so)? ever)? / {count++;}

END{print count;}

Listing 50 where.awk

```awk
#where
BEGIN{count=0;IGNORECASE=1;}
/
where((so)?ever)? / {count++;}
END{print count;}
```

Listing 51 which.awk

```awk
#which
BEGIN{count=0;IGNORECASE=1;}
/
which((so)?ever)? / {count++;}
END{print count;}
```

Listing 52 who.awk

```awk
#who
BEGIN{count=0;IGNORECASE=1;}
/
who((so)?ever)? / {count++;}
END{print count;}
```
Listing 53 whose.awk

# whose

BEGIN{count=0;IGNORECASE=1;}

/\s*whose\((so)?\s*ever)?\s*/ {count++;}

END{print count;}

Listing 54 why.awk

# why

BEGIN{count=0;IGNORECASE=1;}

/\s*why\((so)?\s*ever)?\s*/ {count++;}

END{print count;}