CS 229 Final Paper

Sorting Forum Responses by Relevance to Original Post

Aaron Abajian

Friday, December 13th, 2013

1 Abstract

Online forums provide a medium for discussing niche topics. Forums consist of threads, where each thread consists of an original post followed by response posts. A subset of original posts are questions related to the forum's topic. The responses to these posts are often expert answers due to the niche nature of the forum. However, not all responses will be answers. Users are free to write anything in their response post. In this project, we train a Naïve Bayes classifier and a linear support vector machine (SVM) classifier to distinguish thread *answers* from *non-answers*. Our features are preprocessed whitespace-delimited terms occurring in each response.

We obtained training and testing data by implementing a web crawler and a web parser targeted against the online discussion forum *Slickdeals.net/forums* (1). We crawled $T = 8$ threads whose original posts were questions. There was an average of 94.25 responses for each post, giving a total of $N = 754$ responses for classification. We manually assigned a labeling of 0 or 1 to each response depending upon whether that response attempted to answer the original question.

We performed leave-one-out cross-validation on the eight test questions. The Naïve Bayes classifier correctly classified 70.1% (110/157) of the *answers* and 74% of the *non-answers* (442/597). The linear-kernel SVM correctly classified 80.2% of the *answers* (126/157) and 82.5% (493/597) of the *non-answers*.

2 Background

A growing number of websites feature a Q&A style format where questions and answers are crowdsourced. StackOverflow and MathOverflow cater to a specific technical audience while sites such as Quora and Yahoo! Answers allow questions from any discipline. These sites share similar features such as commenting and feedback systems, the ability to up-vote correct answers, and user rankings based upon response quality.

A common problem for Q&A sites is the generation of content. Users must create an account, post their question, and wait for another user to respond. This dependency on *original* content generation restricts the growth rate of Q&A sites. We identify an alternative to original content generation by considering online forums.

Forums are structured environments that facilitate discussion around shared interests such as technology, sports, pet ownership, cooking techniques, etc. There are numerous forums for nearly every niche topic. The topic of a forum describes its user base, and the collective wisdom of a forum's community provides a rich set of *expert* knowledge about that topic. These attributes suggest that a forum may provide a compelling dataset for a Q&A site.

Consider the structure of a forum. Forums consists of threads, where each thread contains an original post followed by response posts. A subset of original posts will be questions related to that forum's topic. For example, a forum for motorcycling enthusiasts may contain questions about motorcycle engines and parts. However, unlike a Q&A site, responses to forum questions are not required to answer that question. Forum users are free to respond to a question in any manner they see fit. *The purpose of this project is to classify each response to a forum question based upon whether or not that response answers the question.* The set of questions and their answers may then be used to populate the database of a Q&A site.

3 Methods

In order to obtain a large dataset, we initially built a web crawler and a web parser targeted against the online discussion forum *slickdeals.net/forums*. Pseudocode is given below.

```
for each thread in forum:
    originalPost = getFirstPost(thread)
    if(originalPost).isQuestion()
        responses = getRemainingPosts(thread)
        answers = filter(responses)
```

The majority of this pseudocode does not require machine learning; the crawler grabs the HTML for each thread, a deterministic algorithm parses the posts from the HTML, and an English natural language processing program determines if the first post is a question. The machine learning comes in when we attempt to filter the responses for those that answer the original question. Consider the following question from the forum *Slickdeals.net/forums*:

> *Q*: What is the best Black Friday deal on a 70" LED TV?

This user is looking for a Black Friday sale on a big-screen TV. He has asked a frugally-minded community for help. Let's look at some of the responses:

R_1: I'm also looking for a good deal.
R_2: I am in the same market.
R_3: \$998 Vizio at Sam's Club is the best deal.
R_4: I haven't seen any deals yet, guys.
R_5: Walmart has LED Vizio 120HZ for \$998.
R_6: Sharp for \$1799 at Microcenter.

The first, second and fourth users have shared that they are also interested in such a deal. Their responses may provide the original poster with a sense of camaraderie, but they do not answer his question. The third, fifth and sixth responses provide direct answers to the original question.

We crawled $T = 8$ threads whose original posts were questions about product sales, similar to the question above. There was an average of 94.25 responses for each thread, giving a total of $N = 754$ responses for classification. We manually assigned a labeling of 1 or 0 to each response depending upon whether or not that response answered the original question. There were more non-answers than answers, with a ratio of 597 to 157.

We used our manually-assigned labelings to train a Naïve Bayes classifier and a linear-kernel SVM classifier as follows.

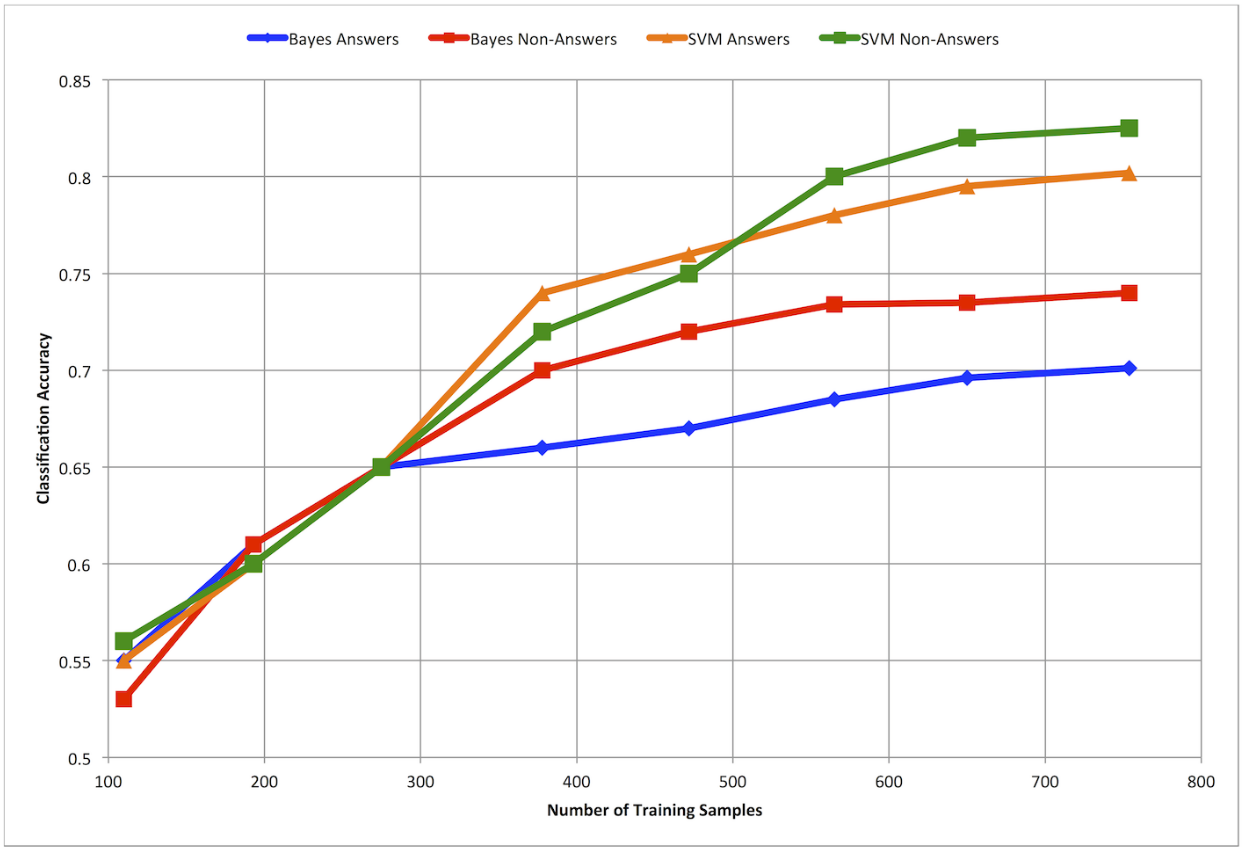

Figure 1: Classification accuracy versus number of training samples for the Naïve Bayes and SVM classifiers. Separated out for *answers* and *non-answers*.

Let $R_1, \ldots, R_N$ be the list of responses to all questions (i.e. $|R| = 754$). Each R_i is tokenized into a set of white-space delimited strings, and each string undergoes a series of preprocessing steps:

1. Strip all non-ASCII characters
2. Map numeric prices (e.g. \$100, \$0.50, \$499, etc.) to a single common term, PRICE_TERM
3. Stem words (e.g. "prices" $\rightarrow$ "price", "reviewing" $\rightarrow$ "review", "lowered" $\rightarrow$ "lower", etc.)
4. Add the concatenation of adjacent terms to the terms list (e.g. "no deal" forms three terms, "no", "deal", and "nodeal")

The frequencies of each term in the *non-answer* set of responses and the *answer* set of responses are used to train the classifiers. For a new response, R_{N+1}, consisting of preprocessed strings $S = s_1, \ldots, s_M$, the Naïve Bayes prediction is given by:

$$\begin{aligned} p(R_{N+1} = 1 \mid S) \\ &= \frac{p\left(S \mid R_{N+1} = 1\right) \cdot p\left(R_{N+1} = 1\right)}{p\left(S\right)} \\ &= \frac{1}{1 + \frac{\prod_{i=1}^{M} p(s_i|R_{N+1}=0)p(R_{N+1}=0)}{\prod_{i=1}^{M} p(s_i|R_{N+1}=1)p(R_{N+1}=1)}} \end{aligned}$$

The implementation is similar to that given in the Lecture Notes (2). Laplace smoothing was applied to account for terms that may appear in R_{N+1}, but not in the training set. We implemented the Naïve Bayes classifier in MATLAB following the formula above. The linear SVM classifier was implemented in C/C++ by Fan, et. al. (3). The frequency count vectors for each term along with their manual classification were provided as features to the linear SVM.

4 Results

We initially removed common words from the feature set, but found that the classifiers' accuracies improved when they were left in (4.8% improvement). Concatenating adjacent words also led to a 7.5% improvement, as described in the methods section. This may be accounted for by the observation that many English words are negated by the word immediately before them (e.g. "no deal"). We performed leave-one-out cross validation on the 8 questions. The overall accuracy demonstrated with the 754 responses was 74% (558/754 correctly classified responses) for the Naïve Bayes classifier, and 82% (619/754 correctly classified responses) for the linear SVM. The Naïve Bayes classifier correctly classified 70.1% (110/157) of the *answers* and 74% of the *non-answers* (442/597). The linear-kernel SVM correctly classified 80.2% of the *answers* (126/157) and 82.5% (493/597) of the *non-answers*. The difference between the *answer* and *non-answer* classification accuracies was likely due to the large number of *non-answer* responses (597) relative to *answers* (157).

Although the classifier accuracies were not exceptional, the utility of the algorithm was clearly visible by sorting the responses by their classification probabilities. For the example given in the Methods section, the algorithm produced the following sorting:

R_3: $998 Vizio at Sam's Club is the best deal.
R_6: Sharp for $1799 at Microcenter.
R_5: Walmart has LED Vizio 120HZ for $998.
R_1: I'm also looking for a good deal.
R_4: I haven't seen any deals yet, guys.
R_2: I am in the same market.

We found that particular terms were more indicative of an *answer* classification over a *non-answer* classification. These terms included "deal", "PRICE_TERM", and (interestingly) "at". The latter of these may have contributed to the 4.8% boost that was found when common words were left in the algorithm.

5 Conclusion

Online forums provide a valuable resource for asking questions about niche topics. Our project demonstrates a preliminary method for organizing responses to forum questions based upon whether the response answers the original question. Since we restricted our consideration to questions pertaining to product deals, a natural extension of this project is to apply the algorithm to arbitrary questions. This will require taking into consideration the content of the question itself; a problem in the domain of natural language processing. We speculate that this will require a separate clustering algorithm in order to group questions by related semantics. Responses to clustered questions (such as all questions beginning with "When" or "Where") could be used to train a classifier as in this paper.

6 References

1. *Slickdeals.net* is a free, user-driven deal sharing site with a mission to provide consumers an avenue to collaborate and share information in order to make the best shopping decisions.

2. CS 229 Lecture Notes

3. R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A Library for Large Linear Classification, Journal of Machine Learning Research 9(2008), 1871-1874. *http://www.csie.ntu.edu.tw/~cjlin/liblinear*