Classifying responses on online discussion forums

Aaron Abajian

aabajian@stanford.edu

Abstract

There are online discussions forums covering an endless variety of topics. Each forum has many topic-specific questions with expert answers. However, the best answer is often buried somewhere in the discussion thread. Given a question on an online forum, we focus on the problem of classifying each response to the question as an *answer* or a *non-answer*. Where the *answer* label indicates if the response made a reasonable attempt at answering the question (even if the answer is incorrect).

We train a recursive neural network (RNN) using 59,205 question-response pairs from ten online forums with true labels obtained through Amazon Mechanical Turk. Our model reads each question-response pair from left-to-right similar to a next-word prediction RNN. The model minimizes the number of incorrectly-classified question-response pairs. After hyperparameter tuning, the model achieved a 76.3% classification accuracy. We qualitatively validate the model by showing common *non-answer* responses such as "thanks", "bump", and "nevermind" result in strong neuronal activation.

Our work builds off of previous work done by Aaron Abajian as part of CS 229 and CS 221. In those classes, a logistic regression (LR) classifier and a support-vector machine (SVM) classifier were trained using hand-selected features. The SVM and LR classifiers outperformed the RNN due to hand-crafted features. We suggest ways for improving the RNN to match their performance.

1 Introduction

Online discussion forums have existed since the early days of the Internet. A forum consists of threads with one or more posts. An original poster (OP) creates a thread by posing a question pertinent to the forum's topic. The OP can expect accurate answers because niche forums attract experts in their field.

One problem with forums is that good answers are often buried somewhere in the discussion thread. Traditional forums do not provide functionality to re-order responses based on *correctness*, they merely list responses chronologically. Moreover, users are free to respond with *non-answers* such as: "I have this same problem!", "Can you provide more information?", "No idea, good luck.", and "Thanks, that solved it." None of these responses constitute *answers* to the OP's question. New visitors with the same question must invariably sift through non-answers to find answers. This often leads to the creation of a new thread with the same question and the ubiquitous non-answer response, "This has been asked many times, why not use the search function next time?"

Question and Answer (Q&A) sites are a recent trend that attempt to resolve these issues. Sites such as StackOverflow, Answers.com, Yahoo! Answers, and Quora emphasize *answer*-only responses to questions that are up-voteable by users. *Non-answers*, such as the examples above, are delegated to comment sections or down-voted from view.

It would be beneficial if responses on traditional forums could be up-voted automatically. In this project, we address a relaxed version of this problem. We seek to distinguish *answers* from *non-answers*. We define an *answer* to be any response that makes a reasonable attempt at answering the question.

We train a recurrent neural network (RNN) that assigns a probability to each question-response pair that indicates how likely the response is to answer the question. Responses with less than 0.5 probability are classified as *non-answers*. Responses may be re-ordered based on their probability assignments so that non-answers are moved to the bottom of the thread. A forum search engine that excludes non-answers increases the likelihood that new visitors will find the answer to their question in a reasonable amount of time.

Note that this work builds off of previous work performed by Aaron Abajian in CS 229 and CS 221. In those projects, a support-vector classifier and a logistic regression classifier were trained on the same classification task. We compare performance of the RNN to those models in the discussion section.

2 Background / Related Work

Question-answering (Q&A) systems have received considerable attention in the past two decades. A variety of models have been trained using decision trees [1], support vectors machines (SVMs) [6] [7], and recurrent neural networks [4]. The majority of these models are trained using large databases of facts and factoids (small snippets of knowledge). Given a question, these models compute a probability distribution over all known facts and return those facts with the highest probabilities.

The decision tree and SVM models typically relay upon n-gram feature vectors and cosine similarity. They may also include special features such as date indicators, parts-of-speech tagging, and named entity tagging. Their performance is limited because they cannot take into consideration long-distance interactions such as questions with multiple sentences and deep relationships between words, phrases and their context.

The neural network models, in particular [4], have been more successful than traditional n-gram based word-matching schemes because they are able to take into consideration the "syntactic glue" that gives meaning to a question and its context. Higher-dimensional learned word and phrase vectors have been shown to capture deep syntactical relationships [5]. For example, Pennington, et. al. show the close proximity to the word vectors for "queen - woman" and "king - man". Perhaps more poignant for question-answering is the ability of learned word vectors to identify new words that mean approximately the same thing. For example, "litoria" and "leptodactylidae" are both appropriate responses to the question, "What are some frog genera?" Their learned word vectors are both similar to "frog", a fact that is quite well hidden in n-gram only features.

Fact-based Q&A systems are inherently limited by their corpus of training data. This is a problem for both SVMs and RNNs. The Internet represents a vast corpus of knowledge, but the information content is not well-structured for Q&A model training. Recent approaches have wielded question and answer data available from sites such as Yahoo! Answers and Quora [2]. The content from these sites is presented in a format that lends itself to training. Models trained on Q&A site data have similar training objectives, but their space of viable answers is greatly expanded.

Yet, the scope of Q&A sites remains small in comparison to the ubiquity of online forums. Forums are structured websites that facilitate question answering, however responses to forum questions need not answer the original question. This leniency makes it difficult to use raw forum data to train a question answering system. Given a forum question, our goal is to distinguish responses that *attempt to answer* the question from those that do not. It is a simplification of question-answering in that we are not searching for the best answer to a question. Rather, our work focuses on the general structure of *non-answers* - responses such as comments, follow-up questions, and concluding remarks. By applying our classifier, *non-answers* to forum questions may be removed and the resulting question-answer pairs used for subsequent model training.

3 Approach

We train a recursive neural network (RNN) that labels responses to forum questions as *answers* or *non-answers*. The *answer* label indicates any response that attempts to answer the question.

Our RNN model is largely based on next-word prediction RNNs. Given the first few words of a sentence, these RNNs predict the next word by assigning a probability distribution over the vocabulary of words. Our model reads each question-response pair from left-to-right and produces a probability distribution over the two classes *answer* and *non-answer*.

3.1 Model

Let x be a question-response pair such that $x = x_1\ x_2\ \cdots\ x_n$. For example:

```
<Q> What color is the sky? </Q> <R> The sky is blue. </R>
```

In the question-response pair above, we have $x_1 = \texttt{<Q>}, x_2 = \texttt{What}, x_3 = \texttt{color}, \cdots, x_{13} = \texttt{</R>}$.

The *answer* and *non-answer* probabilities are given by:

$$P\left(y = k \mid x_1\ x_2\ \cdots\ x_n\right) = \text{softmax}\left(Uh^{(t)}\right)_k = \hat{y}_k$$

Thus, $\hat{y}_1$ is the probability of an *answer* and $\hat{y}_2 = 1 - \hat{y}_1$ is the probability of a *non-answer*. We define the recursive function $h^{(t)}$ similar to next-word prediction RNNs:

$$h_t = \text{sigmoid}\left(Hh_{t-1} + L_{x_t}\right)$$

Where L_{x_t} is the column of our word-vector matrix corresponding to the word x_t.

One difficulty with our approach is that we allow for an arbitrary window size. The question-response pair x has n tokens, resulting in n evaluations of h (e.g. t ranges from 1 to n). We assign a label to the question-reponse pair after the final token (e.g. x_n) is seen. Previous work has demonstrated that such models quickly lose memory due to vanishing gradients.

We address this problem in two ways - by limiting t to some fixed value $t_{\max}$ and by extending the model to include Long Short Term Memory as described by Hochreiter, et. al. [3]. In the latter case we rely heavily upon their implementation in Theano.

As noted above, our objective is to maximize the average probability of correctly labeled pairs:

$$\frac{1}{N}\sum_{i=1}^{N} y^i \cdot \log \hat{y}^i$$

Where N is the number of question-response pairs and y^i is a one-hot vector representing the true label of the ith pair, as recorded by Mechanical Turk workers.

4 Experiment

We trained our RNN classifier using question-response pairs from ten online forums. We examine the classification accuracy as a function of $t_{\max}$ (the maximum number of tokens of the questionr-response pair taken into account) and iteration number.

We also examine the words that give rise to maximal neuronal activation for the *non-answer* classification label.

4.1 Training Data

Our training data consists of question-response pairs from forum threads. As a simple example, consider the question, “What color is the sky?” and example responses, “The sky is blue.”, “I’m not sure.”, “It can be either blue or gray.”, “Why don’t you just Google it?”, “I think it’s purple.” These statements would lead to five training inputs:

```
[1, 0] <Q> What color is the sky? </Q> <R> The sky is blue. </R>
[0, 1] <Q> What color is the sky? </Q> <R> I'm not sure. </R>
[1, 0] <Q> What color is the sky? </Q> <R> It can be either blue or gray. </R>
[0, 1] <Q> What color is the sky? </Q> <R> Why don't you just Google it? </R>
[1, 0] <Q> What color is the sky? </Q> <R> I think it's purple. </R>
```

The first, third and fifth responses are *answers* (label [1,0]) while the second and fourth responses are *non-answers* (label [0,1]). The beginning and end of each question and answer are delimited with special tokens. Note that the last response is an incorrect answer, but it still constitutes an *answer*.

4.2 Data Acquisition and Preprocessing

We built a forum crawler and downloaded 59,205 question-response pairs from ten online forums. We used Amazon Mechanical Turk to hand-label each response as an *answer* or a *non-answer*. Turk workers were instructed to select responses that *attempted to answer* the question. We had two Turk workers label each response, and kept only those responses in which both workers agreed. We equalized the number of positively-labeled and negatively-labeled samples resulting in 27,682 question-answer pairs. We randomized the data and then split it into train (80%), dev (10%), and test (10%) sets. The data is summarized in the table below.

	Percentage of total data set	Number of question-answer pairs
Train	80	22,146
Dev	10	2,768
Test	10	2,768

The data is provided as a supplemental attachment. We preprocessed the question-response pairs by applying the following normalizations:

- Lower-case the text
- Remove all non-ASCII characters and punctuation
- Convert whitespace blocks to single spaces
- Stem words
- Remove common (e.g. stop) words
- Remove words occurring in less than 100 training samples

4.3 Evaluation Metric

We use the number of correctly classified question-response pairs from the test set as our evaluation metric. We also record the words that gave rise to maximal neural activation as a qualitative evaluation.

4.4 Results Without Memory

We initially included both the question (`<Q>...</Q>`) and the response (`<R>...</R>`), but early experiments showed that the response alone was sufficient to classify it as a *non-answer*. We hypothesize reasons for this in the discussion. We also found that our model lost memory after $t_{\max} = 5$ recursive steps (Figure 1). Thus we train on the first five words of each response (not counting the initial `<R>`). The results below are for our model *without* memory (i.e. no Long Term Short Term Memory). As shown in Figure 2, the model quickly learned to classify roughly 70% of the training samples during the first 1,000 iterations. Additional iterations saw gradual performance improvement with a plateau around 76.3%.

In order to qualitatively evaluate the trained model, we fed each vocabulary word into the network as a one-word sample. Encouragingly, words that were strong *non-answer* features for the SVM and LR models proved to be strong signals of *non-answers* in the RNN. In particular, the word "thanks" produced the highest *non-answer* signal, consistent with the fact that responses starting

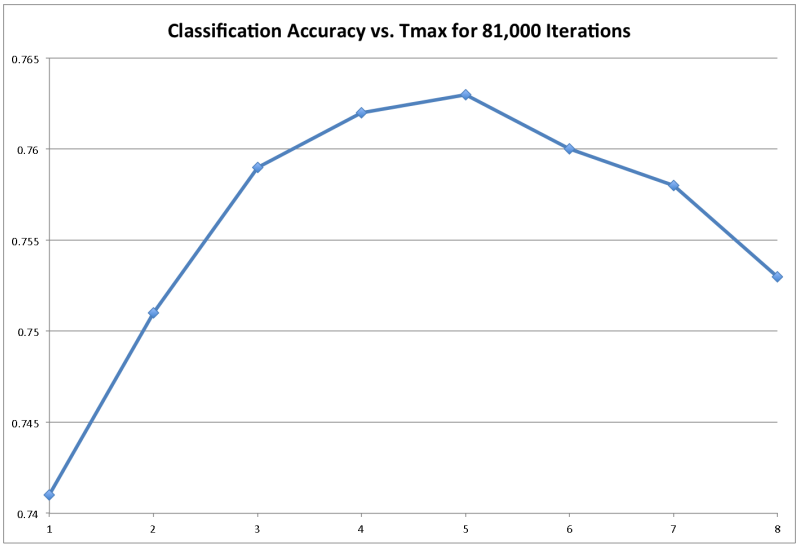

Figure 1: T_{max} denotes the number of question-response tokens taken into consideration. Due to gradient vanishing effects, we lose accuracy with $T_{max} > 5$.

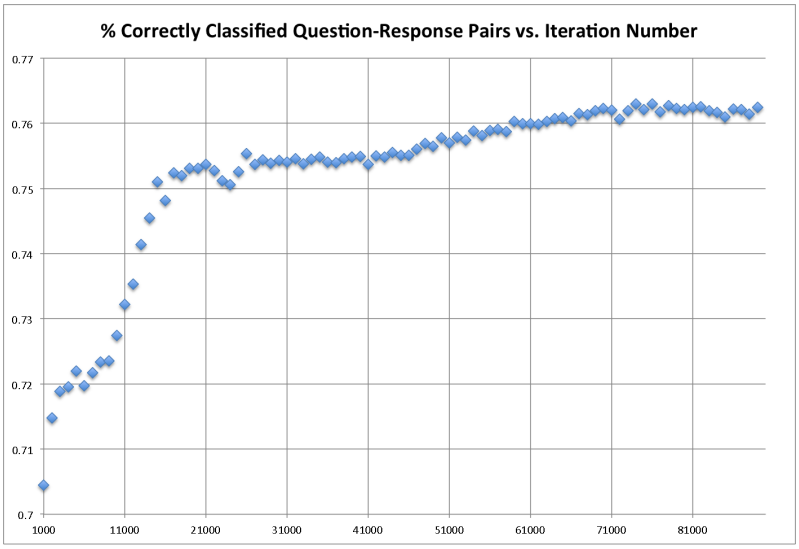

Figure 2: Accuracy as a function of training iteration for $T_{max} = 5$. The model learned quickly during the first thousand iterations, and then slowed.

with "thanks" are very rarely answers. A selection of *non-answer* words that qualitatively validate the model are provided in the table below.

Single-Word Input	Non-Answer Probability
thanks	0.83
bump	0.80
nevermind	0.77
inappropriate	0.65

Table 1: Words with high non-answer probability. Note that the first three of these single-answer responses are typically written by the original poster. In our SVM and LR models, we include an indicator feature that records whether or not the response was by the original poster.

4.5 Results with Memory

We attempted to include Long Short Term Memory using the Theano model by Hochreiter, et. al. In theory, this model should have significantly improved our results because it would resolve issues with vanishing gradients, allowing the model to read the entire question-response pair. We started with their sentiment code as a base and attempted to modify it to train our classifier. Unfortunately, we were not able to produce a converging model. The first few iterations produced a minor improvement (67% classification accuracy), but then stagnated at that value. We suspected a bug in our implementation, but ran out of time before we could resolve it.

5 Conclusion

We previously trained a logistic regression classifier and a support vector classifier to evaluate the feasibility of this problem. In those models, we were able to hand-select features beyond the textual data. For example, we indicated the author of the response, the index of the response, and whether the response was quoted by another response. The logistic classifier performed the best with a test-set accuracy of 89.0% using 7-gram features along with several novel features.

In SVM and LR models, we found that the question contributed little to the training accuracy. This result was confirmed by the RNN - we were able to achieve the same RNN classification accuracy using just the response text (`<R>...</R>`). This suggests a general structure of *non-answers* to forum questions. Our qualitative evaluation revealed that words such as, "thanks", "bump" and "nevermind" are strongly indicative of *non-answers*. Note, however, that these words are often written by the original poster, so the post author feature may be an even stronger feature available to the SVM and LR classifiers.

The utility of a response classifier may be demonstrated using a practical example taken from the Apple Technical Support Discussion Forum:

> **My Apple TV will not connect to WiFi, having put in the pass code it displays connection error -3905. Can anyone help?**
>
> 1. I too have an ATV2...I could only get it to work properly by uninstalling my McAfee security suite.
> **RNN Score: 0.21, LR Score: 0.615**
>
> 2. I get the exact same problem and error code with my NetGear DG384T wireless router; when I try to configure the Wireless network the Apple TV actually lists my wireless SSID but fails to connect when I select it from the list.
> **RNN Score, 0.10, LR Score: 0.291**
>
> 3. Take a chill pill, pal! Turn off MAC Address Filtering and you will instantly get a connection. I read a post by a housewife in Australia who nailed the problem - good on yer, Sheila!
> **RNN Score: 0.11, LR Score: 0.482**
>
> 4. I was having the same problem with my ATV 2nd Gen. Spoke to an Apple tech, determined that the cause of my problem was wireless interference...Apparently a nearby wifi network was using the same channel that mine was.
> **RNN Score: 0.33, LR Score: 0.726**
>
> 5. Thanks everyone for your help, I resolved the issue.
> **RNN Score: 0.052, LR Score: 0.103**

The last answer turns out to be the correct answer. Notice that the 1st and 3rd posts are attempts at answering while the 2nd post is a user who is experiencing the same issue. The last post is by the original poster. Both the RNN classifier and the LR classifier successfully reorder these posts 4, 1, 3, 2, 5. However, the RNN does not classify any of these responses as *answers* (probability < 0.5), likely because the first five tokens of each response is not sufficient to identify them as answers. We can see the strong *non-answer* score of the last response, almost entirely due to the fact that it starts with "thanks".

We found that the question (`<Q>...</Q>`) provided little insight into classifying responses as *non-answers*. We hypothesize that this is because *non-answers* generally have a very similar structure (e.g. they might begin with a token like "thanks" or "bump"). Since we were not looking for *correct* answers, the meaning behind each question was not as important as the structure of each response. Put simply, a response such as "Just Google it" is a *non-answer* to a great many questions, whereas it is an *answer* to very few.

We previously developed a cross-forum search engine that uses only *answers* (probability > 0.5). The search engine is available at:

`http://www.gotoanswer.com`

Note that this search engine relies upon the logistic regression classifier.

Our RNN did not perform as well as the SVM and LR classifiers, but there are quite a few modifications that would likely improve its performance. As a next step, we suggest reducing the question-response space to a specific topic such as computer hardware repair. It is likely that *non-answer* and *answer* responses within a particular subject share common structure. We saw this trend with our SVM and LR classifiers - posts within the Apple Discussion Forum that linked to specific support topics were *answers*, while the phrase "no hackintosh questions" was a common *non-answer*.

References

[1] David Azari, Eric Horvitz, Susan Dumais, and Eric Brill. Web-based question answering: A decision-making perspective. In *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*, pages 11–19. Morgan Kaufmann Publishers Inc., 2002.

[2] Benjamin V Hanrahan, Gregorio Convertino, and Les Nelson. Modeling problem difficulty and expertise in stackoverflow. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work Companion*, pages 91–94. ACM, 2012.

[3] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[4] Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. A neural network for factoid question answering over paragraphs. 2014.

[5] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation.

[6] Jun Suzuki, Yutaka Sasaki, and Eisaku Maeda. Svm answer selection for open-domain question answering. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics, 2002.

[7] Show-Jane Yen, Yu-Chieh Wu, Jie-Chi Yang, Yue-Shi Lee, Chung-Jung Lee, and Jui-Jung Liu. A support vector machine-based context-ranking model for question answering. *Information Sciences*, 224:77–87, 2013.