# Predicting Answer Time for YAHOO! Answers

**Pratik Vasa**
201205502
IIIT-Hyderabad, India
pratik.vasa@students.iiit.ac.in

**Adeel Zafar**
201205503
IIIT-Hyderabad, India
adeel.zafar@students.iiit.ac.in

**Aditi Malik**
201205537
IIIT-Hyderabad, India
aditi.malik@students.iiit.ac.in

## ABSTRACT

Community-based Question Answering sites, such as Yahoo! Answers or Baidu Zhidao, allow users to get answers to complex, detailed and personal questions from other users. These services accumulate large volumes of knowledge through the voluntary services of people across the globe.

Yahoo! Answers is one such community-driven question-and-answer (Q&A) site or a knowledge market operated by Yahoo! It allows users to both submit questions to be answered and answer questions asked by other users. It is a question-centric CQA site, as opposed to more social-centric sites such as Quora.

Askers post new questions and assign them to categories selected from a predefined taxonomy. A question consists of a title, a short summary of the question and a body, containing a detailed description of the question and even additional details. The posted question can be answered by any signed-in user. It remains “open” for four days, or for less if the asker chose a best answer within this period. If no best answer is chosen by the asker, the task is delegated to the community, which votes for the best answer until a clear winner arises. Only then is the question considered “resolved.” In case a question is not answered while “open” it is “deleted” from the site.

However, not all questions are answered within a short time frame since answering a question depends on the ability and willingness of users to address the asker's needs. This may cause undue anxiety for the questioner with regard to when he/she might be able to get the perfect answer.

In this project we try to develop a model that can be used to predict the how long certain new questions would take to be satisfactorily answered by a best answer. We leverage concepts and methods from natural language processing and machine learning in order to extract a wide range of features for our model.

We build a ground set of inductive rules from a corpus of Yahoo! Answers, and experimental results show that our method gives an accuracy of upto 72% in most cases.

## MOTIVATION FOR THE PROBLEM

The set of use cases that we envision for our system would be as follows:-

1. Q&A communities, such as YAHOO! Answers provide a limited time only for a question to be answered. This time period, 4 days in case of YAHOO! Answers might not be enough for questions that might belong to esoteric fields or ones that may require an in depth research. In this case our system could be used to predict how much time a certain question might need to be answered in order to give a “grace” extension period. This will allow for a greater number of questions to be answered and lead to a better user experience.

2. As opposed to search engines, Q&A websites offer a richer source of information for users. Asking a question on these forums usually results in replies that are good in quality in terms of research done by the answerer and the pertinence of the answer to the question. Unfortunately, Q&A websites aren't the best of the sources to get information from in time critical situations since answerers need some time to upload the answer. Here our system of predicting the answer time can be used by a questioner to infer how long their question would take to be answered. Thereby reducing the chronometric uncertainty in using such forums.

## PROBLEM DEFINITION

Our task, in this project, was to develop a model that could simply predict how long a question, put up on a community based Q&A site, would take to be answered.

This seemingly overwhelming task of predicting the future can be reduced to a simple data mining and machine learning task. Thus we set our proximate goal as the leveraging of concepts and methods from natural

language processing and machine learning in order to identify/extract the set of features of a question that are correlated to/affect the time it takes to get the best answer.

From these set of extracted features, we intend to train a model. This model could take as input a question and its related meta-data and respond with the predicted time the question might be responded in.

## PREVIOUS AND RELATED WORK

Some research has been already been done on predicting response times for social media as well as forums like stack overflow.

Shtok et al. [1] have used metrics such as intra-question similarity, inter-question similarity and question-answer similarity to find an answer to an unanswered question from the answers of previously answered questions.

Mahmud et al. [2] describe work on estimation of response time in Twitter. They built a predictive model for the estimation of the answer times using history of user specific wait time, generalized wait time for all users and a time sensitive model which takes into account the time of the day a query was put up.

Yang et al. [3] proposed a method of prediction if a question will be successfully answered or not within a given timeframe. Using two main types of features, basically content features, like topics extracted using SLDA, and heuristic features such as question length, category, category matching, asker history, time of asking the question, and question subjectivity, To create classifiers.

Asaduzzanam et al. [4] also address the problem of prediction of time it takes for a question to be answered for another Q&A website, stackoverflow. They categorized questions into predefined bins each representing a timeframe. Using features such as title length, post length, tag similarity, readability etc. they built a classification model which could predict which of the bins the question belonged to.

## PROPOSED APPROACH

The numeric prediction problem that we have at hand can be easily converted into a classification problem. This is done by dividing the future timeline into discrete bins. The size of these bins would grow near exponentially to account for the sparsity of questions towards the later end of the timeline. We used three different types of binning in order to test our results and to remove any bias that might result due to the selection of the bins.

| | Bin durations | | |
|---|---|---|---|
| **Bin No.** | **Type I** | **Type II** | **Type III** |
| 0 | 0-5 mins | 0-30 secs | 0-10sec |
| 1 | 5-10 mins | 30sec - 1min | 10-30sec |
| 2 | 10-30 mins | 1-2mins | 30sec-1min |
| 3 | 30 mins-1hr | 2-4mins | 1-2mins |
| 4 | 1-2 hrs | 4-15mins | 2-3mins |
| 5 | 2-5hrs | 15-30mins | 3-5mins |
| 6 | 5-10hrs | 30mins-1hr | 5-10mins |
| 7 | 10 hrs - 1day | 1-2hrs | 10-15mins |
| 8 | 1day - 2days | 2-5hrs | 15-30mins |
| 9 | N/A | N/A | 30mins-1hr |
| 10 | N/A | N/A | 1-2hrs |

Do note that the first two binning strategy follow an exponential increase in bin sizes whereas the third one follows a more selected growth pattern chosen by us to classify our dataset more evenly.

Our model would now be a classification model which would try to classify the questions into the above mentioned bins.

The next task at hand was to identify the set of features that would help us predict the answer time. We incorporated two types of features into our models.

**Extracted Features:** These are features that are already available within the data set. These include:-

1. Category of the question: As defined by the 10 toplevel categories defined by Yahoo! Answers.
2. Subject length: Represented by the number of characters in the subject.
3. Description length: Measured in number of characters in the description.
4. Asker history: Average time taken for the questions posted by the asker in the past being answered.
5. Question time: The number of users online varies with the time inasmuch it affects the chances of a question being answered. Question time is the hour of the day when the question was asked.
6. Question day: The day of the week the question

was asked on.

7. Question month: The month of the year the question was asked in.

**Calculated features:** These are the features not directly available in the given corpus but those that need to be calculated from the information within the dataset. These features are as follows:-

1. Readability measures: These are metrics used to classify whether a document is easily readable or not. We used two readability measures, namely Flesch and Fog based on McCallum and Peterson [5]. Both types of readability scores are kept for both, the subject as well as the description. Hence we get four readability measure scores:

   i. Subject Fog score

   ii. Subject Flesch score

   iii. Description Fog score

   iv. Description Flesch score

   In case of missing description or subject we take the average value of the scores over the whole corpus to be the corresponding missing score.

2. Topic labels for Questions: Although we have the top level categories of questions, we need a finer grained categorization for the questions, we do this through topic inference and modeling using LDA.

Since the corpus we have, consists of questions from the same category, namely "Other-Internet", the first feature, top level category of the question is of no use. Hence we have to rely upon the topic modeling phase to produce subcategories for us.

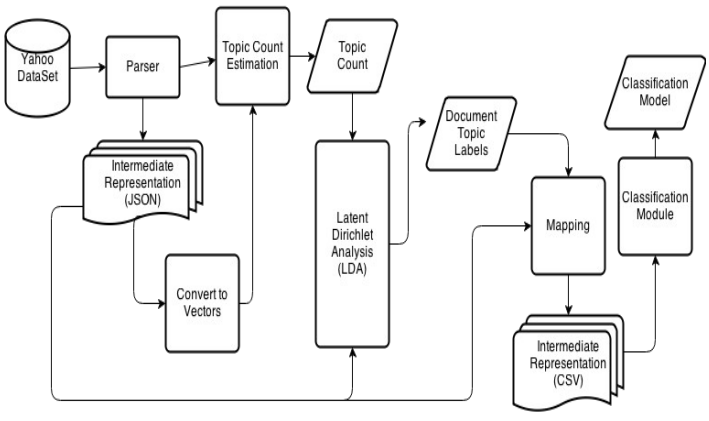


*Fig 1.* Flow Diagram

The above given flow diagram shows how our system works. The execution of the system is dived into various phases.

In the first phase we parse the given dataset to extract the useful features we find and save them in an intermediate format (JSON). This intermediate format is now used as input to the other phases.

The second phase involves converting the intermediate representation into a document vector form. This is done by appending the question and the description into a single string and then tokenizing it, lower casing it and then stemming the tokens procured using the Porter Stemmer. This document vector form is then input into the third an fourth phase.

The third phase of our system is to estimate the number of topics that need to be given to our topic inference phase as input. We take the document vectors obtained in the previous phase and do a simple word count, obtaining a histogram of the count of words with specific frequencies. We take the middle quartile to represent the number of unique ideas within the corpus. This number is taken to be the number of topics within that corpus.

In the fourth phase we take the output of the second and third phases as the inputs and apply Latent Drichlet Allocation on the documents. This gives us a list of topic models that can then be used to classify the documents into subcategories.

The fifth phase is just a representation transformation and mapping phase where we take the outputs of the first phase and fourth phase, map them to each other and then convert it into a format suitable (CSV) for input into our sixth and final phase.

The sixth phase is where we train our model. Using supervised learning algorithms, we created predictive models on our given dataset. In order to test our selection of features, we used three different classification algorithms, namely Naive Bayes, Decision Trees and Maximum entropy modeling.

## DESCRIPTION OF DATASETS

The dataset that we used in our project was provided to us beforehand. It was a 30 MB XML dump of 6000 yahoo questions and their related meta data. Some salient features of the dataset:

1. All of the questions belonged to a single category, name "Other-Internet".

2. All questions were successfully answered.

3. Most of the questions got their best answer

within a few minutes of being asked.

4. All questions and their responses were in English.

5. The questions and their descriptions had all been spell-corrected beforehand.

6. Informal English was used in most questions.

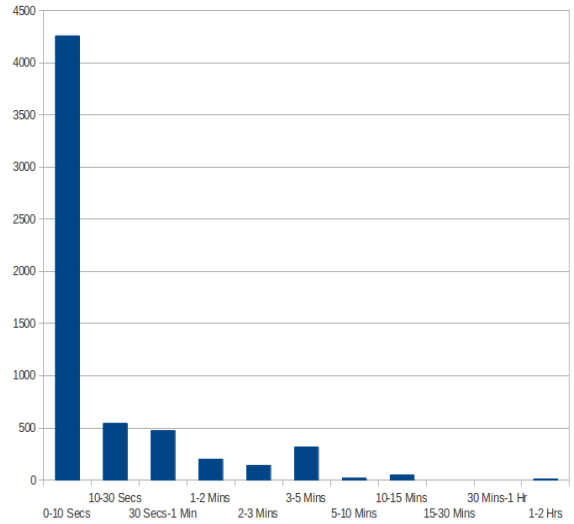


*Fig 2.* Bin Distribution for Type III bins

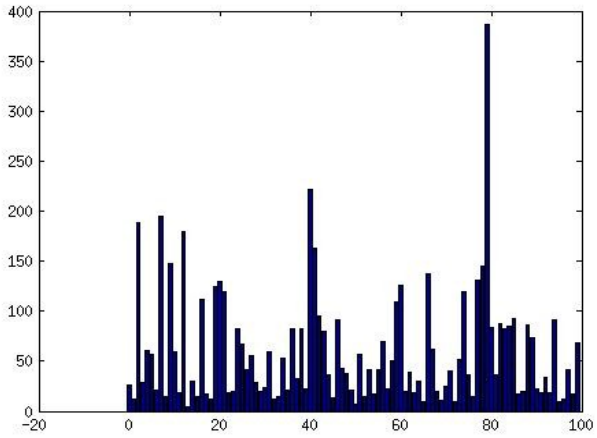


*Fig 3.* Topic Distribution over the corpus

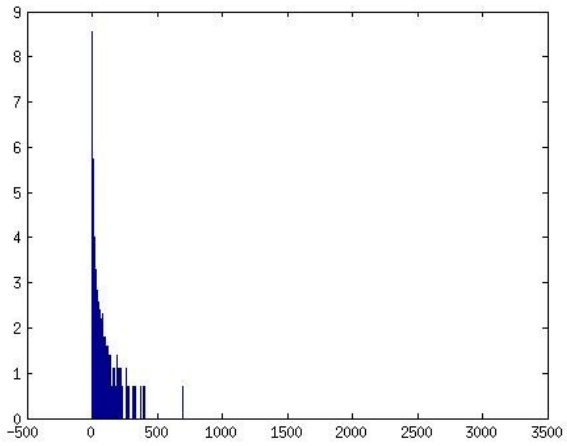


*Fig 4.* Word Count Histogram

## EXPERIMENTAL RESULTS

We performed 10-fold cross-validation on our results where 90% of the data was used for training and the rest of the 10% of data was used for testing. The input set was divided into 10 equivalence partition sets.

The members of this set were chosen randomly to avoid any form of bias that may be due to the different partitioning schemes. At a time 9 of the sets were used to train the classifier and the remaining 1 set was used to test it. This process was repeated for the remaining 9 sets keeping them as testing sets. The accuracies of all these types are then averaged to get the accuracy for a particular class of classifier.

Using the cross-validation API provided by Mallet, a document analysis tool, all these experiments were automated and performed for one hundred iterations. The results of all these iterations were the averaged together in order to iron out any stochastic anomalies.

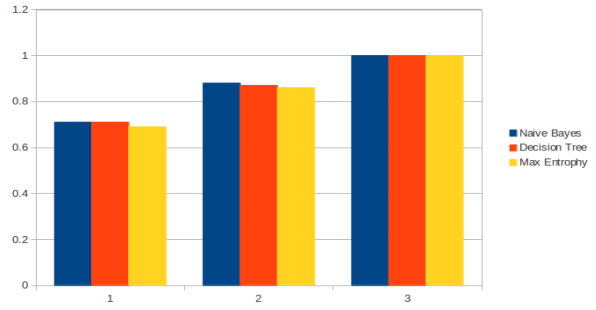


*Fig 3.* Classifier accuracy

## ANALYSIS OF RESULTS

One of the first things that we observe is that there is a definite and large variation between the results when the bining type is changed.

In the case of Type I binning we get cent percent accurate results, this is due to over fitting of the model since most of our data points lie within a single bin, namely the first one, the classifier comes up with a single "null" rule which places all documents into the single bin without making any decision

The second and third binning types give us more realistic results due to the finer detailed binning we did there. Type II bins get an average accuracy 85% and Type I bins give an accuracy of roughly 75%. Though we do observe that the smaller the finer the binning, the lower the accuracy as the model starts avoiding over-training. As per our hypothesis, this accuracy measure would converge upon a final value which would be the definitive accuracy of the system.

Finally, we observe that the different classifiers performed almost similarly when compared over a single bin type. This we thing

### CONCLUSION

In conclusion of our experiments we found that classification models can be used to abstract the predictive nature of numerical models to a certain extent, although the initial binning has a lot of affect on over-fitting and under-fitting of results. Most classification models performed equally well and had a high rate of accuracy when it came to making predictions.

## FUTURE WORK

In this project we worked on a single dataset, i.e. YAHOO! Answers. In the future we would like to test the validity of our system in other forums too and on forums in languages other than English.

## REFERENCES

[1] Anna Shtok, Gideon Dror, Yoelle Maarek, Idan Szpektor. Learning from the past: Answering new questions with Past Answers. *Journal of the Association for Computing Machinery*, 759-768, April 2012.

[2] Jalal Mahmud, Jilin Chen, Jeffrey Nichols. Estimating response time in Twitter. *IBM Journal of Research and Development*,2013

[3] Lichun Yang, Shenghua Bao, Qingliang Lin, Xian Wu, Dingyi Han, Zhong Su and Yong Yu. Analysing and predicting not answered questions in community based question answering services. *AAAI conference on Artificial Intelligence*, 2011.

[4] Muhammad Asaduzzaman, Ahmed Shah Mashiyat, Chanchal K. Roy and Kevin A. Schnieder. Answering Questions about unanswered questions about StackOverflow. *MSR 2013*.

[5] Douglas R. McCallum and James L. Peterson. Computer based Readablity Indexes. Journal of the Association for Computing Machinery, 44-48, 1982.