Ranking Question-Answers Over Social Media

Arjun Malik
Manish Kumar Singh
Dept. of Computer Science and Engineering
IIT(BHU) Varanasi

Supervised By: Prof. Anil Kumar Singh
Dept. of Computer Science and Engineering
IIT(BHU) Varanasi

ABSTRACT

Due to the recent popularity of community question-answering, websites and forums involving information-sharing activities have become potentially great sources of information retrieval in a very specific and targeted fashion. Our work explores and builds on the previous work done in this field to effectively grade question-answer pairs on community websites like Yahoo! Answers and Quora on the basis of their quality and relevance to the asker's query. This kind of grading has been developed in the form of a ranking function that can compare any two given query-question-answer tuples involving the same query to decide the relative acceptability of the question-answer pairs to the query based on a variety of factors.

I. Introduction

Community Question answering sites are forums where a person can ask a question and another person who has relevant information may answer it. This kind of QA is very different from search engine queries as it provides other people's views, opinions or information in the form of self-contained answers in natural language and not links to related texts which the user may go through to find a relevant answer to his query. Therefore this kind of question answering is far more precise than general purpose web search. With the extensive archive of answers that these sites have created in the past few years, an efficient framework for ranking answers on the basis of relevance and quality is needed. Quite a large portion of these answers are actually unsubstantiated or biased opinions or misleading information and to retrieve good quality, relevant and precise factual information from these answers is quite a challenging task. In the QA systems used nowadays any user plays a threefold role on the system : asker, answerer and evaluator. This incorporates a lot of positive feedback in the whole system which can be effectively exploited to automatically measure and test the relevance and quality of answers. However this kind of feedback has its own com-

plications when the credibility of the users is itself questionable. This is an issue that concerns almost all community QA websites as a large fraction of the content on these websites reflects often unsubstantiated and possibly biased opinions of users which are not useful for factual information retrieval and question-answering. In that case we must be careful to incorporate a measure of the credibility and usefulness of a user's evaluation and his/her answers in the form of user's ratings.

II. Generating the dataset

We used factoid questions from two years of the TREC QA track evaluations (years 1999-2000). The text file downloaded from http://trec.nist.gov/data/qa.html was processed to extract a list of queries.
A python based application framework, Scrapy was used for crawling Yahoo Answers! and extracting structured data. Each query from TREC dataset was submitted to the Yahoo Answers! search engine and upto 10-top-ranked related questions according to the Yahoo Answers ranking were retrieved. The list of queries were read and requests were made to crawl the Yahoo Answers! using scapy.Request() API.The html and xml webpages retrieved were then parsed to find relevant question links.
From each of these Yahoo question links, all the answers were retrieved only from the first page among the set of Yahoo Answers! webpages on which answers may be listed.This was done to ensure that only relevant answers are extracted. Again requests were made for each question link using scrapy.Requests() and the retrieved answer webpage was parsed to extract the answer and related features available on the same webpage. Thus, in total, 123750 <query,question,answer> tuples were generated from the 893 TREC queries.

III. Processing the data

The data extracted from Yahoo! Answers was further processed to bring it in a form from which the further extraction of features is convenient. For this purpose, the following steps were taken:

- All blank lines between the text were removed.
- Punctuation was stripped from the entire text.
- Only those queries were retained which had valid question answer pairs along with them.
- Only those questions were retained which had at least one valid answer and the popularity of the question.
- Only those answers were retained which had total number of upvotes and total number of downvotes for the answer and the lifetime of the answer.

After this initial processing, all possible tuples of query-question-answer were extracted from the text file and stored in a python list. This processed data was then used to generate features.

IV. Generating Features

The next step was to generate features from the completely processed data. This task required deciding the features that could be extracted. The following features of the data were the most prominent to measure the relevance and quality of question-answer pairs:

- Number of similar terms in query and question in any given tuple
- Number of similar terms in question and answer in any given tuple
- Number of similar terms in query and answer in any given tuple
- Length of query
- Length of answer
- Popularity of the question
- Lifetime of the answer
- Upvotes for the answer
- Downvotes for the answer

These extracted feature vectors were stored along with the index of the query to which they corresponded in a class. All these features corresponding to different queries were mixed at this point. However for applying the

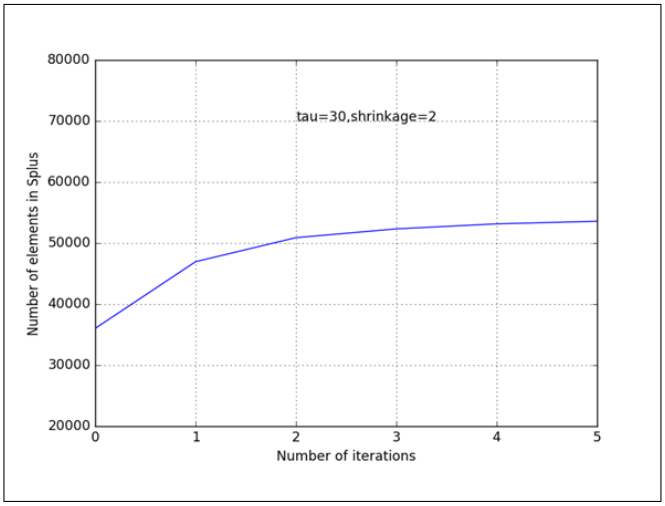
GBRank algorithm we needed these features segregated by the query they were indexed with. Therefore another function to segregate features by their queries was created and the features were then segregated by the query they were indexed with.

V. GBRank Algorithm

Start with an initial guess for the function h0. Although there can be many different possible choices for the initial guess of the ranking function, in our implementation we started out with the initial guess as simply the sum of the features.
STEP 1 : Using h_{k-1} as the current approximation of h, S is separated into two disjoint sets S^+ and S^-. S^+ is the set of those examples for which $h_{k-1}(x_i) \leq h_{k-1}(y_i) + \tau$. S^- is the complementary set.
STEP 2 : Using Gradient Boosting Tree fit a regression function $g_k(x)$ for the following training data :
$((x_i, h_{k-1}(y_i) + \tau), (y_i, h_{k-1}(x_i) - \tau)) for (x_i, y_i) \in S^-$
STEP 3 : From the function obtained in step 2 update the ranking function to minimize the squared hinged loss function using the following update rule:

$$h_k(x) = \frac{k h_{k-1}(x) + \eta g_k(x)}{k+1}$$

where η is the shrinkage factor.
The quality of the so obtained function $h(x)$ can be evaluated by the number of query-question-answer tuples it ranks correctly. This number is the size of the list Splus obtained after training the function $h(x)$.



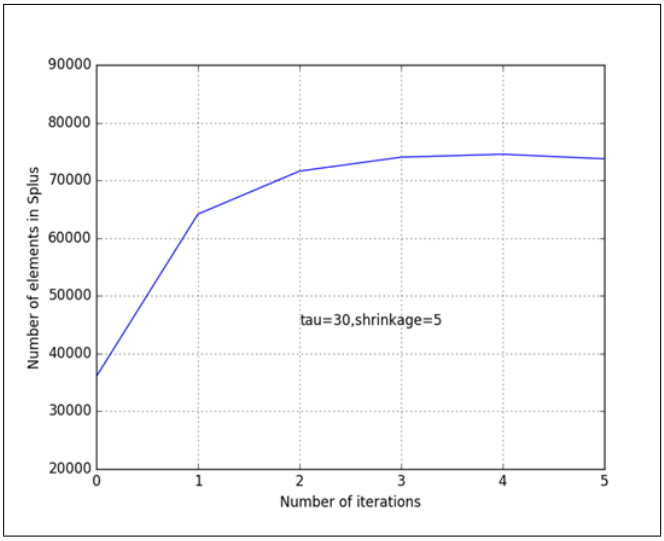
VI. Results

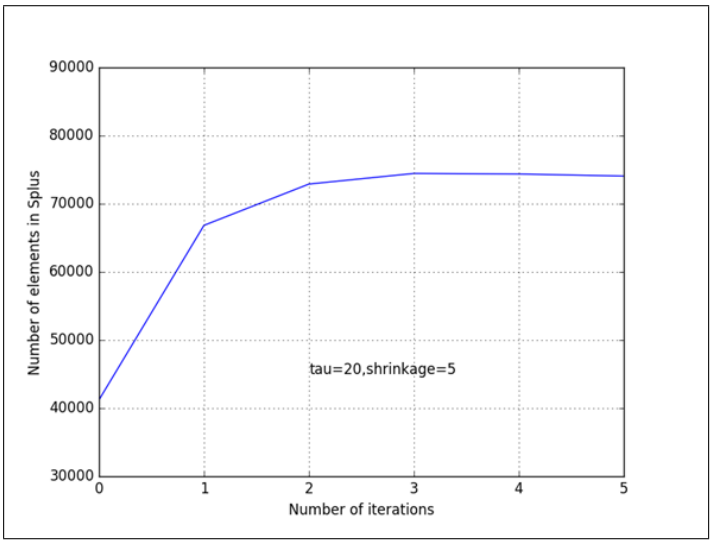
We ran several experiments on the datasets extracted from Yahoo! Answers and TREC datasets. Overall the corpus on which we trained our learning function had about 1000 different queries and after all the processing of data and extraction of features, we obtained about 125000 different query-question-answer tuples. The computation time involved in completing a single experiment was about 3 hours and the following observations were recorded: 1. Size of the list Splus has a heavy dependence on the shrinkage factor. With increase in shrinkage factor, the quality of function h(x) improves rapidly in a lot less number of iterations. 2. The factor tau introduces an element of large margin classification in the function h(x). Increasing tau also improves the quality of the function upto a certain limit. However increasing tau beyond a limit causes size of Splus to fluctuate and hence tau cannot be increased too much. The following graphs were obtained for different values of the constants tau and the shrinkage factor.

VII. Future Work

The kind of feedback incorporated in the community QA websites due to user ratings of

answers and the popularity of questions has its own complications when the credibility of the users is itself questionable. This is an issue that concerns almost all community QA websites as a large fraction of the content on these websites reflects often unsubstantiated and possibly biased opinions of users which are not useful for factual information retrieval and question-answering. In that case we must be careful to incorporate a measure of the credibility and usefulness of a user's evaluation and his/her answers in the form of user's ratings.





References

[1] Jiang Bian, Yandong Liu, Eugene Agichtein, Hongyuan Zha. Finding Right Facts In The Crowd .

[2] Deepak Ravichandran and Eduard Hovy. Learning Surface Text Patterns .

[3] J Friedman. Greedy Function Approximation: A gradient boosting machine .

[4] Dell Zhang. Question Classification using Support Vector Machines
newblock .

[5] Jimmy Lin and Boris Katz. Question Answering from the Web Using Knowledge Annotation and Knowledge Mining Techniques .