Custom Content Delivery System for Stack Overflow

Final Report

Alok Kucheria
Department of Computer Science
NC State University
akucher@ncsu.edu

Raman Preet Singh
Department of Computer Science
NC State University
rpsingh2@ncsu.edu

ABSTRACT

Stack Overflow is a privately held website. It was created to be more open alternative to earlier QnA sites. It features questions and answers on a wide range of topics in computer programming. The content on the website can range from latest topics to age old questions that are ones which are frequently asked. Keeping up with the content thus can be quite cumbersome. A solution that we present to curb this problem was a customized content delivery system for Stack Overflow. It takes as user input data acquired from various users for customizing the content they would like to receive from Stack Overflow. Using those as reference for creating relevant output, we create three different models using archive data as input for training our models and testing the results in real-time on users through feedback

Keywords

Stack Overflow, StackExchange,Recommendation, Customization, Filtering, n-gram, Random Forest, ranking,temporal data

1. INTRODUCTION

In this project we extend the focus to the popular Question-Answer platform, Stack Overflow. Stack Overflow features a large corpus of knowledge and content covering many areas of Computer Science and Software development in particular.

As of now, Stack Overflow provides only a QnA interface. A user can subscribe to a weekly newsletter but has no control over its content. There isn't any feedback mechanism in place for Stack Overflow to know if the user is satisfied or not with its newsletter.

Another topic filter option exists. But it is hidden deep within account settings. It means that a user has to be registered, can subscribe to specific topics but gets only the upcoming questions of that topic every few hours or a day. It is interesting for a issue a user faces for a new release of a software/framework etc and is waiting for its solution. It does not really help in knowledge discovery.

The project captures user sentiments by providing a list of questions to them, answers to which help determine what the user is particularly interested in. The answers are then fed to the algorithm which calculates the right set f information to be delivered to the user. The information is then converted into a PDF and delivered to the user inbox. The user provides feedback as to how relevant the information is to them which is then fed to the algorithm and fine tune the next set of newsletter which is to be delivered.

2. DATA

2.1 Overview

For the creation of a recommendation engine on Stack Overflow, we had two initial options to gather data.

1. Archive data provided from StackExchange
2. Actively scraping data from the website

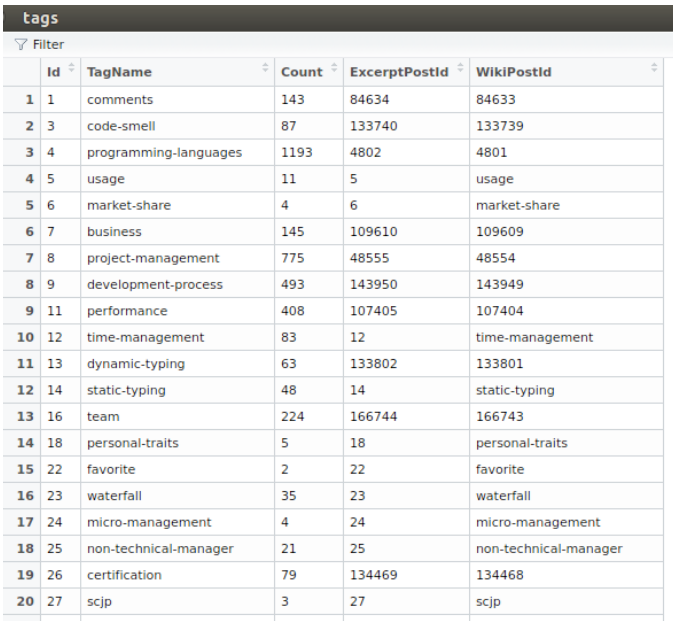
While a simple filtering algorithm would have worked well on scraped data, but random forest and n-grams would suffer from insufficient knowledge. Recommendation engines generally require a large amount of data for training and validation before they can be put to test.

Scarping data from the website in real-time would be able to give us updated data but at the cost of reduced efficiency in the system due to the small dataset.

We decided to go with archive data available publicly under the Attribution-ShareAlike 3.0 license made available at archive.org by Stack Exchange Inc. [?]

The complete dataset for all Stack Exchange sites stood at 26.4 GB in compressed format. About 20.4 GB of it was Stack Overflow data. After decompression, the data set stood well over 60 GB of XML files divided in various categories.

For the sake of timely completion, we choose to use a smaller subset to complete the training process in time. We took up data from programmers.stackexchange.com which is a website very similar to Stack Overflow. Both are part of Stack Exchange and both have similar variety of data. The


tags

Filter

	Id	TagName	Count	ExcerptPostId	WikiPostId
1	1	comments	143	84634	84633
2	3	code-smell	87	133740	133739
3	4	programming-languages	1193	4802	4801
4	5	usage	11	5	usage
5	6	market-share	4	6	market-share
6	7	business	145	109610	109609
7	8	project-management	775	48555	48554
8	9	development-process	493	143950	143949
9	11	performance	408	107405	107404
10	12	time-management	83	12	time-management
11	13	dynamic-typing	63	133802	133801
12	14	static-typing	48	14	static-typing
13	16	team	224	166744	166743
14	18	personal-traits	5	18	personal-traits
15	22	favorite	2	22	favorite
16	23	waterfall	35	23	waterfall
17	24	micro-management	4	24	micro-management
18	25	non-technical-manager	21	25	non-technical-manager
19	26	certification	79	134469	134468
20	27	scjp	3	27	scjp

Figure 1: Snapshot of tags dataset

primary difference being that programmers.stackexchange is focused on working professionals whereas Stack Overflow is a free-for-all.

2.2 File Details

There were seven xml files containing data related to tags, badges, users, comments, votes, posts, post history and post links. Each xml has expanded to include the following data along with a unique id for each entry:

Tags: tagName, count, excerptPostID and wikiPostID

Badges: ID, userID, name, date

Users: reputation, creationDate, displayName, lastAccessDate, websiteURL, location, aboutMe, views, upvotes, downvotes, age, accountID

Comments: postID, score, text, creationDate, userID

Votes: postID, voteTypeID, userID, creationDate

Posts: postTypeID, acceptedAnswerID, creationDate, score, viewCount, body, ownerUserID, lastEditorUserID, lastEditDate, lastActivityDate, title, tags, answerCount, commentCount, favoriteCount, closedDate, communityOwnedDate

PostHistory: postHistoryTypeID, postID, revisionGUID, creationDate, userID, text

PostLinks: creationDate, postID, relatedPostID and linkTypeID

Relevant data included information like tagName, count, postID, acceptedAnswerID, viewCount to count a few. The complete set is a lot of data and can give us more information than is possibly required for our task. We decided on the important variables for our system and used certain fixed parameter like tagName, postID, userID, viewCount, upvotes etc for specific filters as required by users for simple filters and textual and tag features for the learning algorithms. Use has been expanded upon in Section 3 for each algorithm.

3. ALGORITHMS

We decided to implement the following three algorithms for our system.

3.1 Simple Filters

We sorted the data according to user input requirement and presented it as output in an appropriate format. This isn't a unique algorithm per se. This is just a simple implementation of filters using available data to provide the user with what they ask for rather than what they might ask for.

Having asked the user for relevant tags, we extract the tag id from the tagName. Using this tag id, we extract the post information for each post. Using a list of tag id helped us to prioritize posts with the maximum number of matching tags.

These posts are then sorted according to the number of votes for every question-answers post. This gives us the top-voted question and answers. For most active posts, we sort the data by commentCount. Similarly, favorites are counted using favoriteCount.

The data itself was divided based on creationDate to make sure the users do not get the same content again and again.

3.2 Random Forest

Random forests is a general technique of creating decision trees on subsets of data. We generate feature vectors for the training and test data using various techniques. The most common technique used for textual information is tf-idf. It defines how important a word is to a document in a collection like our set of posts.

Using this to train our classifier, we build a model based on the posterior probabilities. This is, in a way, an expansion of the Naive Bayes as our data set is large and diverse.

Using the probabilities, we sort the output and present it to the user as per the input fed into the system per user requirement.

3.3 n-grams

An n-gram is a contiguous sequence of n items from a given sequence of text or speech. The items can be phonemes, syllables, letters, words or base pairs according to the application. The n-grams typically are collected from a text or speech corpus.

Two benefits of n-gram models are simplicity and scalability; with larger n, a model can store more context with a well-understood space time tradeoff, enabling small experiments to scale up efficiently.

Using n-grams, words can be predicted for relevancy from

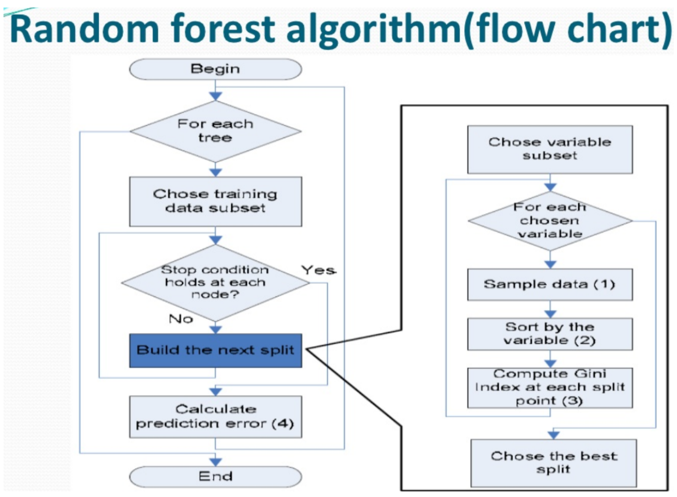

Random forest algorithm(flow chart)
Begin
For each tree
Chose training data subset
Stop condition holds at each node?
Yes
No
Build the next split
Calculate prediction error (4)
End
Chose variable subset
For each chosen variable
Sample data (1)
Sort by the variable (2)
Compute Gini Index at each split point (3)
Chose the best split

Figure 2: Random Forest flowchart src:Musa Hawamdah

the previous n words. Clusters can be formed using this information to extract the relevant information without solely relying on tags and making sure the actual content of the post matters.

For our data, this seemed like an appropriate implementation given the huge text corpus in every question, answer and the discussion that follows. Hence n-grams was used for a complete post rather than just tag based filtering or vote based sorting. This led to increased quality of results at the cost of high resource consumption.

We constructed n-grams from the posts and n-grams from the user input data. We calculated the minimum distance between the text from each of these and form the output posts based on minimum distance.

4. EVALUATION PLAN

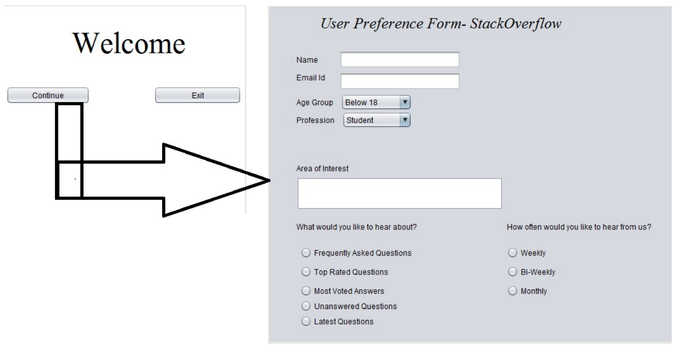
The user survey was the initial piece of the puzzle that was aimed at learning about their sentiments about the Stack Overflow website in general. How comfortable they were with the website, how often do they use it and whether a recommender system like that would really be of any use to them. Based on the results of the initial survey we were convinced that there is a need for such a system and went ahead with it. When we rolled out our first set of results a user feedback form was sent along with which captured user sentiments about the whole concept. Observing their feedback helped us understand the system and what improvements to do to make the system a more user friendly one by providing the right set of information. This was done by getting a rating for each and every report and feeding that rating to the system. Also a user feedback box which had reviews from the user about each of the reports, were also read to understand the shortcomings for each of the report. So we continued this practice of delivering reports to the user inbox over three cycles and finally the best algorithm over the period was chosen and used to generate report for the user going forward. A complete summary of each of the processes are explained in the subsections.

4.1 User Survey and Survey Results

We begin by collecting demographical information about the participant, such as age group and profession. We then solicit information about the participantâĂŹs trends of usage on StackOverflow. Questions include the userâĂŹs areas of interest, their frequency of use of the website, and what they use it for. Having collected this generic information, we then proceed to more specific queries about our proposed area of work. We ask if the user is aware of the existing StackOverflow newsletter, and if so whether it keeps them adequately informed on topics of their interest. Finally, we ask if the user would like a customized newsletter from the website, the parameters of which they may modify.

The survey was conducted on a user base of 150+ people. We estimate that users on average spent âĽĽ5 minutes to read the description, understand the purpose of the survey and answer the questions. We summarise the results below

- Students and professionals comprised âĽĹ80% of the participants.
- Only 11% of our survey participants were from academia.
- Almost 90% of the respondents were in the age range 18-25 or 25-50. This reaffirms our initial observation that most users of the website are either students or professionals.
- Our respondents listed many areas of interest including programming languages(Java, Python, C/C++), databases, networking, operating systems etc. took the survey. This indicates the diversity of areas that StackOverflow provides knowledge in, and the breadth of users who consider it a reliable source of information.
- The respondents who used stackoverflow everyday or at least 2-3 time a week formed 77% of the user base with everyday users comprising 37%. This speaks to the credibility of our user pool, since the most frequent users of the website would be the best informed as to how StackOverflowâĂŹs corpus of knowledge may be leveraged.
- 80% of our survey audience used stackoverflow for specific programming errors and 48% use it to learn languages and algorithms. It is noteworthy that despite the proliferation of online tutoring portals such as Coursera and Khan Academy, almost half our respondents still utilize stackoverflow.com to learn new things.
- The fact that 80% of respondents use StackOverflow for programming errors is overwhelming evidence that the platform is the best available choice for information on debugging in general.
- 75% of the respondents reported that they would like to hear from StackOverflow. This result reaffirms our motivation to provide a content recommendation system for the site.
- 60% of our user pool reported that they were dissatisfied with the existing StackOverflow newsletter. Indeed, 80
- Our survey had a total of 154*12 = 13 people hours(approx).



Name
Email Id
Age Group Below 18
Profession Student
Area of Interest
What would you like to hear about?
- Frequently Asked Questions
- Top Rated Questions
- Most Voted Answers
- Unanswered Questions
- Latest Questions

How often would you like to hear from us?
- Weekly
- Bi-Weekly
- Monthly

Figure 3: Applet for user interaction

4.2 Telemetry

To compare our various solutions, we must compare it from two different aspects, performance and user satisfaction. Here are the observations from the three algorithms:

4.2.1 Time Complexity

As suggested in the introduction, we are working on static data instead of scraping it from the website in real time. This means the complexity cannot be measured in real time either.

In case of simple filters, we only need to sort the data using certain parameters and present it. Although the number of sorts required may be high, it is still computationally simple.

For n-grams and random forests, building the model itself requires a lot of time. Even on the scaled down data from programmers.stackexchange.com, it can take days to create full fledged systems if multiple inputs are fed in to generate output for each user simultaneously. Given n users and m tags, we will need to create n-grams for each of them and compare edit distances for the entire set. Hence, n-grams increases in complexity with the increase of the data set size.

From the perspective of time, simple filter based approach works the fastest followed by random forests and n-grams.

4.2.2 Space Complexity

Although not very important from the end user perspective, this is an important consideration from an engineering perspective. As was the case with time, so it is with space.

Simple filtering approach required the minimum amount of computational power.

Between random forests and n-grams, we again notice that n-grams takes a lot of memory. This is because n-grams has to store all the distances for every iteration.

4.2.3 Accuracy

Having talked about time and space complexities, the algorithmic perspective is covered. But for a user who might require such a data on a weekly or biweekly basis, the time and space complexity may not be of much concern to them. What would really matter is how good the data is.

In terms of accuracy, n-grams winds hands down. It achieves near perfect output for users compared to over 90% for random forests. n-grams give a good reesult at n=2 beyond which the optimization cost is too high for a slight improvement in results.

Simple filters, as expected, have good accuracy but not close to the above two approaches. The other algorithms also have the advantage of learning from feedback which simple filters cannot due to their static nature.

4.2.4 User Feedback

Coming to the most critical component of our system, is the user feedback. The user feedback is generated over two iteration cycles. The first iteration cycle begins when the first set of newsletter from all the three algorithms are sent out to the user. they are accompanied by a user feedback form which contains 2 types of questions :- (a) Rating for each of the newsletter on a scale of 1(worst) - 5(best) (b) Feedback associated to each of the newsletter.

Once the user reviews each newsletter and provides feedback, the feedback gets updated in our system. We go through the feedback to find out which newsletter was the one which the user found most relevant and why and at the same time which was the most irrelevant to the user and why.

Based on this survey the system is fine tuned and for the next delivery the reports are generated from the system which has been customised as per the user preference. In the second iteration cycle the email has newsletters in a randomly arranged order which is different from the previous order of the newsletter. This is to nullify any personal preference which the user might have developed based on the order of the previous newsletter.

Again we run through the same set of feedback. In the end results of both the feedback are combined together and the most relevant piece of algorithm is used to send any further reports to the user. A feedback would still be a regular feature of future newsletter for further enhancement and to provide the best user experience.

4.3 Learning Plan

The system was designed to accept feedback from the user and based on that feedback learn about the shortcomings of the algorithm/s. These shortcomings are then improved by plugging in relevant information and extracting the results based on the feedback.

To do this the system first sends a set of newsletter based on the user preference. The first set consists of newsletter from all the three algorithms. Along with the newsletter, a feedback form is also sent which captures rating for each of the newsletter and a suggestion box just to describe the exact feedback. Exact results fetched using two iteration cycles. In the first iteration cycle the user receives newsletters, completes the feedback and sends the results. In the second cycle, algorithms which did not receive a respectable rating were fine tuned.

The modified algorithms again generate three sets of newsletter and they are resent to the user in their next cycle. The same feedback is taken again. Once we have the results from both the feedback cycles we just send the top rated newsletter only to the user going forward. At the same time the feedback would be maintained to fine tune the newsletter delivery.

5. CONCLUSION

After running system tests for the three algorithms we found out that n-grams produced the best set of results. The accuracy of random forest was next best whereas the simple filter produced results which did not give a particular trend. It varied from case to case.

The performance of random forest degrades as the size and amount of data increased. So following the intuitive approach we would go ahead and conclude that n-grams will be our system of choice. The space and time complexity for n-gram does increase exponentially as the data increases but for our current application which is user focused and delivers reports based on static data rather than real time data. Our focus is to send out the most accurate sets of results and that can be achieved by using n-gram.

6. FUTURE WORK

We had a vision when we started with this project but because of the limited scope and time of this project pertaining to the curriculum we had to leave certain parts out of sight. But having said that, those parts are definitely not out of our mind. So as a part of our future work implementations we would try an add the following features to our system :

- Right now the system uses a static data set which might not be up to date and thus provide out dated results. These results do not satisfy the novel idea with which the system was thought of in the first place, though it solves our current purpose. Therefore we would be adding a real time data streaming to the system which will fetch the latest results from Stack Overflow. The user would thus be seeing the most relevant results related to his query.
- The next implementation would be to add OAuth to the system to link the actual user profiles to the system instead of doing a use survey at the start. This will help us monitor the user activity and fetch user personal data from Stack Overflow and we would thus be able to recommend relevant data more correctly apart from the user choice and as per a general system.

7. REFERENCES

[1] academia.edu
[2] https://en.wikipedia.org/wiki/recommender-system
[3] http://stackoverflow.com/research/developer-survey-2015
[4] stackoverflow.com
[5]programmers.stackexchange.com
[6]https://archive.org/details/stackexchange
[7]Using TF-IDF to Determine Word Relevance in Document Queries, Juan Ramos
[8]A Prediction System for Web Requests using N-gram Sequence Models, Zhong Su, Qiang Yang, Hongjiang Zhang