

University of Connecticut
Computer Science and Engineering Department

CSE 5095- 003
Special Topics for Computer Science: Data Mining
Fall 2015

Project Midterm Report

Topic: Stack Overflow Statistical Analysis

Date of Submission: October 20th, 2015

Submitted To:

Dr. Swapna Gokhale

Submitted By:

Areej Althubaity (Areej.Althubaity@uconn.edu)

Malak Alsabban (Malak.Alsabban@uconn.edu)

Misbah Khan (Misbah.Khan@uconn.edu)

Table of Contents

1. Task Given	2
2. Introduction	3
3. Data Description	3
4. Tools Used	5
5. Approach	5
5.1 Initial Setup	5
5.2 Data Collection	6
5.3 Statistical Analysis	6
6. Timeline	6
7. Code and Results	7
8. Challenges Faced	10
9. Conclusion and Future Work	10

1. TASK GIVEN:

Stack Overflow (SO) allows programmers to ask and answer questions related to their work. The site discourages opinion based questions that are likely to generate discussions rather than objective answers. Questions that do not meet the community's criteria will be down voted and may be removed by moderators. Questions are tagged by the asker as being related to at least one and up to five topics. Answers accumulate votes and can also be accepted by the original question asker independently of how many up or down votes it has accumulated. Users gain reputation points when their questions and answers are upvoted. As users gain reputation they gain more and more privileges until they become site moderators.

SO has become an indispensable reference for professional and hobbyist programmers alike. A search for any programming related question will inevitably lead you to a SO post and when it does not, you can post your own question which will likely be answered very quickly by a member of the community. This makes SO a great resource for anyone interested in studying programmers and programming across disciplines. We are interested in analyzing the behavior of the site's users to gain insights into the behavior of programmers in general.

Approach:

1. Accumulate numerical and identifying information across the data set.

For each question you will need to collect the number of answers and comments it received, score, view count, tags, and its asker's reputation. For answers you will need the number of comments it received, score, view count, tags applied to the question it is answering, whether or not it was the accepted answer, and its owner's reputation. For comments you will need its score, the tags of the question it is associated with directly or indirectly through the answer it is associated with, and its owner's reputation.

2. Statistically analyze the data you have accumulated and then discuss your findings.

At a minimum you must perform and discuss the following:

- a. Plot and discuss the distribution of answers and views per question; number of comments per question and answer; the reputation of the owner and the score for questions, comments, and answers; the reputation of the owner and the score for accepted and unaccepted answers separately. Do questions and answers have similar distributions of scores and number of comments? Is there a relationship between a question's score and the number of answers or comments it receives? Does the reputation of users who ask questions, answer questions, or leave comments tend to be significantly different? Compare and discuss the distributions of answers that were accepted and those that were not, is user reputation and score strongly correlated with an answer being accepted?
- b. Aggregate the data by tag and repeat the analysis. Are there certain tags that tend to result in more answers or comments? Are there any that tend to attract users with particularly high or low reputation? Qualitatively discuss any patterns you see in the different tags.

2. INTRODUCTION:

‘Stack Overflow’ is a question and answer site for professional and enthusiast programmers. This project involves the analysis of the stackoverflow.com website statistics, which mainly deals with measures like number of questions, answers, comments, views, upvotes, downvotes, details about users, users’ ratings, etc.

3. DATA DESCRIPTION:

The dataset containing the stack overflow website statistics, mainly consists of 7 tables. Namely, posts, comments, posthistory, badges, postlinks, users and votes. For our analysis in this project, we will mainly be dealing with only 4 of these tables- posts, comments, users and votes.

The table definitions for the four tables which will be used in this project are as follows:

Posts Table Definition:

```
posts:
- Id
- PostTypeId
  - 1: Question
  - 2: Answer
- ParentID (only present if PostTypeId is 2)
- AcceptedAnswerId (only present if PostTypeId is 1)
- CreationDate
- Score
- ViewCount
- Body
- OwnerUserId
- LastEditorUserId
- LastEditorDisplayName="Jeff Atwood"
- LastEditDate="2009-03-05T22:28:34.823"
- LastActivityDate="2009-03-11T12:51:01.480"
- CommunityOwnedDate="2009-03-11T12:51:01.480"
- ClosedDate="2009-03-11T12:51:01.480"
- Title=
- Tags=
- AnswerCount
- CommentCount
- FavoriteCount
```

Comments Table Definition:

```
comments:
- Id
- PostId
- Score
- Text, e.g.: "@Stu Thompson: Seems possible to me -
why not try it?"
- CreationDate, e.g.: "2008-09-06T08:07:10.730"
- UserId
```

Users Table Definition:

```
users:
  - Id
  - Reputation
  - CreationDate
  - DisplayName
  - EmailHash
  - LastAccessDate
  - WebsiteUrl
  - Location
  - Age
  - AboutMe
  - Views
  - UpVotes
  - DownVotes
```

Votes Table Definition:

```
votes:
  - Id
  - PostId
  - VoteTypeId
    - ` 1`: AcceptedByOriginator
    - ` 2`: UpMod
    - ` 3`: DownMod
    - ` 4`: Offensive
    - ` 5`: Favorite - if VoteTypeId = 5 UserId will be
populated
    - ` 6`: Close
    - ` 7`: Reopen
    - ` 8`: BountyStart
    - ` 9`: BountyClose
    - `10`: Deletion
    - `11`: Undeletion
    - `12`: Spam
    - `13`: InformModerator
  - CreationDate
  - UserId (only for VoteTypeId 5)
  - BountyAmount (only for VoteTypeId 9)
```

4. TOOLS USED:

- PostgreSQL-

PostgreSQL is a powerful, open-source object-relational database system. We used PostgreSQL to restore our data dump to a readable format, to verify whether the load was successful or not, to read and understand our data files, and to collect our data into numerical and identifying information across the data set.

- R (via RStudio)-

We chose to perform the statistical and graphical analysis over the collected sample through R programming language. R language is known by most of the scientists and data miners for data analysis. We installed RStudio, which is an open source, and free IDE for R language to help us in performing the statistical analysis.

5. APPROACH:

We decomposed the project into three main tasks, each of which is further divided into subtasks. The three main tasks are as follows:

1. Initial Setup- To set our machines to read and execute our analysis on the dataset.
2. Data Collection- Get familiar with the data, and accumulate numerical and identifying information across the data set.
3. Analysis- Statistically analyze the data we have accumulated and then discuss the findings.

5.1 Initial Setup:

This includes the following tasks:

- Downloading the datafile
- Extracting the datafile
 - Resolving all the issues encountered in the process
- Installing PostgreSQL Server on our systems
 - Resolving all the issues encountered in the process
- Loading Stack Overflow data dump
 - Resolving all the issues encountered in the process
 - Verifying that the load was 100% complete and accurate
 - Execute SQL queries to retrieve specific information.
- Installing R and RStudio
- Connect the database/ data to RStudio.

5.2 Data Collection:

This includes the following tasks:

- Creating table for containing all questions. Each tuple in this table represents a question, and its details. These details include the number of answers and comments it received, score, view count, tags, and its asker's reputation.
- Creating table for containing all answers. Each tuple in this table represents an answer, and its details. These details include the number of comments it received, score, view count, tags applied to the question it is answering, whether or not it was the accepted answer, and its owner's reputation.
- Creating table for containing all comments. Each tuple in this table represents a comment, and its details. These details include its score, the tags of the question it is associated with directly or indirectly through the answer it is associated with, and its owner's reputation.

5.3 Analysis:

This includes exploration like plotting, analyzing and discussing the distribution of various statistical parameters. Finding correlation between parameters, discuss any patterns seen in the different tags, etc. This section will be explored in the coming weeks.

6. TIMELINE:

Our project timeline can be represented as follows:

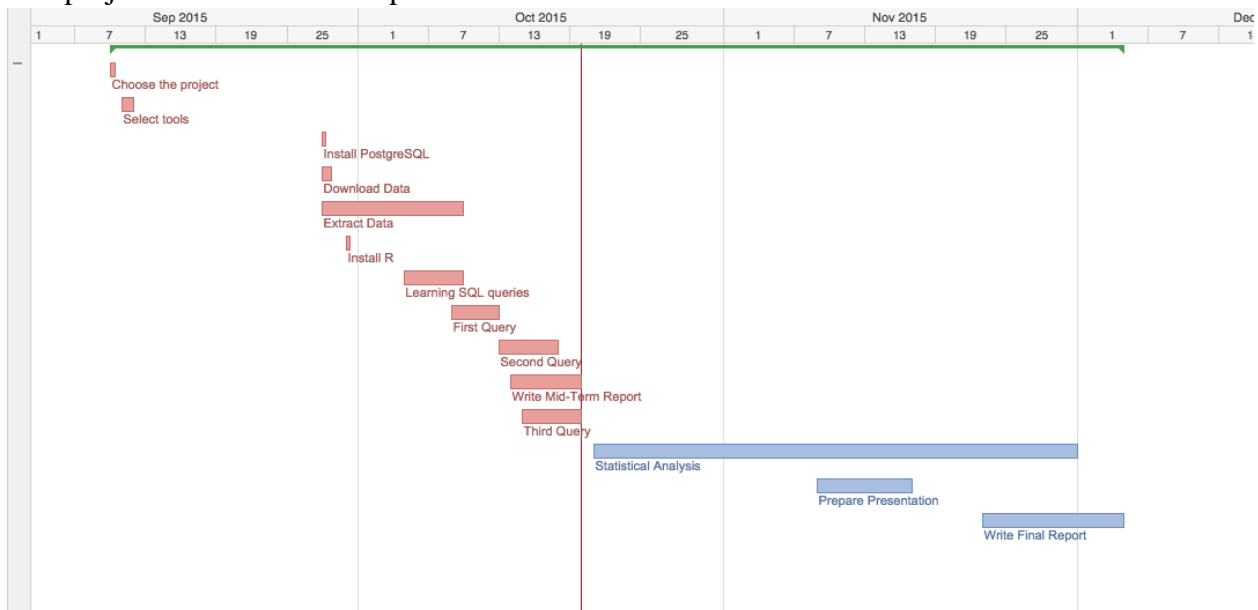


Figure 1: Project Timeline

Our current status and progress is shown in the timeline above. We have successfully completed ‘Phase 1: Initial Setup’ phase and ‘Phase 2: Data Collection’ phase. We are now beginning to work on ‘Phase 3: Statistical Analysis’.

7. CODE AND RESULTS:

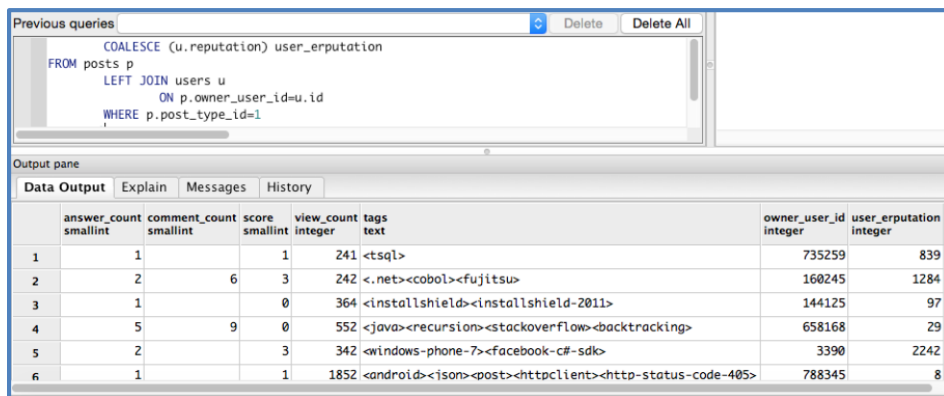
Task 1: Creating table for containing all questions.

First Query: Here for each question, we collect the number of answers and comments it received, score, view count, tags, and its asker’s reputation as in screenshot 1.

SQL Query:

```
SELECT p.answer_count, p.comment_count, p.score, p.view_count, p.tags, p.owner_user_id,
       COALESCE (u.reputation) user_erputation
FROM posts p
LEFT JOIN users u
      ON p.owner_user_id = u.id
WHERE p.post_type_id = 1
```

Results:



The screenshot shows a PostgreSQL console window. The top pane displays the SQL query: `SELECT COALESCE (u.reputation) user_erputation FROM posts p LEFT JOIN users u ON p.owner_user_id=u.id WHERE p.post_type_id=1`. The bottom pane, titled 'Data Output', shows the results of the query in a table with 8 columns: `answer_count`, `comment_count`, `score`, `view_count`, `tags`, `owner_user_id`, and `user_erputation`. The results are as follows:

	answer_count smallint	comment_count smallint	score smallint	view_count integer	tags text	owner_user_id integer	user_erputation integer
1	1		1	241	<tsql>	735259	839
2	2		6	3	<.net><cobol><fujitsu>	160245	1284
3	1		0	364	<installshield><installshield-2011>	144125	97
4	5		9	0	<java><recursion><stackoverflow><backtracking>	658168	29
5	2		3	342	<windows-phone-7><facebook-c#-sdk>	3390	2242
6	1		1	1852	<android><json><post><httpClient><http-status-code-405>	788345	8

Screenshot 1: Results of first query in PostgreSQL console

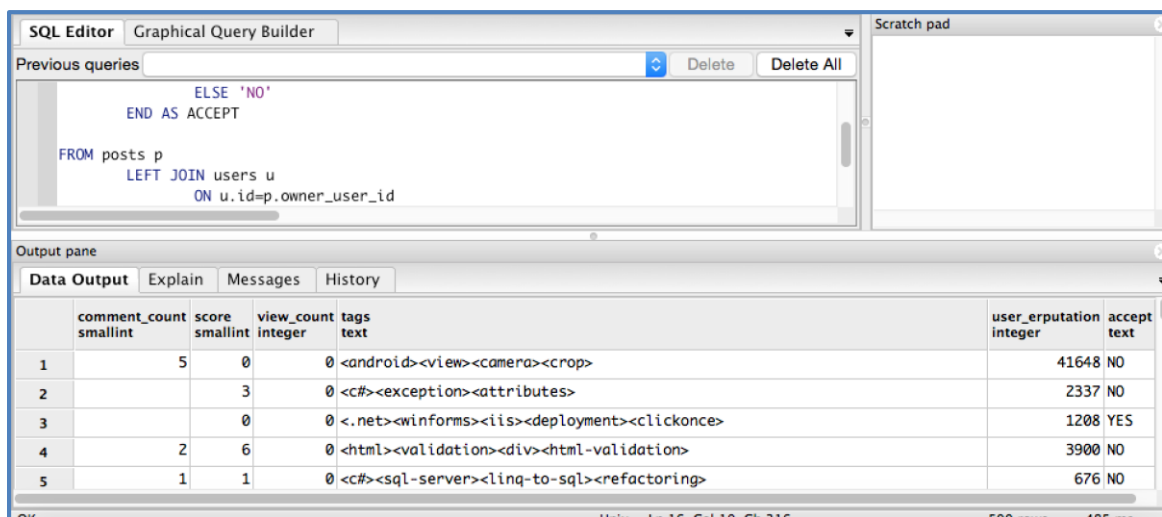
Task 2: Creating table for containing all answers.

Second Query: For each answer, we collect the number of comments it received, score, view count, tags applied to the question it is answering, whether or not it was the accepted answer, and its owner's reputation as in screenshot 2.

SQL Query:

```
SELECT p.comment_count, p.score, p.view_count, q.tags,
       COALESCE (u.reputation) user_erputation,
       CASE q.accepted_answer_id
         WHEN p.id THEN 'YES'
         ELSE 'NO'
       END AS ACCEPT
FROM posts p
LEFT JOIN users u
  ON u.id=p.owner_user_id
LEFT JOIN posts q
  ON p.parent_id = q.id
WHERE p.post_type_id = 2
```

Results:



The screenshot shows the PostgreSQL console interface. The top section is the 'SQL Editor' with tabs for 'SQL Editor' and 'Graphical Query Builder'. Below it is the 'Previous queries' section showing the executed query. The bottom section is the 'Output pane' with tabs for 'Data Output', 'Explain', 'Messages', and 'History'. The 'Data Output' tab is active, displaying the results of the query in a table format.

	comment_count smallint	score smallint	view_count integer	tags text	user_erputation integer	accept text
1	5	0	0	<android><view><camera><crop>	41648	NO
2		3	0	<c#><exception><attributes>	2337	NO
3		0	0	<.net><winforms><iis><deployment><clickonce>	1208	YES
4	2	6	0	<html><validation><div><html-validation>	3900	NO
5	1	1	0	<c#><sql-server><linq-to-sql><refactoring>	676	NO

Screenshot 2: Second query results in PostgreSQL console

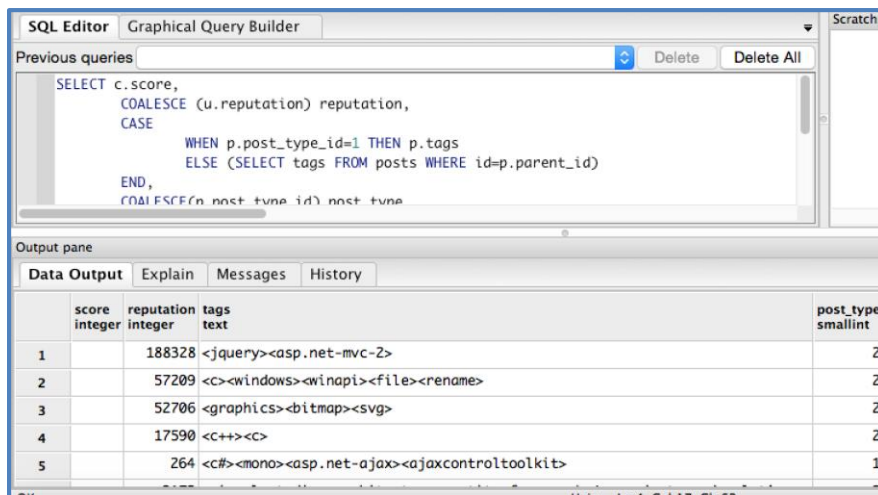
Task 3: Creating table for containing all comments.

Third Query: For each comment we collect its score, the tags of the question it is associated with directly or indirectly through the answer it is associated with, and its owner's reputation.

SQL Query:

```
SELECT c.score,
       COALESCE (u.reputation) reputation,
       CASE
         WHEN p.post_type_id=1 THEN p.tags
         ELSE (SELECT tags FROM posts WHERE id=p.parent_id)
       END,
       COALESCE(p.post_type_id) post_type
FROM comments c
LEFT JOIN users u
  ON c.user_id=u.id
LEFT JOIN posts p
  ON (c.post_id=p.id)
```

Results:



The screenshot shows a PostgreSQL console window with the SQL query from the previous block entered in the 'SQL Editor' tab. The 'Output pane' is active, showing the 'Data Output' tab with a table of results. The table has five columns: an index, 'score' (integer), 'reputation' (integer), 'tags' (text), and 'post_type' (smallint). There are five rows of data.

	score integer	reputation integer	tags text	post_type smallint
1		188328	<jquery><asp.net-mvc-2>	2
2		57209	<c><windows><winapi><file><rename>	2
3		52706	<graphics><bitmap><svg>	2
4		17590	<c++><c>	2
5		264	<c#><mono><asp.net-ajax><ajaxcontroltoolkit>	1

Screenshot 3: Results of third query in PostgreSQL console

8. CHALLENGES FACED:

There were many challenges we faced in our entire process till date. Some of them are listed as follows:

- The data download link was not available on the day when the entire team decided to meet for an entire day to work over the project. We tried our best to get the data from other sources, but couldn't succeed.
- Once the download link became available, we had problems with decompressing the compressed data files. The files either could not be decompressed at all, or were decompressed partially or with errors. We tried a wide variety of decompression software, with the ability to decompress huge data files (12+ GB). We finally used *Archive Utility* for MAC OS and *7-Zip* for Windows.
- While installing PostgreSQL, we faced many unstated errors (which were recognized only while we were working with the actual data dump). We had to uninstall and reinstall the PostgreSQL database server innumerable times for successful functioning of the software and data dump restore.
- Restoring the data dump to a template database wasn't an easy task. We tried almost every psql command (such as `pg_restore`) and SQL query to retrieve the data. Some of them threw errors, while others did not throw any errors but did not restore the data. Finally, we were successfully able to get the data live into the database server through psql with the following command "`\i /path/file.dump`".

9. CONCLUSION AND FUTURE WORK:

We have successfully completed two tasks of the three tasks (as mentioned in the 'Approach' section of this document). We need to work on the third main task, dealing with performing statistical analysis on the collected data.