STACK OVERFLOW - Query Mining

Nikita Jituri
Department of Computer Science,
University of Maryland,
Baltimore County.
njituri1@umbc.edu

Phanindra Kumar Kannaji
Department of Computer Science,
University of Maryland,
Baltimore County.
pkanna1@umbc.edu

Sneha Tatipally Venkat
Department of Computer Science,
University of Maryland,
Baltimore County.
sneha6@umbc.edu

***Abstract*— Tagging lets users explore related content, and is very useful on question answers sites like Stack Overflow, Quora and other Stack Exchange sites etc. In this paper, we present a system that is able to automatically identify tags to questions from the question-answering site Stack Overflow and also shows the statistics such as number of questions related to a tag, total number of answers and many more stats related to a tag.**

Keywords- Tags, Pig, SQLite, MapReduce.

I. INTRODUCTION

Question-answer sites work on a simple premise: that any user can pose a question, and in turn other users – potentially many of them – will provide an answer. Tens of thousands of questions are asked and answered every day on question and answer (Q&A) websites such as StackOverflow. StackOverflow is the most popular site available today for software developers to post technical questions and get high quality answers from the community. StackOverflow uses tags to suggest users related questions. More than six thousand new questions is asked on StackOverflow every weekday. The main motivation is to implement complex SQL queries in Pig scripts and to provide an interface for the user to get information about the technologies the user is working on.

In this project we predict tags based on the content of a question. More formally, given a question q consisting of n words $a_1,..., a_n$ we want to assign $1 \leq k \leq 5$ tags $t_1,..., t_k$ from a limited list of tags T. After identifying the tags we also show the statistics per tag per year basis such as the total number of questions related to that tag, total number of answers, total number of accepted answers, total number of closed questions, total number of open questions, total number of active users and chance of getting the answer to the question. We will also show the top five users and also top five questions related to that tags. We also show the trends of the tag from 2008 to 2015.

Stack Overflow Dataset

Amongst the four V's of big data, the StackOverflow dataset consists of Volume and veracity. The dataset we used was from StackOverflow consists of all questions answers posted on stackoverflow.com from 2008-2015. It also consists the tags, users list and badges. The questions are all related to computer programming and can be posted by anyone with an account, which is free to create. Each question is tagged by the author with the tags that are most representative of the post. Example tags are programming languages such as PHP, Common-Lisp, MySQL etc. and also general topics such as databases, optimization, arrays etc.

The dataset is 46GB. The dataset consists of 11 million questions, 17 million answers, 42 million comments, 38 million tags and 3.4 million users. The sample dataset is as below.

	A	B	C	D	E	G	H	I	J	K	L	M	N	O	P
1	PostId	PostCreationDate	OwnerUserId	OwnerCreationDate	ReputationAtPostCreation	Title	BodyMarkdown	Tag1	Tag2	Tag3	Tag4	Tag5	PostClosedDate	OpenStatus	
2	10035653	4/5/2012 20:37	1159226	1/19/2012 18:46	1	what is the best u	I know this question can be	c++					4/5/2012 23:31	too localized	
3	8922537	1/19/2012 7:38	1157921	1/19/2012 7:31	1	upload a zip file ≤	I make a up-loader to `upload`	php	xml	cakep zip			1/19/2012 16:43	not a real ques	
4	5962216	5/11/2011 9:43	696219	4/7/2011 6:38	40	UIButton Backgro	Can anyone tell me how to	iphone	sdk-4.0					open	
5	10070625	4/9/2012 8:16	490895	10/29/2010 4:59	1	using the pt_regs f	I've recently started getting	linux	modu	kernel				open	
6	8960935	1/22/2012 12:10	1017103	10/27/2011 18:26	28	Can I use LIMIT fu	Can I use LIMIT function in	mysql	limit					open	
7	11760939	8/1/2012 13:59	504617	11/11/2010 14:25	50	Ubuntu: Compiz,	I've fresh installed Ubuntu	ubunt	nautil	compiz			8/2/2012 14:06	off topic	
8	12667304	10/1/2012 3:47	1693381	9/24/2012 3:38	10	Remove all contr	I'm building a flow layout	c#	flowlayoutpanel					open	
9	8524835	12/15/2011 18:26	1100507	12/15/2011 18:18	1	After disable Uni	I disable Unity in Ubuntu 11.1	ubunt	unity	panels			##########	off topic	
10	10454249	5/4/2012 18:29	528211	12/2/2010 15:51	2422	When does finali	I have an app that recently	delph	delphi-2010					open	
11	11853208	8/7/2012 20:01	1553248	7/26/2012 1:38	38	char* to const wc	I need to convert `character`	c++	pointe	char	wchar			open	
12	11933076	8/13/2012 11:06	1545519	7/23/2012 10:02	23	my jquery plugin	I have my own jquery plugin	jquer	jquer	jquer	jquery-selectors			open	
13	4709933	1/17/2011 3:58	504845	11/11/2010 17:37	191	What Ruby blog e	What blog engines written in	ruby	blogs	blog-engine			9/30/2012 13:31	not constructiv	
14	5029559	2/17/2011 13:29	633799	2/17/2011 12:47	1	Crystal report: Ur	This is the code of Crystal	asp.n	sql	serve	crystal-reports			open	
15	8737926	1/5/2012 5:04	1131392	1/5/2012 5:00	1	support for iptab	Am a new comer in linux	linux	cento	iptabl	portforwar		1/5/2012 20:25	off topic	
16	10394417	5/1/2012 6:35	1066909	11/26/2011 13:12	20	Shooting compet	I'm going to write an	c++	.net	opencv			5/1/2012 18:43	not constructiv	
17	11569888	7/19/2012 21:52	903027	8/19/2011 19:36	21	Populate JPlayer	I'm trying to create a	jquer	xml	jplaye	playlist			open	
18	7603756	9/29/2011 21:44	972023	9/29/2011 21:31	1	FB SSL requireme	I have created app html	faceb	ssl				10/2/2011 10:57	too localized	
19	11233728	6/27/2012 19:38	977466	10/3/2011 21:28	140	Iphone App Relea	Me and My friend had an app	iphon	ios	app-store			6/27/2012 21:35	off topic	
20	35488	8/30/2008 0:05	572	8/6/2008 20:56	2192	Is there a good tc	The one that comes with Win	windc	path-variables				9/9/2012 17:39	off topic	
21	4784166	1/24/2011 16:08	554038	12/25/2010 23:36	25	Learning Discrete	I'm currently taking a	colleg	discre	textbook			1/24/2011 16:55	off topic	
22	2400828	3/8/2010 11:12	238052	12/24/2009 5:14	149	inner class withir	is that possible to create a	interf	java	inner-oops				open	
23	6014534	5/16/2011 7:56	624429	2/19/2011 14:36	110	What is these lin	`T FindBy(object key);`	c#	.net	vb.net				open	
24	7562498	9/26/2011 23:24	277329	2/19/2010 21:15	11	Suggestions for c	I am planning to create a	php	pytho	conte	imdb		9/26/2011 23:52	not a real ques	
25	8169136	11/17/2011 14:51	840576	7/12/2011 11:18	1	Google Map Mark	I need cluster map marker	google-maps					##########	not a real ques	

Figure 1: Sample dataset

The datasets is XML format. It consists of 8 different XML files such as badges, comments, posts, posts history, users, tags and votes.

II. RELATED WORK

Numerous works have been done previously on tags identification. All those works are related on how to find the

tags in the question posted by doing semantic analysis or some other kind of analysis. One of the works includes the work done by Sebastian Schuster, Wanying Zhu, Yiying Cheng. There system consists of a programming language detection system and a SVM using content-based features. [1] The other work includes the work done by Clayton Stanley and Michael D. Byrne for predicting tags. They developed an ACT-R inspired Bayesian probabilistic model that can predict the hashtags used by the author of the post. [5]

III. ARCHITECTURE

First the raw dataset is uploaded into HDFS. Then we wrote Pig scripts to do the map reduce jobs. PigLatin is a Data Analytical language used to create Map-Reduce jobs to run on large datasets. We choose Apache Pig because it is more concise that general map-reduce. A 200 lines Java code written for MapReduce can be reduced to 10 lines of Pig code. The results after doing the map reduce job are fed into the database. We have used SQLite over MySQL as database. The main reason to choose SQLite is that Django framework is more compatible with SQLite. Django is a free and open source web application framework, written in Python, which follows the model–view–controller (MVC) architectural pattern. Its primary goal is to ease the creation of complex, database-driven websites. It also includes the ability to launch a FastCGI server, enabling use behind any web server which supports FastCGI. [3]

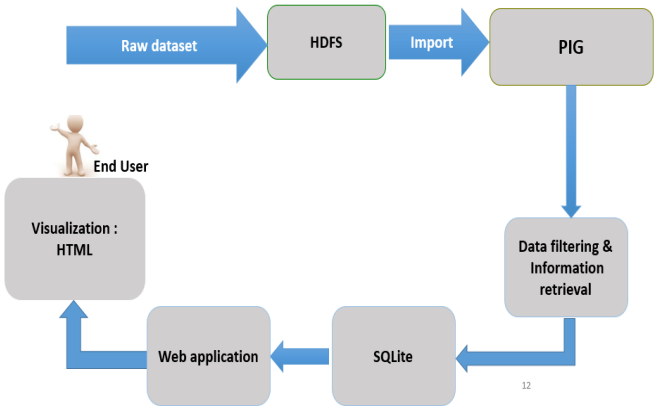

Figure 2: Architecture design of our project.

When there are more number of users who want to access the page then one prefer other RDBMS databases over SQLite. Since the project was a demo version we preferred SQLite. The data was processed and stored there and sent to the web application part as on when requested by the server about the data.

We used bootstrap for designing the web application since it is a free and open-source collection of tools. It contains HTML and CSS based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. It aims to ease the development of dynamic websites and web applications. [4]

We have used AJAX for creating web applications. AJAX stands for Asynchronous JavaScript and XML. It is a new technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS, and Java Script. [12]

We have used google charts for showing the trends of the statistics. Google charts are used in a web-based development and instructional environment that allows people to generate a chart through some data as input. This study has the implications for the creation of charts and other visual statistical content in a web-based environment. It actually creates a PNG image of the chart which helps developers as part of their enhancement strategies of their website. It supports a variety of charts that people can include in their web pages.

IV. METHOD

First the XML is uploaded into the HDFS. Once the data is uploaded we ran the each of the Pig scripts to do the map-reduce jobs. Each Pig script does 4-9 MapReduce jobs in a single run. The analysis part is done as shown in the figure below.

- Example: TotalNumberofActiveUsers(tag,year)
- Description: Count total number of active users for a given tag per year:
- Input values: Post(year, tag1, tag2, tag3, tag4, tag5, ownerUserId);

Figure 3: Analysis of the dataset

In the above example we are using Posts.xml file from the dataset. The final result will be the total number of active users for the given tag in that year. The grouping is done on the basis of tag and year. The inputs will be tag, year, owner user Id. We will calculate distinct owner user ID's so as to know the total number of users in order to avoid repetitions.

First Pig program which we have written gives us the following results. They are:

- Total number of related questions
- Total number of answers
- Total number of accepted answers
- Total score
- Total number of deleted questions
- Total number of closed questions
- Chance of getting an answer

The above results are for individual tag per year. The grouping in this script is done based on tag and year. The first Pig script is as below:

```
DEFINE XPath org.apache.pig.piggybank.evaluation.xml.XPath();
POSTS = LOAD '/user/sneha6/data/Posts' using org.apache.pig.piggybank.storage.XMLLoader('row') as (x:chararray);
QUES1 = FOREACH POSTS GENERATE
FLATTEN(REGEX_EXTRACT_ALL(x,'<row\\s*Id=".*?"\\s*PostTypeId="1"\\s*(AcceptedAnswerId=".*?"\\s*)?CreationDate="(....).*?"\\s*(DeletionDate=".*?"\\s*)?Score="(.*)"\\s*ViewCount=".*?"\\s*.*Tags="&lt;([^&]*)&gt;"\\s*AnswerCount="(.*)"\\s*CommentCount=".*?"\\s*(FavoriteCount=".*?"\\s*)?(ClosedDate=".*?"\\s*)?(CommunityOwnedDate=".*?"\\s*)?\\/>')) AS ( accAnsId:chararray, cYear:chararray, delDate:chararray, score:chararray, tag1:chararray,
ansCount:chararray,favCount:chararray , closedDate:chararray, commDate:chararray);
QUES2 = FOREACH POSTS GENERATE
FLATTEN(REGEX_EXTRACT_ALL(x,'<row\\s*Id=".*?"\\s*PostTypeId="1"\\s*(AcceptedAnswerId=".*?"\\s*)?CreationDate="(....).*?"\\s*(DeletionDate=".*?"\\s*)?Score="(.*)"\\s*ViewCount=".*?"\\s*.*Tags="&lt;([^&]*)&gt;&lt;([^&]*)&gt;"\\s*AnswerCount="(.*)"\\s*CommentCount=".*?"\\s*(FavoriteCount=".*?"\\s*)?(ClosedDate=".*?"\\s*)?(CommunityOwnedDate=".*?"\\s*)?\\/>')) AS ( accAnsId:chararray, cYear:chararray, delDate:chararray, score:chararray,
tag1:chararray, tag2:chararray, ansCount:chararray,favCount:chararray , closedDate:chararray, commDate:chararray);
QUES3 = FOREACH POSTS GENERATE
FLATTEN(REGEX_EXTRACT_ALL(x,'<row\\s*Id=".*?"\\s*PostTypeId="1"\\s*(AcceptedAnswerId=".*?"\\s*)?CreationDate="(....).*?"\\s*(DeletionDate=".*?"\\s*)?Score="(.*)"\\s*ViewCount=".*?"\\s*.*Tags="&lt;([^&]*)&gt;&lt;([^&]*)&gt;&lt;([^&]*)&gt;"\\s*AnswerCount="(.*)"\\s*CommentCount=".*?"\\s*(FavoriteCount=".*?"\\s*)?(ClosedDate=".*?"\\s*)?(CommunityOwnedDate=".*?"\\s*)?\\/>')) AS ( accAnsId:chararray, cYear:chararray, delDate:chararray,
score:chararray, tag1:chararray, tag2:chararray, tag3:chararray, ansCount:chararray,favCount:chararray , closedDate:chararray, commDate:chararray);
QUES4 = FOREACH POSTS GENERATE
FLATTEN(REGEX_EXTRACT_ALL(x,'<row\\s*Id=".*?"\\s*PostTypeId="1"\\s*(AcceptedAnswerId=".*?"\\s*)?CreationDate="(....).*?"\\s*(DeletionDate=".*?"\\s*)?Score="(.*)"\\s*ViewCount=".*?"\\s*.*Tags="&lt;([^&]*)&gt;&lt;([^&]*)&gt;&lt;([^&]*)&gt;&lt;([^&]*)&gt;"\\s*AnswerCount="(.*)"\\s*CommentCount=".*?"\\s*(FavoriteCount=".*?"\\s*)?(ClosedDate=".*?"\\s*)?(CommunityOwnedDate=".*?"\\s*)?\\/>')) AS ( accAnsId:chararray, cYear:chararray,
delDate:chararray, score:chararray, tag1:chararray, tag2:chararray,tag3:chararray,tag4:chararray, ansCount:chararray,favCount:chararray ,
closedDate:chararray, commDate:chararray);
QUES5 = FOREACH POSTS GENERATE
FLATTEN(REGEX_EXTRACT_ALL(x,'<row\\s*Id=".*?"\\s*PostTypeId="1"\\s*(AcceptedAnswerId=".*?"\\s*)?CreationDate="(....).*?"\\s*(DeletionDate=".*?"\\s*)?Score="(.*)"\\s*ViewCount=".*?"\\s*.*Tags="&lt;([^&]*)&gt;&lt;([^&]*)&gt;&lt;([^&]*)&gt;&lt;([^&]*)&gt;&lt;([^&]*)&gt;"\\s*AnswerCount="(.*)"\\s*Comm
entCount=".*?"\\s*(FavoriteCount=".*?"\\s*)?(ClosedDate=".*?"\\s*)?(CommunityOwnedDate=".*?"\\s*)?\\/>')) AS ( accAnsId:chararray,
cYear:chararray, delDate:chararray, score:chararray, tag1:chararray, tag2:chararray,tag3:chararray,tag4:chararray,tag5:chararray,
ansCount:chararray,favCount:chararray , closedDate:chararray, commDate:chararray);
A11 = FOREACH QUES1 GENERATE (INT) ((accAnsId is NULL)?0:1) AS accAnsFlag, (INT) cYear, (INT) ((delDate is NULL)?0:1) AS delDateFlag, (INT) score,
tag1 as tag, (INT) ansCount, (INT) ((closedDate is NULL)?0:1) AS closedDateFlag;
A21 = FOREACH QUES2 GENERATE (INT) ((accAnsId is NULL)?0:1) AS accAnsFlag, (INT) cYear, (INT) ((delDate is NULL)?0:1) AS delDateFlag, (INT) score,
tag1 as tag, (INT) ansCount, (INT) ((closedDate is NULL)?0:1) AS closedDateFlag;
A22 = FOREACH QUES2 GENERATE (INT) ((accAnsId is NULL)?0:1) AS accAnsFlag, (INT) cYear, (INT) ((delDate is NULL)?0:1) AS delDateFlag, (INT) score,
tag2 as tag, (INT) ansCount, (INT) ((closedDate is NULL)?0:1) AS closedDateFlag;
A31 = FOREACH QUES3 GENERATE (INT) ((accAnsId is NULL)?0:1) AS accAnsFlag, (INT) cYear, (INT) ((delDate is NULL)?0:1) AS delDateFlag, (INT) score,
tag1 as tag, (INT) ansCount, (INT) ((closedDate is NULL)?0:1) AS closedDateFlag;
A32 = FOREACH QUES3 GENERATE (INT) ((accAnsId is NULL)?0:1) AS accAnsFlag, (INT) cYear, (INT) ((delDate is NULL)?0:1) AS delDateFlag, (INT) score,
tag2 as tag, (INT) ansCount, (INT) ((closedDate is NULL)?0:1) AS closedDateFlag;
A33 = FOREACH QUES3 GENERATE (INT) ((accAnsId is NULL)?0:1) AS accAnsFlag, (INT) cYear, (INT) ((delDate is NULL)?0:1) AS delDateFlag, (INT) score,
```

Figure 4: First Pig script

```
A33 = FOREACH QUES3 GENERATE (INT) ((accAnsId is NULL)?0:1) AS accAnsFlag, (INT) cYear, (INT) ((delDate is NULL)?0:1) AS delDateFlag, (INT) score,
tag3 as tag, (INT) ansCount, (INT) ((closedDate is NULL)?0:1) AS closedDateFlag;
A41 = FOREACH QUES4 GENERATE (INT) ((accAnsId is NULL)?0:1) AS accAnsFlag, (INT) cYear, (INT) ((delDate is NULL)?0:1) AS delDateFlag, (INT) score,
tag1 as tag, (INT) ansCount, (INT) ((closedDate is NULL)?0:1) AS closedDateFlag;
A42 = FOREACH QUES4 GENERATE (INT) ((accAnsId is NULL)?0:1) AS accAnsFlag, (INT) cYear, (INT) ((delDate is NULL)?0:1) AS delDateFlag, (INT) score,
tag2 as tag, (INT) ansCount, (INT) ((closedDate is NULL)?0:1) AS closedDateFlag;
A43 = FOREACH QUES4 GENERATE (INT) ((accAnsId is NULL)?0:1) AS accAnsFlag, (INT) cYear, (INT) ((delDate is NULL)?0:1) AS delDateFlag, (INT) score,
tag3 as tag, (INT) ansCount, (INT) ((closedDate is NULL)?0:1) AS closedDateFlag;
A44 = FOREACH QUES4 GENERATE (INT) ((accAnsId is NULL)?0:1) AS accAnsFlag, (INT) cYear, (INT) ((delDate is NULL)?0:1) AS delDateFlag, (INT) score,
tag4 as tag, (INT) ansCount, (INT) ((closedDate is NULL)?0:1) AS closedDateFlag;
A51 = FOREACH QUES5 GENERATE (INT) ((accAnsId is NULL)?0:1) AS accAnsFlag, (INT) cYear, (INT) ((delDate is NULL)?0:1) AS delDateFlag, (INT) score,
tag1 as tag, (INT) ansCount, (INT) ((closedDate is NULL)?0:1) AS closedDateFlag;
A52 = FOREACH QUES5 GENERATE (INT) ((accAnsId is NULL)?0:1) AS accAnsFlag, (INT) cYear, (INT) ((delDate is NULL)?0:1) AS delDateFlag, (INT) score,
tag2 as tag, (INT) ansCount, (INT) ((closedDate is NULL)?0:1) AS closedDateFlag;
A53 = FOREACH QUES5 GENERATE (INT) ((accAnsId is NULL)?0:1) AS accAnsFlag, (INT) cYear, (INT) ((delDate is NULL)?0:1) AS delDateFlag, (INT) score,
tag3 as tag, (INT) ansCount, (INT) ((closedDate is NULL)?0:1) AS closedDateFlag;
A54 = FOREACH QUES5 GENERATE (INT) ((accAnsId is NULL)?0:1) AS accAnsFlag, (INT) cYear, (INT) ((delDate is NULL)?0:1) AS delDateFlag, (INT) score,
tag4 as tag, (INT) ansCount, (INT) ((closedDate is NULL)?0:1) AS closedDateFlag;
A55 = FOREACH QUES5 GENERATE (INT) ((accAnsId is NULL)?0:1) AS accAnsFlag, (INT) cYear, (INT) ((delDate is NULL)?0:1) AS delDateFlag, (INT) score,
tag5 as tag, (INT) ansCount, (INT) ((closedDate is NULL)?0:1) AS closedDateFlag;
U1  = UNION ONSCHEMA A11, A21;
U2  = UNION ONSCHEMA U1, A22;
U3  = UNION ONSCHEMA U2, A31;
U4  = UNION ONSCHEMA U3 , A32;
U5  = UNION ONSCHEMA U4 , A33;
U6  = UNION ONSCHEMA U5 , A41;
U7  = UNION ONSCHEMA U6 , A42;
U8  = UNION ONSCHEMA U7 , A43;
U9  = UNION ONSCHEMA U8 , A44;
U10 = UNION ONSCHEMA U9 , A51;
U11 = UNION ONSCHEMA U10, A52;
U12 = UNION ONSCHEMA U11, A53;
U13 = UNION ONSCHEMA U12, A54;
U14 = UNION ONSCHEMA U13, A55;
B = GROUP U14 BY (tag,cYear);
C = FOREACH B GENERATE FLATTEN(group) as (tag, cYear), COUNT(U14.cYear) as qCount,SUM(U14.accAnsFlag) AS accAnsCount, SUM(U14.delDateFlag) AS
delQuesCount, SUM(U14.score) AS scoreTotal, SUM(U14.ansCount) AS answerCount, SUM(U14.closedDateFlag) AS closedQCount;
D = FILTER C BY tag is not null;
STORE D INTO '/users/sneha6/data/output/postsoutput1';
```

Figure 5: continuation of First Pig script

The above Pig script has taken 1 hour and 30 minutes to process the data, do the appropriate map reduce job and finally provide with the desired results.

The second Pig script which we have written gives the total number active users present on that tag per year i.e. it gives us the information about total number of active users for that tag provided on per year basis. The second Pig script is as below:

```
DEFINE XPath org.apache.pig.piggybank.evaluation.xml.XPath();
POSTS = LOAD '/user/sneha6/data/Posts1' using org.apache.pig.piggybank.storage.XMLLoader('row') as (x:chararray);
QUES1 = FOREACH POSTS GENERATE
FLATTEN(REGEX_EXTRACT_ALL(x,'<row\\s*Id=".*?"\\s*PostTypeId="1"\\s*.*CreationDate="(....).*?"\\s*.*Tags="&lt;([^&]*)&gt;"\\s*.*\\/>')) AS (
id:chararray, cYear:chararray, tag1:chararray);
QUES2 = FOREACH POSTS GENERATE
FLATTEN(REGEX_EXTRACT_ALL(x,'<row\\s*Id=".*?"\\s*PostTypeId="1"\\s*.*CreationDate="(....).*?"\\s*.*Tags="&lt;([^&]*)&gt;&lt;([^&]*)&gt;"\\s*.*?\\/>'
)) AS ( id:chararray, cYear:chararray, tag1:chararray, tag2:chararray);
QUES3 = FOREACH POSTS GENERATE
FLATTEN(REGEX_EXTRACT_ALL(x,'<row\\s*Id=".*?"\\s*PostTypeId="1"\\s*.*CreationDate="(....).*?"\\s*.*Tags="&lt;([^&]*)&gt;&lt;([^&]*)&gt;&lt;([^&]*)&
gt;"\\s*.*\\/>')) AS ( id:chararray, cYear:chararray, tag1:chararray, tag2:chararray, tag3:chararray);
QUES4 = FOREACH POSTS GENERATE
FLATTEN(REGEX_EXTRACT_ALL(x,'<row\\s*Id=".*?"\\s*PostTypeId="1"\\s*.*CreationDate="(....).*?"\\s*.*Tags="&lt;([^&]*)&gt;&lt;([^&]*)&gt;&lt;([^&]*)&g
t;&lt;([^&]*)&gt;"\\s*.*\\/>')) AS ( id:chararray, cYear:chararray, tag1:chararray, tag2:chararray,tag3:chararray,tag4:chararray);
QUES5 = FOREACH POSTS GENERATE
FLATTEN(REGEX_EXTRACT_ALL(x,'<row\\s*Id=".*?"\\s*PostTypeId="1"\\s*.*CreationDate="(....).*?"\\s*.*Tags="&lt;([^&]*)&gt;&lt;([^&]*)&gt;&lt;([^&]*)&g
t;&lt;([^&]*)&gt;&lt;([^&]*)&gt;"\\s*.*\\/>')) AS ( id:chararray, cYear:chararray, tag1:chararray, tag2:chararray,tag3:chararray,tag4:chararray,
tag5:chararray);
A11 = FOREACH QUES1 GENERATE (INT) id, (INT) cYear, tag1 as tag;
A21 = FOREACH QUES2 GENERATE (INT) id, (INT) cYear, tag1 as tag;
A22 = FOREACH QUES2 GENERATE (INT) id, (INT) cYear, tag2 as tag;
A31 = FOREACH QUES3 GENERATE (INT) id, (INT) cYear, tag1 as tag;
A32 = FOREACH QUES3 GENERATE (INT) id, (INT) cYear, tag2 as tag;
A33 = FOREACH QUES3 GENERATE (INT) id, (INT) cYear, tag3 as tag;
A41 = FOREACH QUES4 GENERATE (INT) id, (INT) cYear, tag1 as tag;
A42 = FOREACH QUES4 GENERATE (INT) id, (INT) cYear, tag2 as tag;
A43 = FOREACH QUES4 GENERATE (INT) id, (INT) cYear, tag3 as tag;
A44 = FOREACH QUES4 GENERATE (INT) id, (INT) cYear, tag4 as tag;
A51 = FOREACH QUES5 GENERATE (INT) id, (INT) cYear, tag1 as tag;
A52 = FOREACH QUES5 GENERATE (INT) id, (INT) cYear, tag2 as tag;
A53 = FOREACH QUES5 GENERATE (INT) id, (INT) cYear, tag3 as tag;
A54 = FOREACH QUES5 GENERATE (INT) id, (INT) cYear, tag4 as tag;
A55 = FOREACH QUES5 GENERATE (INT) id, (INT) cYear, tag5 as tag;
U1  = UNION ONSCHEMA A11, A21;
U2  = UNION ONSCHEMA U1, A22;
U3  = UNION ONSCHEMA U2, A31;
U4  = UNION ONSCHEMA U3 , A32;
U5  = UNION ONSCHEMA U4 , A33;
U6  = UNION ONSCHEMA U5 , A41;
U7  = UNION ONSCHEMA U6 , A42;
U8  = UNION ONSCHEMA U7 , A43;
U9  = UNION ONSCHEMA U8 , A44;
U10 = UNION ONSCHEMA U9 , A51;
U11 = UNION ONSCHEMA U10, A52;
U12 = UNION ONSCHEMA U11, A53;
U13 = UNION ONSCHEMA U12, A54;
U14 = UNION ONSCHEMA U13, A55;
```

Figure 6: Second Pig script

```
A32 = FOREACH QUES3 GENERATE (INT) id, (INT) cYear, tag2 as tag;
A33 = FOREACH QUES3 GENERATE (INT) id, (INT) cYear, tag3 as tag;
A41 = FOREACH QUES4 GENERATE (INT) id, (INT) cYear, tag1 as tag;
A42 = FOREACH QUES4 GENERATE (INT) id, (INT) cYear, tag2 as tag;
A43 = FOREACH QUES4 GENERATE (INT) id, (INT) cYear, tag3 as tag;
A44 = FOREACH QUES4 GENERATE (INT) id, (INT) cYear, tag4 as tag;
A51 = FOREACH QUES5 GENERATE (INT) id, (INT) cYear, tag1 as tag;
A52 = FOREACH QUES5 GENERATE (INT) id, (INT) cYear, tag2 as tag;
A53 = FOREACH QUES5 GENERATE (INT) id, (INT) cYear, tag3 as tag;
A54 = FOREACH QUES5 GENERATE (INT) id, (INT) cYear, tag4 as tag;
A55 = FOREACH QUES5 GENERATE (INT) id, (INT) cYear, tag5 as tag;
U1  = UNION ONSCHEMA A11, A21;
U2  = UNION ONSCHEMA U1, A22;
U3  = UNION ONSCHEMA U2, A31;
U4  = UNION ONSCHEMA U3 , A32;
U5  = UNION ONSCHEMA U4 , A33;
U6  = UNION ONSCHEMA U5 , A41;
U7  = UNION ONSCHEMA U6 , A42;
U8  = UNION ONSCHEMA U7 , A43;
U9  = UNION ONSCHEMA U8 , A44;
U10 = UNION ONSCHEMA U9 , A51;
U11 = UNION ONSCHEMA U10, A52;
U12 = UNION ONSCHEMA U11, A53;
U13 = UNION ONSCHEMA U12, A54;
U14 = UNION ONSCHEMA U13, A55;
ANSW = FOREACH POSTS GENERATE
FLATTEN(REGEX_EXTRACT_ALL(x,'<row\\s*Id=".*"\\s*PostTypeId="2"\\s*ParentId="(.*?)"\\s*.*OwnerUserId="(.*?)"\\s*.*\\/>')) AS (parentId:chararray,
ownerUserId:chararray);
B11 = FOREACH ANSW GENERATE (INT) parentId, (INT) ownerUserId;
B = JOIN U14 BY id FULL OUTER, B11 BY parentId;
C = GROUP B BY (tag,cYear);
D = FOREACH C GENERATE FLATTEN(group) AS (tag,cYear), COUNT(B.ownerUserId) AS ownerFlag;
dump D;
```

Figure 7: Continuation of second Pig script

The above Pig script has taken 1 hour to process the data, do the appropriate map reduce jobs and finally provide with the desired results.

The third Pig script which we have written gives the Top 5 questions. The Top 5 questions are provided for individual tag. This Pig script has taken 45 minutes to process the data, do the appropriate map reduce jobs and finally provide with the desired results.

The third Pig script is as below:

```
DEFINE XPath org.apache.pig.piggybank.evaluation.xml.XPath();
POSTS = LOAD '/user/snehal6/data/Posts1' using org.apache.pig.piggybank.storage.XMLLoader('row') as (x:chararray);
QUES1 = FOREACH POSTS GENERATE
FLATTEN(REGEX_EXTRACT_ALL(x,'<row\\s*Id="(.*)"\\s*PostTypeId="1"\\s*.*CreationDate=".*?"\\s*.*Score=".*"\\s*ViewCount="(.*?)"\\s*.*Title="(.*)"\\s*Tag
s="&lt;([^&]*)&gt;&lt;([^&]*)&gt;"\\s*AnswerCount=".*"\\s*CommentCount=".*?"\\s*.*\\/>')) AS ( id:chararray, score:chararray, title:chararray, tag1:chararray);

QUES2 = FOREACH POSTS GENERATE
FLATTEN(REGEX_EXTRACT_ALL(x,'<row\\s*Id="(.*)"\\s*PostTypeId="1"\\s*.*CreationDate=".*?"\\s*.*Score=".*"\\s*ViewCount="(.*?)"\\s*.*Title="(.*)"\\s*Tag
s="&lt;([^&]*)&gt;&lt;([^&]*)&gt;"\\s*AnswerCount=".*"\\s*CommentCount=".*?"\\s*.*\\/>')) AS ( id:chararray, score:chararray, title:chararray,
tag1:chararray, tag2:chararray);

QUES3 = FOREACH POSTS GENERATE
FLATTEN(REGEX_EXTRACT_ALL(x,'<row\\s*Id="(.*)"\\s*PostTypeId="1"\\s*.*CreationDate=".*?"\\s*.*Score=".*"\\s*ViewCount="(.*?)"\\s*.*Title="(.*)"\\s*Tag
s="&lt;([^&]*)&gt;&lt;([^&]*)&gt;&lt;([^&]*)&gt;"\\s*AnswerCount=".*"\\s*CommentCount=".*?"\\s*.*\\/>')) AS ( id:chararray, score:chararray,
title:chararray, tag1:chararray, tag2:chararray, tag3:chararray);

QUES4 = FOREACH POSTS GENERATE
FLATTEN(REGEX_EXTRACT_ALL(x,'<row\\s*Id="(.*)"\\s*PostTypeId="1"\\s*.*CreationDate=".*?"\\s*.*Score=".*"\\s*ViewCount="(.*?)"\\s*.*Title="(.*)"\\s*Tag
s="&lt;([^&]*)&gt;&lt;([^&]*)&gt;&lt;([^&]*)&gt;&lt;([^&]*)&gt;"\\s*AnswerCount=".*"\\s*CommentCount=".*?"\\s*.*\\/>')) AS ( id:chararray,
score:chararray, title:chararray, tag1:chararray, tag2:chararray,tag3:chararray,tag4:chararray);

QUES5 = FOREACH POSTS GENERATE
FLATTEN(REGEX_EXTRACT_ALL(x,'<row\\s*Id="(.*)"\\s*PostTypeId="1"\\s*.*CreationDate=".*?"\\s*.*Score=".*"\\s*ViewCount="(.*?)"\\s*.*Title="(.*)"\\s*Tag
s="&lt;([^&]*)&gt;&lt;([^&]*)&gt;&lt;([^&]*)&gt;&lt;([^&]*)&gt;&lt;([^&]*)&gt;"\\s*AnswerCount=".*"\\s*CommentCount=".*?"\\s*.*\\/>')) AS (
id:chararray, score:chararray, title:chararray, tag1:chararray, tag2:chararray, tag3:chararray,tag4:chararray, tag5:chararray);


A11 = FOREACH QUES1 GENERATE (INT) id, tag1 as tag, (INT) score, title, tag1 as tag1, '' as tag2, '' as tag3, '' as tag4, '' as tag5;
A21 = FOREACH QUES2 GENERATE (INT) id, tag1 as tag, (INT) score, title, tag1 as tag1, tag2 as tag2, '' as tag3, '' as tag4, '' as tag5;
A22 = FOREACH QUES2 GENERATE (INT) id, tag2 as tag, (INT) score, title, tag1 as tag1, tag2 as tag2, '' as tag3, '' as tag4, '' as tag5;
A31 = FOREACH QUES3 GENERATE (INT) id, tag1 as tag, (INT) score, title, tag1 as tag1, tag2 as tag2, tag3 as tag3, '' as tag4, '' as tag5;
A32 = FOREACH QUES3 GENERATE (INT) id, tag2 as tag, (INT) score, title, tag1 as tag1, tag2 as tag2, tag3 as tag3, '' as tag4, '' as tag5;
A33 = FOREACH QUES3 GENERATE (INT) id, tag3 as tag, (INT) score, title, tag1 as tag1, tag2 as tag2, tag3 as tag3, '' as tag4, '' as tag5;
A41 = FOREACH QUES4 GENERATE (INT) id, tag1 as tag, (INT) score, title, tag1 as tag1, tag2 as tag2, tag3 as tag3, tag4 as tag4, '' as tag5;
A42 = FOREACH QUES4 GENERATE (INT) id, tag2 as tag, (INT) score, title, tag1 as tag1, tag2 as tag2, tag3 as tag3, tag4 as tag4, '' as tag5;
A43 = FOREACH QUES4 GENERATE (INT) id, tag3 as tag, (INT) score, title, tag1 as tag1, tag2 as tag2, tag3 as tag3, tag4 as tag4, '' as tag5;
A44 = FOREACH QUES4 GENERATE (INT) id, tag4 as tag, (INT) score, title, tag1 as tag1, tag2 as tag2, tag3 as tag3, tag4 as tag4, '' as tag5;
A51 = FOREACH QUES5 GENERATE (INT) id, tag1 as tag, (INT) score, title, tag1 as tag1, tag2 as tag2, tag3 as tag3, tag4 as tag4, tag5 as tag5;
A52 = FOREACH QUES5 GENERATE (INT) id, tag2 as tag, (INT) score, title, tag1 as tag1, tag2 as tag2, tag3 as tag3, tag4 as tag4, tag5 as tag5;
A53 = FOREACH QUES5 GENERATE (INT) id, tag3 as tag, (INT) score, title, tag1 as tag1, tag2 as tag2, tag3 as tag3, tag4 as tag4, tag5 as tag5;
```

Figure 8: Third Pig script

```
query_1bt query_2bt query_3bt query_4bt
A44 = FOREACH QUES4 GENERATE (INT) id, tag4 as tag, (INT) score, title, tag1 as tag1, tag2 as tag2, tag3 as tag3, tag4 as tag4, '' as tag5;
A51 = FOREACH QUES5 GENERATE (INT) id, tag1 as tag, (INT) score, title, tag1 as tag1, tag2 as tag2, tag3 as tag3, tag4 as tag4, tag5 as tag5;
A52 = FOREACH QUES5 GENERATE (INT) id, tag2 as tag, (INT) score, title, tag1 as tag1, tag2 as tag2, tag3 as tag3, tag4 as tag4, tag5 as tag5;
A53 = FOREACH QUES5 GENERATE (INT) id, tag3 as tag, (INT) score, title, tag1 as tag1, tag2 as tag2, tag3 as tag3, tag4 as tag4, tag5 as tag5;
A54 = FOREACH QUES5 GENERATE (INT) id, tag4 as tag, (INT) score, title, tag1 as tag1, tag2 as tag2, tag3 as tag3, tag4 as tag4, tag5 as tag5;
A55 = FOREACH QUES5 GENERATE (INT) id, tag5 as tag, (INT) score, title, tag1 as tag1, tag2 as tag2, tag3 as tag3, tag4 as tag4, tag5 as tag5;


U1  = UNION ONSCHEMA A11, A21;
U2  = UNION ONSCHEMA U1 , A22;
U3  = UNION ONSCHEMA U2, A31;
U4  = UNION ONSCHEMA U3 , A32;
U5  = UNION ONSCHEMA U4 , A33;
U6  = UNION ONSCHEMA U5 , A41;
U7  = UNION ONSCHEMA U6 , A42;
U8  = UNION ONSCHEMA U7 , A43;
U9  = UNION ONSCHEMA U8 , A44;
U10 = UNION ONSCHEMA U9 , A51;
U11 = UNION ONSCHEMA U10, A52;
U12 = UNION ONSCHEMA U11, A53;
U13 = UNION ONSCHEMA U12, A54;
U14 = UNION ONSCHEMA U13, A55;

B = GROUP U14 BY (tag);
C = FOREACH B {
    D = order U14 BY score DESC;
    E = LIMIT D 5;
    GENERATE group, FLATTEN(E);
    };

F = FOREACH C GENERATE group as tag, id as id, score as score, title as title, tag1 as tag1, tag2 as tag2, tag3 as tag3, tag4 as tag4, tag5 as tag5;

G = FILTER F BY tag is not null;

DUMP G;
```

Figure 9: Continuation of third Pig script

The fourth Pig script which we have written provides us with the information of Top 5 users for the tag provided. This Pig script has taken 2 hours 45 minutes to process the data, do the appropriate map reduce jobs and finally provide with the desired results.

The fourth Pig script is as below:

```
DEFINE XPath org.apache.pig.piggybank.evaluation.xml.XPath();
POSTS = LOAD '/user/snehal6/data/Posts1' using org.apache.pig.piggybank.storage.XMLLoader('row') as (x:chararray);
USERSF = LOAD '/user/snehal6/data/Users' using org.apache.pig.piggybank.storage.XMLLoader('row') as (x:chararray);
QUES1 = FOREACH POSTS GENERATE
FLATTEN(REGEX_EXTRACT_ALL(x,'<row\\s*Id="(.*)"\\s*PostTypeId="1"\\s*(AcceptedAnswerId=".*?"\\s*)*CreationDate="(....).*?"\\s*(DeletionDate=".*?"\\s*)*Score="(.*)"\\s*Vie
wCount=".*?"\\s*.*Tags="&lt;([^&]*)&gt;&lt;([^&]*)&gt;"\\s*AnswerCount=".*"\\s*CommentCount=".*?"\\s*(FavoriteCount=".*?"\\s*)*(ClosedDate=".*?"\\s*)*(CommunityOwnedDate=".*?"\\s*)*\
\/>')) AS ( id:chararray, acceptedId:chararray, cYear:chararray, delDate:chararray, score:chararray, tag1:chararray, tag2:chararray, ansCount:chararray,favCount:chararray ,
closedDate:chararray, commDate:chararray);
QUES2 = FOREACH POSTS GENERATE
FLATTEN(REGEX_EXTRACT_ALL(x,'<row\\s*Id="(.*)"\\s*PostTypeId="1"\\s*(AcceptedAnswerId=".*?"\\s*)*CreationDate="(....).*?"\\s*(DeletionDate=".*?"\\s*)*Score="(.*)"\\s*Vie
wCount=".*?"\\s*.*Tags="&lt;([^&]*)&gt;&lt;([^&]*)&gt;"\\s*AnswerCount=".*"\\s*CommentCount=".*?"\\s*(FavoriteCount=".*?"\\s*)*(ClosedDate=".*?"\\s*)*(CommunityOwnedDat
e=".*?"\\s*)*\/>')) AS ( id:chararray, acceId:chararray, cYear:chararray, delDate:chararray, score:chararray, tag1:chararray, tag2:chararray,
ansCount:chararray,favCount:chararray , closedDate:chararray, commDate:chararray);
QUES3 = FOREACH POSTS GENERATE
FLATTEN(REGEX_EXTRACT_ALL(x,'<row\\s*Id="(.*)"\\s*PostTypeId="1"\\s*(AcceptedAnswerId=".*?"\\s*)*CreationDate="(....).*?"\\s*(DeletionDate=".*?"\\s*)*Score="(.*)"\\s*Vie
wCount=".*?"\\s*.*Tags="&lt;([^&]*)&gt;&lt;([^&]*)&gt;"\\s*AnswerCount=".*"\\s*CommentCount=".*?"\\s*(FavoriteCount=".*?"\\s*)*(ClosedDate=".*?"\\s*)*(Co
mmunityOwnedDate=".*?"\\s*)*\/>')) AS ( id:chararray, acceId:chararray, cYear:chararray, delDate:chararray, score:chararray, tag1:chararray, tag2:chararray,
tag3:chararray, ansCount:chararray,favCount:chararray , closedDate:chararray, commDate:chararray);
QUES4 = FOREACH POSTS GENERATE
FLATTEN(REGEX_EXTRACT_ALL(x,'<row\\s*Id="(.*)"\\s*PostTypeId="1"\\s*(AcceptedAnswerId=".*?"\\s*)*CreationDate="(....).*?"\\s*(DeletionDate=".*?"\\s*)*Score="(.*)"\\s*Vie
wCount=".*?"\\s*.*Tags="&lt;([^&]*)&gt;&lt;([^&]*)&gt;&lt;([^&]*)&gt;&lt;([^&]*)&gt;"\\s*AnswerCount=".*"\\s*CommentCount=".*?"\\s*(FavoriteCount=".*?"\\s*)*(ClosedDate=
".*?"\\s*)*(CommunityOwnedDate=".*?"\\s*)*\/>')) AS ( id:chararray, acceId:chararray, cYear:chararray, delDate:chararray, score:chararray, tag1:chararray,
tag2:chararray,tag3:chararray,tag4:chararray, ansCount:chararray,favCount:chararray , closedDate:chararray, commDate:chararray);
QUES5 = FOREACH POSTS GENERATE
FLATTEN(REGEX_EXTRACT_ALL(x,'<row\\s*Id="(.*)"\\s*PostTypeId="1"\\s*(AcceptedAnswerId=".*?"\\s*)*CreationDate="(....).*?"\\s*(DeletionDate=".*?"\\s*)*Score="(.*)"\\s*Vie
wCount=".*?"\\s*.*Tags="&lt;([^&]*)&gt;&lt;([^&]*)&gt;&lt;([^&]*)&gt;&lt;([^&]*)&gt;&lt;([^&]*)&gt;"\\s*AnswerCount=".*"\\s*CommentCount=".*?"\\s*(FavoriteCount=".*?"\\s*
)*(ClosedDate=".*?"\\s*)*(CommunityOwnedDate=".*?"\\s*)*\/>')) AS ( id:chararray, acceId:chararray, cYear:chararray, delDate:chararray, score:chararray,
tag1:chararray, tag2:chararray,tag3:chararray,tag4:chararray,tag5:chararray, ansCount:chararray,favCount:chararray , closedDate:chararray, commDate:chararray);
A11 = FOREACH QUES1 GENERATE (INT) id, tag1 as tag;
A21 = FOREACH QUES2 GENERATE (INT) id, tag1 as tag;
A22 = FOREACH QUES2 GENERATE (INT) id, tag2 as tag;
A31 = FOREACH QUES3 GENERATE (INT) id, tag1 as tag;
A32 = FOREACH QUES3 GENERATE (INT) id, tag2 as tag;
A33 = FOREACH QUES3 GENERATE (INT) id, tag3 as tag;
A41 = FOREACH QUES4 GENERATE (INT) id, tag1 as tag;
A42 = FOREACH QUES4 GENERATE (INT) id, tag2 as tag;
A43 = FOREACH QUES4 GENERATE (INT) id, tag3 as tag;
A44 = FOREACH QUES4 GENERATE (INT) id, tag4 as tag;
A51 = FOREACH QUES5 GENERATE (INT) id, tag1 as tag;
A52 = FOREACH QUES5 GENERATE (INT) id, tag2 as tag;
```

Figure 10: Fourth Pig script

```
A52 = FOREACH QUES5 GENERATE (INT) id, tag2 as tag;
A53 = FOREACH QUES5 GENERATE (INT) id, tag3 as tag;
A54 = FOREACH QUES5 GENERATE (INT) id, tag4 as tag;
A55 = FOREACH QUES5 GENERATE (INT) id, tag5 as tag;
U1  = UNION ONSCHEMA A11, A21;
U2  = UNION ONSCHEMA U1 , A22;
U3  = UNION ONSCHEMA U2 , A31;
U4  = UNION ONSCHEMA U3 , A32;
U5  = UNION ONSCHEMA U4 , A33;
U6  = UNION ONSCHEMA U5 , A41;
U7  = UNION ONSCHEMA U6 , A42;
U8  = UNION ONSCHEMA U7 , A43;
U9  = UNION ONSCHEMA U8 , A44;
U10 = UNION ONSCHEMA U9 , A51;
U11 = UNION ONSCHEMA U10, A52;
U12 = UNION ONSCHEMA U11, A53;
U13 = UNION ONSCHEMA U12, A54;
U14 = UNION ONSCHEMA U13, A55;
ANSW = FOREACH POSTS GENERATE FLATTEN(REGEX_EXTRACT_ALL(x,'<row\\s*Id="(.*)"\\s*PostTypeId="2"\\s*ParentId="(.*)"\\s*.*OwnerUserId="(.*)"\\s*.*\\/>')) AS
(answerId:chararray, questionId:chararray, ownerUserId:chararray);
ANSW1 = FOREACH ANSW GENERATE (INT) answerId, (INT) questionId, (INT) ownerUserId;
USERS = FOREACH USERSF GENERATE
FLATTEN(REGEX_EXTRACT_ALL(x,'<row\\s*Id="(.*)"\\s*Reputation=".*"\\s*CreationDate=".*"\\s*DisplayName="(.*)"\\s*LastAccessDate=".*"\\s*Views=".*"\\s*UpVotes=".*"\\s*DownVot
es=".*"\\s*(ProfileImageUrl="(.*)"|#)\\s*AccountId=".*"\\s*\\/>')) AS (usersId:chararray, displayName:chararray, profileImageUrl:chararray, profileImage:chararray);
USERS1 = FOREACH USERS GENERATE (INT) usersId, displayName, profileImage;
B = JOIN U14 BY id, ANSW1 BY questionId;
C = JOIN B BY ownerUserId FULL, USERS1 BY usersId;
D = GROUP C BY (tag,ownerUserId);
E = FOREACH D GENERATE FLATTEN(group) AS (tag, ownerUserId), COUNT(C.answerId) AS ansCount, C.displayName as displayName, C.profileImage as profileImage;
E1 = FOREACH E GENERATE tag,ownerUserId, ansCount, FLATTEN(displayName) AS displayName, FLATTEN(profileImage) AS profileImage;
F =  GROUP E1 BY (tag);
G = FOREACH F{
    H = ORDER E1 BY ansCount DESC;
    I = DISTINCT H;
    J = LIMIT I 5;
    GENERATE group, FLATTEN(J);
    };
K = FILTER G BY tag is not null;
DUMP K;
```

Figure 11: Continuation of fourth Pig script

The results from these are uploaded into SQLite. In the service we provide, there is a text area present where one can type the question they would like to post. Once the individual types the question the next job is to find out the tags in the question. For that the whole question will be parsed and each word will be compared to the words in the Posts.xml file. Once the comparison is done the words which

match with that of those present in the database would be displayed on the screen once the user submits the "Get Tags" request on the screen. Once the query is submitted all the tags which have been identified will be displayed on the screen. Then the user has to select a minimum of 1 tag and a maximum of 5 tags from the tags list provided below. A check box is provided for the user to select the tags. Once the user clicks on the "Get Stats" button few database queries will take place in the background. The results obtained are displayed on the screen. The results are displayed per tag.

First the results of a tag selected from the given list is displayed. Then similarly for all the tags which the user has selected will be displayed below each other. Amongst the selected tags a tag is taken first and for that tag the statistics are shown first. The statistics include Total number of related questions, Total number of answers, Total number of accepted answers, Total number of closed questions, Total number of current open questions, Total numbers of users active on the related topic, Chance of getting an answer. Then after this Top 5 questions related to that tag are displayed on the screen. Then the Top 5 users related to that tag are displayed on the screen. Then a graph is shown showing the trends of that tag in the past years from 2008 to 2015.

After the above results for a single tag is presented the same will be repeated for other tags which the user has marked in the checkbox.

V. RESULTS

The result of first Pig script is as below:

```
(mapreduce,2008,9,4,0,279,61,1)
(mechanize,2014,314,115,0,128,262,7)
(merb-auth,2008,2,2,0,1,2,0)
(mergeinfo,2012,5,4,0,20,7,0)
(mergetool,2009,4,2,0,44,9,0)
(messenger,2013,28,13,0,26,27,1)
(metatable,2014,15,6,0,28,20,0)
```

Figure 12: Results of first Pig script

The image shown above gives the statistics of each tag per year. The first column consists of tag, second column consists of year, third column consists of number of questions related to that tag in that year, fourth column consists of total number of accepted answers related to that tag in that year, fifth column consists of number of deleted questions related to that tag in that year, sixth column consists of the total score related to that tag in that year, seventh column consists of total number of answers related to that tag in that year, eigth column consists of total number of closed questions related to that tag in that year.

The results of second Pig script is as below:

```
(internet-explorer-9,2010,150)
(intrusion-detection,2010,3)
(io-completion-ports,2012,11)
(ios-standalone-mode,2012,5)
(iphone-softkeyboard,2008,5)
(iphone-softkeyboard,2013,10)
```

Figure 13: Results of second Pig script

The image shown above gives the results of total active users for each tag per year. The first column consists of tag name, second column consists of year, and third column consists of total number of active users for that tag in that year.

The result of third Pig script is as below:

```
(apache-pig,3356259,90017,Difference between Pig and Hive? Why have both?,hadoop,hive,apache-pig,piglatin,)
(apache-pig,13911501,28664,When to use Hadoop, HBase, Hive and Pig?,hadoop,hbase,hive,apache-pig,)
(apache-pig,9900761,25476,PIG how to count a number of rows in alias,hadoop,apache-pig,,,)
(apache-pig,3515481,20119,Pig Latin: Load multiple files from a date range (part of the directory structure),hadoop,apache-pig,piglatin,,)
(apache-pig,5013003,13885,How do I parse JSON in Pig?,json,apache-pig,piglatin,,)
(hadoop2,22316187,2125,Datanode not starts correctly,hadoop,hadoop2,,,)
```

Figure 14: Results of third Pig script

The image shown above gives the results of top five queations for each tag. The first column consists of the tag, second column consists of the question ID, third column consists of the score, fourth column consists of the question, fifth column consists of the tags.

The result of fourth Pig script is as below:

```
(allegro,1968,78,Konrad Rudolph,)
(allegro,4381,39,Vicent Marti,)
(amazon-javascript-sdk,174184,3,TJ-,)
(angular-seed,39396,14,Carl G,)
(angular-seed,345944,9,Liad Livnat,)
(angular-seed,38611,8,zilupe,)
(angular-seed,1691,1,olore,)
```

Figure 15: Results of fourth Pig script

The image shown above gives the top five users for each tag. The first column consits of tag name, second column

consists of owner user ID, third column consists of total number of answers the user has given related to that tag, fourth column consists of the name of that user, fifth column consists of the profile image of the user if it is present in the database.

In the page we have developed a text box is given for the user to write the question that they would like to post. For example let the question be “What is the best database for Django”. After typing the question if the user clicks on “Get Tags” button the question is parsed through the database to compare with that of the tags present in the database and the words which match with those present in database will be shown below with a check box beside it. Now the user has to select the tags present in the checkboxes and select a minimum of 1 tag and maximum of 5 tags. In the example below two tags have been detected and the user has selected the one tag amongst the two. The results will be provided for the selected tag.

Figure 16: question entered, the tags predicted and user selects the tags.

Now if the user clicks on “Get Stats” button the statistics will be shown as below.

Figure 17: Statistics for the tag selected

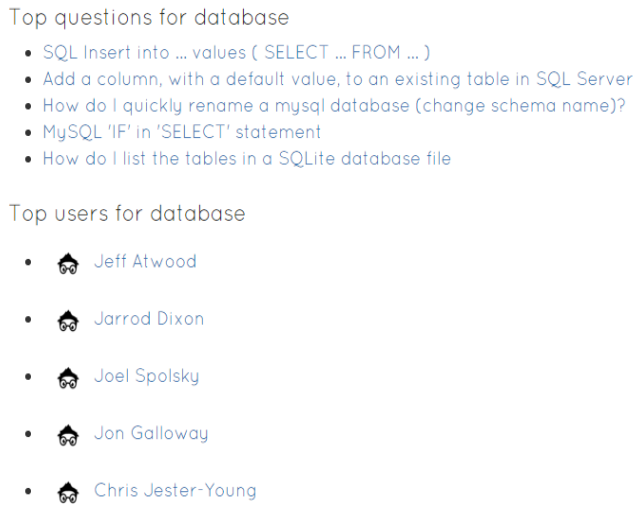

Figure 18: Top 5 questions and Top 5 users for the selected tag

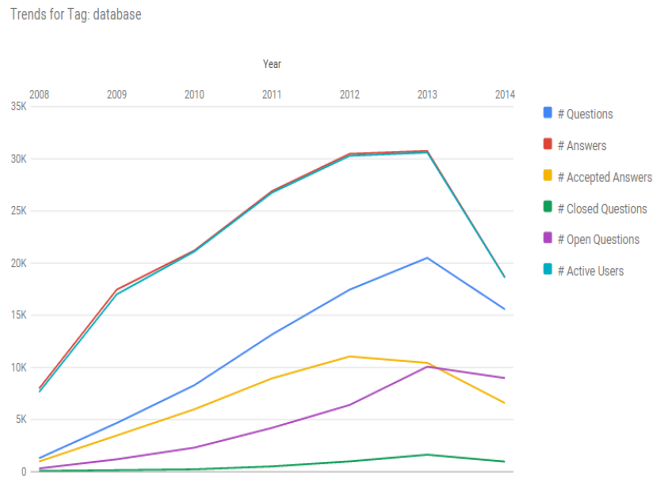

Figure 19: Trends of the selected tag

The trends of the tags are given for the individual tag based on the year. On X-axis we have the year and on the Y-axis we have the value related to that.

Thus this is how our site takes the input i.e. the question and outputs the statistics related to the tags of the question posted.

VI. Conclusion

The results we have shown here gives the user an idea about the statistics of the tags the user has selected. In addition to the statistics we also added few other additional services like giving the top five questions related to the tags and also the top five users of the tags. In the end we have also shown the statistics in a chart based on year. One can get a general idea about the technologies trends over a period of time from 2008-2015.

ACKNOWLEDGMENT

We would like to thank Doctor Karuna Joshi P for her excellent guidance and feedback throughout the model development and analysis process.

REFERENCES

[1] Clayton Stanley, Michael D. Byrne. 2013. **Predicting Tags for StackOverflow Posts** In *12th International Conference on Cognitive Modelling*. , pages 414-419.

[2] Fabio Calefato, Filippo Lanubile, Maria Concetta Marasciulo, Nicole Novielli. **Mining Successful Answers in Stack Overflow**. In Proceedings of MSR 2015, The 12th Working Conference on Mining Software Repositories.

[3] https://en.wikipedia.org/wiki/Django_(web_framework)

[4] https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework)

[5] http://iccm-conference.org/2013-proceedings/papers/0077/paper0077.pdf

[6] https://www.kaggle.com/c/predict-closed-questions-on-stack-overflow/data

[7] http://blog.stackoverflow.com/2009/06/stack-overflow-creative-commons-data-dump/

[8] http://stackoverflow.com/help/closed-questions

[9] http://stackoverflow.com/questions/10059594/a-simple-explanation-of-naive-bayes-classification

[10] http://meta.stackexchange.com/questions/2677/database-schema-documentation-for-the-public-data-dump-and-sede

[11] https://pig.apache.org/docs/r0.7.0/piglatin_ref2.html#Arithmetic+Operators+and+More

[12] https://pig.apache.org/docs/r0.9.1/func.html#size

[13] https://www.qubole.com/resources/cheatsheet/pig-function-cheat-sheet/

[14] http://www.tutorialspoint.com/ajax/pdf/what_is_ajax.pdf