Data Extraction and Visualization Application

e-Nable’s Google+ Community

Shashank Narkhede

February 2016

Abstract

In the age of the Internet, data is abundantly available. Many social networking sites generate huge amount of data on daily basis. This data is in various forms and can be utilized for various purposes. It is important to have a mechanism to utilize this data efficiently. This paper deals specifically with Google+. We discuss a mechanism to automatically extract data from a Google+ community, maintain the data and use the data for carrying out various type of data analysis and producing visualizations.

1 Introduction

In this paper we use Google+ private community page(e-Nable). The Google+ page for e-Nable is a community that wants to build a network for design, customization, and fabrication of 3D-printed prosthetics. Currently, Google does not provide any an API(application program interface) to readily access information from a community page. This information is very vital for various organizations to enhance their functioning. This data can be utilized to categorize different individuals from their posts and comments on the community. Moreover, new insights and patterns can be discovered from the interaction between individuals that can help the community grow and help more people.

Data present in private Google+ communities is accessible only to the members of the community. The community is gradually growing, which increases the posts and discussion that happen in the community. Although the data is accessible to the members the data is distributed across different categories, posts and comments. There is no way to collectively analyse this data. Many important posts can be left unaddressed or unnoticed. As the members in the community increase, this number will increase further. This data can be used to understand various aspects of the community. There can be interesting patterns among the users that can help further enhance development of the community. We decided to understand architecture of Google+ community pages and de-
velop an application that will extract required data and store this data into a
database. We specifically focus on e-Nable community in this paper.

Major objectives of this project were:

1. Build an infrastructure to collect data from Google+ community page
2. Perform analysis on the data collected
3. Create visualizations on the data
4. Package a platform that can be developed in future

2 Related Work

[KNB13] is a web extraction suite. However, it was complex to understand the
tool’s GUI and automating this tool against a Google+ login could have been
difficult. This tool used a browser like interface to extract data from web pages.
Other tools analyzed include [SHN13], [SMdR07], [KT14], [VdSdMC06] and
[GFS13].

Although all these methods provide a way for web extraction, most of these
tools require manual intervention. Another problem with private Google+ com-

2

munities was the log in required to access these community pages. It was difficult
to integrate log in process with aforementioned tools. Moreover, required appli-
cation required customization in data collection process as well as in managing
storage of this data. Due to all these reasons we decided to build a new system
customized to achieve our objectives.

3 Data

Data used for the project was collected from e-Nable’s Google+ private com-
munity page. This data included posts, users and comment from the com-

2

munity. Data was cleaned before analyzing. Data included over 2500 users,
over 9000 posts and over 50000 comments. Community has around 8500 users,
around 2500 users had either a post or comment in the community.

4 Methods

Considering the web-page architecture of Google+ community page, following
approach was finalized.

4.1 Data Extraction:

The first block represents Google+ community page for e-Nable. This was a
private page and a new user was created to access the community page and
proceed with data extraction. The url for the community page is [https://plus.google.com/u/0/communities/102497715636887179986](https://plus.google.com/u/0/communities/102497715636887179986)
User used for analysis was enabledataforme@gmail.com. Python script was used to create requests and access the community page. After understanding the structure of the Google+ community web page, Beautiful Soup was used to modify the parse tree generated by the community page and parse required data. Every hour top posts from the community page were extracted and the URL for these posts was stored into a url.txt file. This file was updated throughout the day. At the end of the day, all details from these posts were extracted, structured and stored into the database (discussed in later section). This architecture was simple and robust in terms of customization and by end of this stage we had a list of URL’s for all the posts created or commented on a particular day.

4.2 Data Storage:

The data was structured into three components
1. Post: This consists of all data related to post. This was the biggest collection consisting of URL link of the post, date of the post, category, number of people who +1ed the post, number of comments in the post, user associate with the post and the content of the post.

2. User: This consists of userId and user name.

3. Comment: This consists of the comment, user associated with the comment and the post associated with the comment.

The data was dynamic and many attributes were unavailable in some posts. We decided to use a schema-less database and used MongoDB for data storage. Two databases were created one was for testing purpose and one was for actual application. Names of the databases were 'test_database' and 'live_database'. Before storing a post, if the user did not exist in the user collection, the user collection was updated first and then the Post collection was updated. Similarly, for comment collection, the user was updated first followed by validation of existence of post and finally, the comment collection was updated. One the architecture was tested on test_database, the process was automated and cron jobs were scheduled to carry out the process on Linux machines.

Complete code for the project can be found at https://github.com/IamShank/finalProject.git

Figure 3: crontab entries to execute data extraction and carry out database update in live_database

At this stage, a database was set up, data was extracted from Google+ community page on daily basis and was updated into the database.

Note: Before starting the entire process, all old post URL’s were separately extracted and updated in url.txt file and database was updated for all of them. Also once the database is updated url.txt file is cleaned up daily.

4.3 Data Analysis and Visualization:

Initially, for analysing major words talked in the community we used https://github.com/amueller/word_cloud/tree/master/wordcloud for creating word clouds from the corpus. From the data collected in MongoDB, we extracted posts and comments for every individual separately. The major word talked about in the community was hand. Following table shows number of posts in each category.
<table>
<thead>
<tr>
<th>Category</th>
<th>Number of posts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cool and Relevant Stuff</td>
<td>1064</td>
</tr>
<tr>
<td>Sales/promo codes etc.</td>
<td>16</td>
</tr>
<tr>
<td>Recorded Hangouts (HOAs)</td>
<td>66</td>
</tr>
<tr>
<td>From Affiliated Sites</td>
<td>48</td>
</tr>
<tr>
<td>Handomatic</td>
<td>46</td>
</tr>
<tr>
<td>e-Nable Volunteer Jobs</td>
<td>210</td>
</tr>
<tr>
<td>General Discussion</td>
<td>2451</td>
</tr>
<tr>
<td>Research &amp; Development</td>
<td>1618</td>
</tr>
<tr>
<td>Seeking a Device</td>
<td>219</td>
</tr>
<tr>
<td>Teams</td>
<td>15</td>
</tr>
<tr>
<td>Organizational Support Team</td>
<td>109</td>
</tr>
<tr>
<td>Introductions</td>
<td>851</td>
</tr>
<tr>
<td>Resources</td>
<td>154</td>
</tr>
<tr>
<td>Case Studies Stories &amp; Prints</td>
<td>746</td>
</tr>
<tr>
<td>Real World Events</td>
<td>115</td>
</tr>
<tr>
<td>Education</td>
<td>366</td>
</tr>
<tr>
<td>Myoelectric Prosthesis</td>
<td>104</td>
</tr>
<tr>
<td>Announcements &amp; Admin</td>
<td>472</td>
</tr>
<tr>
<td>Legal Documents $$</td>
<td>47</td>
</tr>
</tbody>
</table>

After understanding the extent of interactivity required for the visualizations, we chose plotly for the visualizations. Plotly is a visualization tool that provides online as well as offline plots and also has interfaces in many languages. Plotly also provides zoom and hover capability on the generated charts. After exploring basic plotly graphs, we plotted a graph of number of posts in every category. We installed plotly and produced offline plots. We also registered for a plotly API and produced online plots. However, plotly has limitations on number of online charts as a free user. We extracted posts on each day since the start of the community and generated a chart for date vs number of posts on each day. This provided a birds eye view about the activity in the community. We also generated similar chart for comments. One important chart plotted was user on the Y axis, date on X-Axis and each point in the chart denoted when a particular user posted in the community. It was a cumulative chart to depict how community has grown over time.

4.4 Security Approaches for sharing data with other developers:

1. **Wrapper around MongoDB**: Build a web server on our machine and create minimum required API’s for data access.
   - Advantages: Control and customize user access to our data
   - Disadvantages: Spend time on creating and maintaining these API’s and
create documentation for API usage.

2. **Remote Access**: Remote access to mongodb server which can leverage inbuilt mongodb capabilities to control user based access to our data. This process is well documented.
   Advantages: Can be achieved quickly
   Can have limited access(read/update on collections)
   Disadvantages: Need to create roles for specific purposes.

3. **Use MongoDB labs**: MongoDB labs can host the server on-line.
   Advantages: Consistent performance across locations for visualizations.
   Disadvantages: Not very matured service (More data will incur cost, 500 MB free data allowed) and Data will be in cloud.

4. **Dummy(Subset) Database with remote access**: Create a database by changing names and urls and limiting the content of the database.
   Advantages: Real data will never be shared with other developers.
   Disadvantages: Encrypting desired parameters to dump in dummy database might create issues in redirection (in case of complex ideas for visualization).

5 **Results**

Following are various visualizations generated as a part of this project.
Figure 4: Date vs Number of Posts for e-Nable Google+ community
This plot shows date versus the number of posts in the community. The community was more active in the period of July 2014-March 2015. It would be interesting to discover the reason for this activity.

Figure 5: Date vs Number of Comments for e-Nable Google+ community
Figure 6: Pie-chart showing different categories of posts
Majority of discussion in the community was labelled general discussion by the post owner. However, it was interesting to observe that the second best was Research and Development which indicated enthusiasm of the community.

Figure 7: Number of posts in each category on the community’s Google+ page
Figure 8: Word cloud representing all user corpus
It is clear that the most frequent word talked about in the community was hand. It was interesting to see frequency of Thank, this can be an indication of people helped by the community or the helping nature of people in the community.
Figure 9: Word Cloud representing user corpus with a different scaling

WORDS = hand, will, Thank, one, work, help, design, 3D, print, finger, need, use, make, printer, NABLE, know, part, also, great, printing, see, think, good, file, new, look, printed, time, ll, Hi, well, way, ve, thing, idea, much, want, people, using, Raptor, arm, working, now, palm, please, first, post, really, don, prosthetic, community, right, sure, try, us, let, love, project, might, material, anyone, something, used, re, may, group, device, thumb, https, test, looking, lot, video, made, able, take, find, going, google, year, filament, still, link, gauntlet, model, enable, wrist, little, better, support, send, email, recipient, Hello, go, problem, got, two, someone, build, bit, everyone, done, back, PLA, student, put, best, size, around, give, interested, start, next, week, seem, STL, hope, Cyborg, form, many, nice, day, thought, fit, maybe, question, making, area, Beast, version, Jon, getting, open, cool, thingiverse, different, even, school, keep, pretty, grip, free, small, end, issue, contact, come, Ye, awesome, trying, add, share, yet, AB, already, point, together, Welcome, though, found, guy, process, de, org, joint, team, enough, since, picture, happy, volunteer, pin, another, set, scale, photo, Maker, Jen, screw, forward, soon, available, experience, without, place, etc, name, Owen, cost, possible, old, information, feel, user, hole, piece, currently, show, Brown
Figure 10: Plot representing Posts by all users across the community over time. This chart provides complete insight on the activity of the community from start. It consists of users who posted in the community over time. Using this chart as basis, any post in the community can be visualized in future.
6 Conclusion and Future Work

A basic application was successfully designed to extract data from Google+ community page. This application was built for e-Nable’s community page, however, we believe that this will work on any other Google+ community because the structure of Google+ page remains same. We successfully demonstrated generation of interactive visualizations using plotly. The application was monitored over some time and we observed that the application performed data extraction efficiently.

Currently, we are unable to extract Java-script generated content from the community web page. In future, we plan to extract this content and perform through analysis on the data collectively. We also plan to perform topic modelling on available data. From visualizations perspective, we plan to enhance interactivity of Figure 10. We plan to add a click event which will enable authenticated community members to directly see the post on Google+ community page.

References


[VdSdMC06] Márcio L. A. Vidal, Altigran S. da Silva, Edleno S. de Moura, and João M. B. Cavalcanti. Gogetit!: A tool for generating structure-
Appendices

A Data Extraction Instructions:

Anaconda distribution of python was used. The system works on python 2.7. This package uses enabledataforme@gmail.com as the user to access the community. Credentials are present in the code’s CONSTANT.py file.

- **lib folder:** This folder consists of class files. These files can be modified to add more attributes in the collection. To add these attributes into the database other dataStore.py will have to be modified.

- **MongoDB Interaction:** pymongo was used to interact with MongoDB. Database name: live_database
  LIVE Collection names are:
  - users
  - Post
  - Comment

Following commands were used to create index in MongoDB collections opening a python interactive shell.
```
from pymongo import MongoClient
client = MongoClient('localhost', 27017)
db = client.live_database
db.users.create_index("name")
db.Post.create_index("url")
db.Comment.create_index("comment")
```
Remove all entries from the MongoDB collection:
```
db.test_users.delete_many()
db.test_Comment.delete_many()
db.test_Post.delete_many()
```

E.g., To quickly interact with the database, open a python interactive shell and type following commands, it will display names of all users in the user collection. from pymongo import MongoClient
```
client = MongoClient('localhost', 27017)
for users in db.users.find() :
    users['name']
```
• **daemon:** Figure 3 shows crontab entries. There are two files one runs on hourly basis and the other on daily basis. The first entry denotes that this command runs every 26th minute of an hour and the second command denotes that this command runs at 00:01 everyday. First file stores post URL’s in url.txt file and second file extracts data from all URL’s and updates the database. This file consists of small functions to extract various attributes from google+ community web page. If the path of url.txt has to be changed, it can be modified in storePostURLs.py. Make sure to change the path in dataStore.py as well. Ideally, this can later be controlled using CONSTANT.py file once all paths are fixed. The crontab entries can be modified using following command:

```
VISUAL=vi crontab -e
```

Once the file is modified and saved, the job immediately starts working.

Experiments were made to extract more user data e.g. gender, education etc, however this was not completely integrated with the daemon as more experimentation is required with the code. The experiments can be found in getPostData.py file.

### B Visualization:

- **word cloud:** The wordcloud module can be installed directly for conda using instructions on [https://github.com/amueller/word_cloud](https://github.com/amueller/word_cloud). After installation, use following command:

```
python getAllUserCorpus.py live
```

This will extract details from live database, append all posts and comments by a user in a json file and store it with user name under corpus folder in same directory. Once json files are generated for every user, navigate to corpus folder using cd command and use following command (This will generate figures 8 and 9) to create the wordcloud:

```
python processCorpus.py
```

- **Plotly visualizations:** To install plotly follow instructions on [https://plot.ly/python/getting-started/](https://plot.ly/python/getting-started/). Plotly API key is already registered and credentials are stored in every file using plotly(This can be centralized using CONSTANT.py later). All plotly visualizations were based on a pickle dump of database. This can be modified and run directly on the database if required later. To get the pickle file, run the following command:

```
python getPostsPickleDump.py live
```
This will create a file allPostData.p with all required information for visualizations. To generate Figure 4, run the following command:
```
python processPosts.py
```

To generate Figure 5, run the following command:
```
python DateVsComments.py
```
To generate Figure 6, run the following command:
```
python processCategoryPosts.py
```
Note: This also creates word cloud for a particular category (Education Category is hard coded currently)

To generate Figure 7, run the following command:
```
python CategoryVsPosts.py
```

To generate Figure 10, run the following command:
```
python theCumultivePlot.py
```

All visualizations are set to run offline by default. On-line visualizations can easily be generated by replacing `plotly.offline.plot` with `py.plot` in the code files. All files already have a plotly username and registered API key.