Analyzing User Behavior on Facebook's "Hurricane Sandy Lost and Found Pets" Page to Improve Support for Pet Matching in Crisis Informatics Applications

Amrutha Rajiv

University of Colorado at Boulder, amrutha.rajiv@colorado.edu

Follow this and additional works at: https://scholar.colorado.edu/csci_gradetds

Part of the Computer Sciences Commons

Recommended Citation

https://scholar.colorado.edu/csci_gradetds/77
ANALYZING USER BEHAVIOR ON FACEBOOK'S "HURRICANE SANDY LOST AND FOUND PETS" PAGE TO IMPROVE SUPPORT FOR PET MATCHING IN CRISIS INFORMATICS APPLICATIONS

by

AMRUTHA RAJIV

B.E., Anna University, India. 2010

A thesis submitted to the
Faculty of the Graduate School of the
University of Colorado in partial fulfillment
of the requirement for the degree of
Master of Science
Department of Computer Science
2013
This thesis entitled:

Analyzing User Behavior on Facebook's "Hurricane Sandy Lost and Found Pets" Page to Improve Support for Pet Matching in Crisis Informatics Applications written by Amrutha Rajiv

has been approved for the Department of Computer Science

________________________________________

Kenneth Anderson

________________________________________

Leysia Palen

________________________________________

James Martin

Date______________

The final copy of this thesis has been examined by the signatories, and we Find that both the content and the form meet acceptable presentation standards Of scholarly work in the above mentioned discipline.
Rajiv, Amrutha (M.S., Computer Science)

Analyzing User Behavior on Facebook's "Hurricane Sandy Lost and Found Pets" Page to Improve Support for Pet Matching in Crisis Informatics Applications

Thesis directed by Associate Professor Kenneth Anderson

In the aftermath of mass emergencies, a large number of pets are displaced from their families. Pet reunification in crisis situations is therefore a serious problem. A number of digital volunteers were seen participating in interactions on social media services such as Facebook and Twitter with the intention of pet to family reunification. We are developing software to aid the members of the public with these activities, EmergencyPetMatcher (EPM), a collaborative web application that aims to help connect people with their lost pets. The goal of this thesis is to develop algorithms and monitoring capabilities to identify those digital volunteers (within EPM) who are actively involved with pet reunification and amplify/augment their actions to be more effective and to encourage them to continue to participate in the system. Since EPM is not deployed, we turned to the "Hurricane Sandy Lost and Found Pets" Facebook page to enable an analysis of on-line pet matching behavior. This Facebook page is dedicated to finding homes for lost and found pets from the aftermath of Hurricane Sandy. We developed a software tool to extract data from public Facebook pages such as this one. We also present an analysis of the Facebook data. The analysis details various user behaviors that are congruent to what we expect to see on EPM. We describe how we used the Facebook data to simulate expected behavior in EPM, generating log files that mimic the behavior we saw in Facebook. In particular, these log files represent activities of hypothetical EPM users who exhibit behavior similar to users seen on the Facebook page. We then present an
analysis of the EPM log file data and draw conclusions about how these activities can be used to reward future EPM users to bolster further pet reunification activities on EPM, helping to create a more efficient and effective crisis informatics application in support of pet reunification activities.
Dedication

This work is dedicated to my parents Revathy and Rajiv and my brother Ashwin.
Acknowledgements

I would like to thank my advisor, Dr. Kenneth Anderson for guiding me through this thesis, I am very grateful for his patience and support throughout this project. I would also like to thank him for giving me the opportunity to work on Emergency Pet Matcher. I want to thank my committee members Dr. Leysia Palen and Dr. James Martin for reviewing and evaluating my thesis. I would like to thank Joanne White for helping me understand many aspects of the Facebook data. I express my sincere thanks to the Emergency Pet Matcher team – Mario Barrenechea, Sahar Jambi, Mazin Hakeem and Lise St. Denis, for their support and encouragement throughout this project. Lastly I express my gratitude to all my friends who gave me valuable comments by reviewing many chapters of this thesis.
Contents

1 Introduction ........................................................................................................................................... 1

2 Background ........................................................................................................................................ 4

  2.1 Emergency Pet Matcher (EPM) ........................................................................................................ 4

    2.1.1 Pet Report.................................................................................................................................. 5
    2.1.2 Matching a pet............................................................................................................................ 7
    2.1.3 Pet Match .................................................................................................................................... 8
    2.1.4 Successful Matches .................................................................................................................... 9
    2.1.5 EPM Users .................................................................................................................................. 10

  2.2 EPM Log Files ................................................................................................................................. 12

    2.2.1 Apache Server Log Files ........................................................................................................... 12
    2.2.2 User Activity Log Files .............................................................................................................. 13

  2.3 Analyzing Facebook Behavior to Support matching on EPM ....................................................... 13

3 The Data Extraction Process .................................................................................................................. 15

  3.1 The Graph API ................................................................................................................................ 15

    3.1.1 The post object .......................................................................................................................... 16

  3.2 Function of the data extraction tool ................................................................................................. 19

  3.3 Process flow and description of the tool ......................................................................................... 19

  3.4 Database schema for stored Facebook Data .................................................................................... 20

  3.5 Challenges in getting data from Facebook ...................................................................................... 22

  3.6 Evaluation of the tool ....................................................................................................................... 23

4 Analysis of the Facebook Data ................................................................................................................ 25

  4.1 Splunk .............................................................................................................................................. 25

    4.1.1 Challenges ................................................................................................................................. 26

  4.2 Findings in the Sandy’s Pets Data ..................................................................................................... 26

    4.2.1 Lost or Found Pet Posts .............................................................................................................. 28
    4.2.2 User activities: Shares, Likes and Comments ........................................................................... 33
    4.2.3 Patterns in Comments ............................................................................................................... 34

  4.3 Anomalies in the Facebook data ....................................................................................................... 38

5 Translation of Facebook data to EPM Data ............................................................................................ 40

  5.1 Protecting User Privacy ................................................................................................................... 41
5.1 Generating EPM Log File data ................................................................. 41
5.2 Pets ........................................................................................................... 41
5.3 Pet report submissions ............................................................................. 42
  5.3.1 Generating data for Pet Report Submissions ...................................... 43
5.4 Viewing Pet Reports ................................................................................ 43
  5.4.1 Generating data for Pet Report Views .............................................. 44
5.5 Commenting on a Facebook Post and Bookmarking on EPM ................. 44
  5.5.1 Generating data for bookmarking a pet ........................................... 45
5.6 Creating and Voting on Pet Matches ....................................................... 45
  5.6.1 Suggesting Matches via comments – Creating Pet Matches .............. 46
  5.6.2 Viewing Pet Matches ....................................................................... 47
  5.6.3 Voting on Pet Matches ..................................................................... 48
5.7 Challenges ............................................................................................... 49
5.8 EPM data .................................................................................................. 49

6 Analysis of EPM Data .................................................................................. 50
  6.1 Extracting Data from EPM log files ....................................................... 50
  6.2 Extraction of Features .......................................................................... 51
    6.2.1 Challenges in Extracting Features ................................................. 52
  6.3 Classifying the Data ............................................................................... 53
    6.3.1 Constructing a training set .............................................................. 53
    6.3.2 Choosing classifiers ...................................................................... 54
    6.3.3 Challenges in Building a Model to Classify EPM Users ............... 55

7 Conclusion ..................................................................................................... 56
  7.1 Summary ................................................................................................. 56
  7.2 Discussion ............................................................................................... 57
  7.3 Contributions ........................................................................................... 58
  7.4 Future Work ............................................................................................. 59

Bibliography ..................................................................................................... 61

APPENDIX ........................................................................................................ 63
  A Python Code For Tool Developed To Extract Data From A Facebook Page... 63
Tables

Table

Table 4.1 Number of shares, likes and comments per month................................. 33
Table 6.1 Confusion matrix for the model built using a Decision Tree Classifier .... 54
Table 6.2 Confusion Matrix for the model built on a Naive Bayes Classifier ........ 55
Figures

Figure 2.1 Screenshot of EPM’s homepage ................................................................. 5
Figure 2.2 Screenshot of pet report submission form ............................................. 6
Figure 2.3 Screenshot of pet report dialog page ....................................................... 7
Figure 2.4 Screenshot of the matching workspace ............................................... 8
Figure 2.5 Screenshot of the pet match page ........................................................... 9
Figure 2.6 Screenshot of the verification page ....................................................... 10
Figure 2.7 Screenshot of a Profile page ................................................................. 11
Figure 2.8 Screenshot of bookmarks Page ........................................................... 11
Figure 3.1 A screenshot of a section of the Sandy’s Pets page where the user generated posts can be viewed ................................................................. 16
Figure 3.2 Process flow execution of the data extraction tool ............................... 20
Figure 3.3 Relational database Schema for the SQLite database showing tables that store albums and photos related information ............................................ 21
Figure 3.4 Relational database Schema for the SQLite database showing tables that store posts related information ......................................................... 22
Figure 4.1 Representation of data in Splunk .......................................................... 26
Figure 4.2 Number of posts generated on the FB page by month ....................... 27
Figure 4.3 Distribution of the posts generated by users between the categories Lost, Found, craigslist posts, adopted pets, pets that were re-united with families and Unknown posts ................................................................. 30
Figure 4.4 Distribution of posts generated by page owner labelled as Lost, Found, craigslist posts, adopted pets, pets that were re-united with families, rainbow bridge pets and Unknown posts. ................................................................. 32
Figure 4.5 The graph shows a comparison of the number of likes and shares for every month between Oct-12 and May-13 ................................................................. 34
Figure 4.6 Graph showing the frequency of the top 20 comments used to comment about sharing a post ................................................................. 35
Figure 4.7 Comparison of Number of Shares, number of comments and number of comments about sharing ................................................................. 36
Figure 4.8 Screenshot of matching activity as seen on the Sandy’s Pets page .... 37
Figure 4.9 Graph showing number of comments with matching activities and number of unique users who authored those comments ................................................................. 38
Figure 4.10 Screenshot of Facebook posts showing duplicate posts .................. 39
Figure 5.1 Screenshot of a post about a Found Cat on the Sandy's Pets page .... 42
Figure 5.2 Screenshot of user553's activity log showing the user's pet report submission activity ................................................................. 43
Figure 5.3 Screenshot of Apache server logs showing "GET" requests indicating pet report views ........................................................................................................ 44
Figure 5.4 Screenshot of activity log of a user who bookmarked a pet report .... 45
Figure 5.5 Apache Server Log instance of a match proposed by user398 between pet reports with ID's 815 and 666 ........................................................................................................ 47
Figure 5.6 Screenshot of user14076’s Activity Log showing the user's action of proposing a petmatch with ID 127 ........................................................................................................ 47
Figure 5.7 Screenshot of Apache server log showing a record of “GET” request when user398 viewed Pet Match with ID 216 ........................................................................................................ 48
Figure 6.1 Schema diagram of database tables used to store log information .......... 51
Chapter 1

Introduction

Research has identified new forms of digital behaviors emerging in the form of volunteers self-organizing and coming together on social media platforms, including in the aftermath of crisis situations [20]. These behaviors are studied by crisis informatics [16,20], a domain that looks at the ways members of the public make use of computer-mediated communication to make sense of mass emergency events and to organize, coordinate, and collaborate during these events. Pet to family reunification activities have been observed among interactions of many such users on social media platforms such as Facebook [4] and Twitter [21].

Crisis informatics seeks not only to understand these behaviors but also to develop techniques and tools to make members of the public more effective in responding to these events. As a result, we are developing software called EmergencyPetMatcher (EPM) to aid members of the public with pet reunification activities. EPM is a collaborative web application whose goal is to reunite lost pets with their families.

The pet displacement problem in mass emergency events is significant. After Hurricane Katrina an estimated 200,000 pets were displaced from their owners and only 10,000 of these animals were reunited with their families. Pets are an integral part of over sixty percent of American households and the loss of a pet causes psychological distress among owners. This distress is then typically compounded by the hardships imposed by the encompassing mass emergency event [2, 13]. As a result, our work on EPM is important; it will enable people not directly impacted by
a mass emergency to volunteer their time in helping to track pets and reunite them with their owners, helping to reduce the stress of impacted families while providing digital volunteers with an opportunity to help in a way that has tangible, positive impacts.

EPM is a collaborative and interactive system that brings the digital volunteer community together to report on lost pets and create and vote on their matches. With EPM, a volunteer can propose a match between a lost pet report and a found pet report; this match is then added to the pool of potential matches on EPM that is available for a crowd of volunteers to review. An “upvote” by a volunteer would favor the match whereas a “downvote” would mean the volunteer disagrees with the match. Every match on the system accumulates “upvotes” and “downvotes” until a predefined threshold is met. In particular, a pet match reaches the threshold when there are at least five “upvotes” more than there are “downvotes.” Crossing the threshold indicates that the crowd deems a match as highly probable; at that point, the owners of the pet reports are contacted to verify the success of the match.

The goal of this thesis is to identify user behavior that aids in pet to family reunification and model this user behavior to support effective pet matching on EPM as well as amplify/augment their actions to be even more effective and to encourage them to continue to participate in the system.

To identify those user behaviors that seem to contribute to successful pet matches, EPM would had to have been used for matching lost and found pets by its users and have information about its users’ activities that can be used for analysis. Since EPM has not yet been deployed, we turned to a public page on Facebook called “Hurricane Sandy Lost and Found Pets” that was created by digital volunteers after Hurricane Sandy in late 2012 to work on reuniting lost pets with their owners [10]. Facebook is not designed to support pet reunification but these volunteers adapted
Facebook’s capabilities—photo albums, project pages, and the ability to comment on these items—to allow pet reunification work to occur. Since the activities that occurred on the Sandy’s Pets page is similar to what will occur in EPM, information from this Facebook page was scraped using a tool that was built on the Facebook Graph API. A description of the tool, the functions of the tool and its process flow is provided in chapter 3.

The data from the Facebook page was then analyzed. Patterns and observations from this analysis is discussed in Chapter 4. The behavioral patterns found, were useful while translating the data into EPM activities. This translation is explained in Chapter 5. For instance a user posting a photo of a lost pet was translated into the activity of the user submitting a lost pet report on EPM. The Facebook data was thus used to simulate representative user activity within EPM.

A machine learning model was used to label users based on their activity. The Sandy’s Pets Page data informed this model and this is discussed in detail in Chapter 6. In Chapter 7, the consequences of the machine learning model is addressed. The section describes how the model would affect EPM and how the results of the model can improve the effectiveness of the pet to family reunification process.
Chapter 2

Background

This chapter describes Emergency Pet Matcher (EPM), a crisis informatics application that is being developed to support pet to family reunification. It includes a description of EPM’s methods to track a user’s activity on the system. This section also discusses how EPM’s log files could be used to analyze user behavior to build a model that makes the pet matching process more efficient.

2.1 Emergency Pet Matcher (EPM)

EPM is an interactive web application that enables its users to upload and match pet reports of lost and found pets. EPM encourages collaboration among users to find the best match for a pet by voting on the matches. EPM’s homepage shows the latest pets uploaded to the system and provides users with navigational links related to the various workflows that EPM supports.
2.1.1 Pet Report

A Pet Report is a comprehensive report with information meant to aid EPM users in finding a match for a particular pet. Users can submit pet reports by navigating to the submission form from the homepage of EPM. Users can submit details such as:

- pet type (dog/cat/bird)
- pet status - lost/found
- date when lost or found
- sex of the pet
- location where pet was lost or found
- pet size
- Microchip ID
• tag and collar information
• spayed or neutered information
• pet’s age: baby/young/adult/senior
• coat color of pet
• pet-breed
• an additional pet description
• pet image

Figure 2.2 Screenshot of pet report submission form

This information is to aid EPM users in creating matches between lost and found pets. The EPM homepage shows thumbnails of all the pet reports on the system. When users click on a thumbnail, a dialog page opens up where users can see the pet image along with information related to the pet. The dialog page has buttons that allow users to bookmark the pet report to follow the pet report’s updates (such as a pet match); users can also create a match for the pet by clicking on the match
this pet button. The pet report dialog page also shows a list of matches made for the pet by other users on the system and a list of people working to create matches.

2.1.2 Matching a pet

Users can navigate to the matching workspace by clicking on the “match this pet” button on the pet report dialog page. Here they can choose a match for their target pet from a list of pets. The pets are ordered according to the number of matching attributes. Users can drag and drop a pet to the workspace where the target pet’s information and the chosen pet’s information are displayed side by side. Once the users have decided to make a match they click on “propose match” where the users are taken to a dialog page that requests the users to confirm their match. The pet match is then saved on EPM and is available for all users to vote on.
2.1.3 **Pet Match**

All pet matches for a pet are displayed as thumbnails on the pet report dialog page. Users can see the pet match dialog page when they click on a thumbnail. The match page shows a side by side comparison of the two pets involved in the match. It shows the pet images along with pet descriptions. Users have the option of sharing this match on their Twitter or Facebook feed. Users may vote on the match by clicking on either the “upvote” or the “downvote” button. The match page also shows a list of all other users who have voted on this match.
2.1.4 Successful Matches

When the “upvotes” for a pet match exceeds its “downvotes” by five in number, emails are sent out to the contacts of the lost and found pets informing them of the match. They are encouraged to meet and confirm the match. The users can consequently let EPM know whether or not the match was successful by verifying the match through the links provided in their emails. A successful match triggers EPM to close all other matches for the lost and found pets that were reunited; EPM closes the lost and found pet reports as well so that no new matches can be created for these pets.
2.1.5 EPM Users

Users can register on EPM with their email addresses, Facebook accounts or Twitter accounts. Registered users have profile pages that summarize their activities on EPM. They can see data corresponding to their submitted pet reports, proposed matches, bookmarked pets, and the pets they are working on.

EPM users have the capability of following other users to be updated with their activities. A collection of the latest activities is shown on the activity feed on EPM's homepage. Users get notified when the other users propose matches or upload pet reports. Users also get notified of all matches proposed for their bookmarked pets. Their bookmarked pets can also be viewed on a separate tab on the homepage.

A user can send a message to any other EPM user by clicking on the “send message” link on the recipient’s profile page. This messaging mechanism assures that email addresses of all users are protected, since email addresses are not shown on the profile pages. The message is sent via EPM’s email address to the recipient’s email.
Users get rewarded with reputation points for their activities on EPM. They accumulate points when they submit pet reports, create matches and vote on them. They receive additional points when the match they create or upvote is successful and lose points when their match is deemed unsuccessful. The purpose of this reputation system is to motivate users to continue to contribute to the work of EPM.

Figure 2.7 Screenshot of a Profile page

Figure 2.8 Screenshot of bookmarks Page
2.2 EPM Log Files

EPM operates with an Apache Server [1] and an Nginx proxy [15]. Both these servers maintain log files of all requests sent to the server. The server requests are a record of all EPM pages accessed by the users and all requests sent to the server to submit information such as pet reports and pet matches. Specific user actions such as submitting and bookmarking pet reports, creating and voting on matches and following other users are logged by the EPM server in user-specific activity log files.

2.2.1 Apache Server Log Files

The Apache server log files are generated in a format specified in the apache server configuration file. The configuration currently in place on EPM's apache server is –

```
LogFormat "%h %l %u %t "%r" %>s %b %{X-Remote-User-Name}o %{X-Remote-User-Id}o"
```

Where,

%h – IP address of the client system that made the request

%l – RFC 1413 identity of the client

%u – username recognized by HTTP authentication

%t – timestamp associated with the requested sent to access

%r – request line from the client

>%s – Status Code that server sends back to the client

%{X-Remote-User-Name}o – username as recognized by the Django server

%{X-Remote-User-Id}o – user ID as recognized by the Django server

The following is an instance of a line in EPM's Apache server log file:
2.2.2 User Activity Log Files

The purpose of these activity logs is to populate activity feeds of EPM users. The string in each line of an activity log file follows the following format:

<Timestamp of Activity> <activity> <user-name> <activity related string> <object ID of related object: pet report/pet match>

The string is tailored to each activity. The following is an instance of a user’s activity log. It shows that a user submitted a pet report:

Wed Mar 13 10:31:57 2013 [PETREPORT_SUBMITTED]: user124 submitted the PetReport for {XYZ} with ID{823}

2.3 Analyzing Facebook Behavior to Support matching on EPM

The pet matching process on EPM is dependent on its users’ expertise on choosing the right pet matches to vote on. When a pet match receives “upvotes” such that, it has at least 5 “upvotes” more than its “downvotes,” emails are sent to the contacts of the lost and found pets that a match has been found for their pets. It would be beneficial to EPM and its users if the pet matching process was quicker when users with a higher success rate vote on a pet match. The behavior of users on EPM should be analyzed so that the pet matching process can be quickened depending on the expertise of the users creating or voting on the match. This can be done if EPM user behavior is modeled based on its users’ past successes and the activities of such users in general.

The log files can be used to analyze EPM user behavior. The data contained in log files can be used to identify a sequence of actions taken by a user before matching a pet. This would certainly help in modeling pet matching behavior of users. Since
EPM has not been deployed yet, the data required for this analysis is not available in its log files.

“Hurricane Sandy lost and found pets” is a Facebook page that was created to help re-unite displaced pets with their families during the aftermath of Hurricane Sandy [10]. This page has many active users who work towards reuniting pets with families. It is highly probable that users of the Sandy’s Pets page would likely be future users of EPM. Since the page serves a similar purpose to EPM, activities from the Sandy’s Pets page can be used to infer the types of activities that eventually will populate EPM log files. This is similar to work in [8] that describes the use of a genetic algorithm to generate stock recommendations made by weighting the picks of “the crowd,” where in this particular context, the crowd was represented by financial analysts whose stock picks are available in popular financial newsletters. This work demonstrated that by providing extra weight to the picks of especially successful financial analysts, their algorithm was able to generate stock picks that were, on average, significantly better than any single analyst’s predictions. In this thesis, we attempt to model the behavior of Facebook users performing pet reunification tasks in order to generate realistic behavior for simulated EPM users. We then attempt to generate a model that can identify especially effective pet matching behavior and use that to boost the reputation of EPM users that exhibit that type of behavior.
Chapter 3

The Data Extraction Process

This chapter explains the specifics of the structure of the data being extracted, the tool we engineered to retrieve this data using Facebook’s API, the challenges we encountered while developing this tool and a brief discussion about its evaluation.

The primary goal of the data extraction process is to build a data store, which tracks user activity on the Sandy’s Pets Facebook page [10]. Tracked events include posts about lost/found pets and about pet reunification, suggested matches for pets and adoptions/follow-up on those that are found. The aforementioned tool can pull this data from any public page on Facebook and store it in a SQLite database [19]. Building the tool required an extensive understanding of Facebook’s Graph API and the structuring of their data.

3.1 The Graph API

Facebook gives us access to its data via its “Graph API.”[5] The Graph API is a HTTP-based API that represents all data on Facebook as objects on a social graph. This interface is a means for developers to engineer applications that retrieve data from Facebook as well as post data on to Facebook. To access data using this API, we need to acquire an access token from Facebook that has the necessary permissions. Only basic permissions were required we used since all the information we got from the Sandy’s Pets page is publicly available data. The Graph API returns all its data in the JSON [11] format.

We primarily collected two types of data from a Facebook page – Page Feed and Albums.
The Feed is an array of Post objects on the Facebook page. There are two types of posts relevant to our analysis –

1. *Posts generated by the owner of the Page*: All posts generated by the page admin appear on the forefront of the page and is visible to the user. These posts also appear on the newsfeeds of Facebook users who have liked the page.

2. *Posts by everyone else*: These appear in a small section on the page. Every post posted by anyone other than the page owner goes to this section. These posts are not necessarily seen on the newsfeed of everyone who liked the Facebook page and are generally hard to locate, especially while looking for specific posts. Fig. 3.1 shows a screenshot of user generated posts as seen on the Sandy’s Pets page.

![Figure 3.1 Screenshot of a section of the Sandy’s Pets page where the user generated posts can be viewed](image)

### 3.1.1 The post object

A post object for page-owner posts has the following attributes -

- **Id**: Post ID
• From – Information about creator of Post, this is the page name for this type of posts
  o Id
  o Category
  o name
• Message – Content of the post
• Picture – a link to the picture if available
• Link – a link to the post
• Name – name of the link
• Caption – caption of the link
• Description – description of the link
• Properties – list of properties for an uploaded video
  o Name
  o Text
  o Href
• Icon – icon representing type of post
• Actions – actions associated with the post (usually like and comment)
  o Name
  o Link
• Privacy – privacy settings of the post
• Type – type of the post (photo/status/video)
• Status_type – type of the status: added_photos, wall-post, created_note, wall_post, published_story, etc.
• Object_id – Facebook object id for the associated photo or video
• Created_time – timestamp for when the post was created
• Updated_time – timestamp for when the post was last updated
• Shares – the action representing a user sharing the post on their page or another user’s page
  o Count

• Likes – action representing the user liking the post
  o Data
    ▪ Name
    ▪ Id
  o Count

• Comments – user comment related information in the post
  o Data
    ▪ Id – comment id
    ▪ From – information pertaining to the user who posted the comment
      • Name
      • Id
    ▪ Message – content of the comment
    ▪ Created_time – timestamp for when the comment was created
    ▪ Like_count – number of likes for the comment

• Paging
  o Previous – there’s a previous page available for the data, this attribute will hold a URL for that page
  o Next – if there’s a next page for the data, this attribute will hold the URL for the next page

User posts have all the above-stated information, except for the shares attribute. In addition to the above, they also have the “To” attribute that holds the value of the page that the user posted to.
To

- Data
  - Id
  - Category
  - Name

3.2 Function of the data extraction tool

The data extraction tool performed the following functions

- Return all feed posts, albums and photos with their related comments and likes
- Store unique and accurate copies of this information in the database.

The tool was also able to continue where the extraction process was aborted when the user restarts the tool to continue data extraction for that page. This restart was required when the extraction process exceeded the API’s limits.

3.3 Process flow and description of the tool

The tool was coded in Python. The python library facepy [6], was used to interface with Facebook’s Graph API. Facepy provides easy interaction with Graph API to retrieve an access token as well as retrieve Facebook objects. The process flow shown in Fig. 3.2 is specific to retrieving posts and similar to the process of retrieving albums as well.
3.4 Database schema for stored Facebook Data

A database was created on SQLite with multiple tables that are used to store the data pulled from Facebook. Figures 3.3 and 3.4 show the schema used in creating this database. The schema was designed to facilitate quick analysis of the extracted data.
Figure 3.3 Relational database Schema for the SQLite database showing tables that store albums and photos related information
3.5 Challenges in getting data from Facebook

Initial challenges included finding the best way to extract and store the information retrieved using the graph API. We initially used files to store the JSON data and eventually we found it very cumbersome to extract and analyze JSON data from a text file. We also understood that it would be most effective to analyze the data if it was stored in a relational database. Another notable challenge was pagination on the retrieved data. Also, facepy could only retrieve the first page of a Facebook object. We utilized a python module called “requests” to send the “GET” requests to retrieve the remaining pages.
A limitation with the graph API is that it imposes a restriction such that users can make only 600 API calls during a 10 minute time period using a single access token. Sometimes, this limit was reached before the 10 minute time period ended and necessitated restarts.

There is a lot of data that cannot be retrieved using this API. For example, “shares” is an attribute of the post object where it is only possible to retrieve the number of shares for a post. Since we would only retrieve publically available content, getting information about users sharing a post was not always possible. This limitation arose because the user determines the privacy level of a shared post. While it might be possible to view this data on the Facebook page, it cannot be retrieved using the API.

Facebook data is stored as JSON arrays with a paging variable, to deal with items that overflow an array. A single array returned by Facebook has 25 data objects. Data for “likes” and “comments” have paging as well and hence the likes or comments for a post have to be retrieved separately. This data is retrieved iteratively for each page of likes on a post and therefore can be time consuming. To retrieve more objects we would have to request the link available with the “next” attribute. The data is stored in reverse chronological order and hence the first post retrieved from the page would be the most recent one.

3.6 Evaluation of the tool

The tool was evaluated by using it to retrieve data from other pages similar to the Sandy’s Pets page. One example of a test page used is the Facebook page for Moore Oklahoma Tornado Lost and Found Animals. The successful execution of the data retrieval process was validated by testing its performance with several such pages. The tool’s functionalities were also verified by repeatedly retrieving information from the Sandy’s Pets page at different time periods. The tool’s tolerance
to interruptions (and the consequent duplication of data) was tested by stopping the data extraction and resuming it after a delay. Several manual comparisons of the extracted data and data on the Facebook page were done to verify its authenticity.
Chapter 4

Analysis of the Facebook Data

The current section is a description of the analysis of the Sandy’s Pets data and the findings from that. This data was extracted using the tool we described in Chapter 3 and stored in a SQLite database. The results described here are based on the information stored in this database. A tool called Splunk was used to analyze the textual content of the post and comments. Python scripts were also used to collect some of the statistics. The outcomes of this analysis were used in translating this data to activities seen on EPM.

4.1 Splunk

Splunk [17] is a tool that allows the import and analysis of the data on the SQLite datasets. Splunk was used in this project to facilitate easier viewing and comprehension of the data. Splunk sorts data into events where each event is associated with the timestamp of event creation. The Facebook posts were sorted according to their created-times. Fig 4.1 illustrates how data is structured by Splunk for viewing. Splunk also provides interesting statistics by default such as, the top users who posted comments or posts. These statistics make it easy to visually sift through hundreds of posts quickly. Splunk worked really well for identifying activities that translated to the simulated EPM data, like cases of, identifying matching activities.
4.1.1 Challenges

Splunk DB Connect [18] was used to import the SQLite database into Splunk. However, there were a few problems with Splunk interpreting SQLite’s timestamp format. This constraint was a big hurdle as Splunk indexes all data as events using the timestamp value from every database tuple. This issue was fixed by editing Splunk’s DB’s input settings manually by adding a Format String to represent the timestamp value. Splunk also indexed post data incorrectly as it had carriage returns. Splunk thought this was an indicator of the end of the value for the “post” field.

4.2 Findings in the Sandy’s Pets Data

Hurricane Sandy Lost and Found pet’s page was created on 10/28/2012 [10]. The data we used for the analysis was collected on 05-30-2013. Here are some numbers about the data obtained by querying the SQLite database using SQL.

- 4,421 unique posts.
- 31 duplicate posts that were excluded from analysis.
- 24 albums
• 1,077 photos in total
• 25,405 comments on page feed posts
• 10,774 comments on photos
• 232,410 likes on page feed posts
• 100,878 likes on photos
• 31,153 unique users

While looking at the post data it was clear that the page was most active in November, when 2267 posts were generated. It is obvious after looking at Fig. 4.2 which shows that, there was a high volume of posts on the Sandy’s Pets page during October and November. The posts were about lost and found pets, pets that were available for adoption, pets that had been adopted, pets that needed fostering, pets that were temporarily hosted at shelters, pets that did not survive and posts about pets re-uniting with their family.

![Figure 4.2 Number of posts generated on the FB page distributed by day](image-url)
4.2.1 Lost or Found Pet Posts

The bag-of-words approach [12] was used to find the most frequently occurring words in posts. “Bag of words” represents a collection of words extracted from text and is used to classify the text or the document depending on the words contained in it. The top 10 most frequent words used in posts were as follows. The numbers in parenthesis indicate the number of occurrences of each across all the posts combined.

1. Please (1867)
2. have (1296)
3. found (1204)
4. help (1127)
5. dog (951)
6. her (947)
7. lost (924)
8. home (893)
9. from (868)
10. pets (812)

By looking at the top ten most frequent words we cannot say much about the posts that contain one of those words without knowing the context in which the words were used. Therefore, we further explored all posts by looking at posts with keywords “lost” or “found” or “home.” We formed the following rules to label a post as lost or found –

- All posts that were of type – status were eliminated. These were just status updates generated by the page owner and not posts about lost and found pets. The type of a post is automatically determined by Facebook before it is published on a page.
• Only posts with URL’s were analyzed, because almost all posts that were about lost and found pets contained a URL that linked it to a picture of the pet.

• A post was labeled as lost, if it had the words “lost” or “missing” and did not have the words “returned”, “found”, “finder,” “foster,” “journey,” “adopt,” “adopter,” “adopted,” “adopting,” “home,” “reunited,” “fostered,” “rescue,” “rescued” or “rainbow bridge.”

• A post as labeled “found”, if it had the words “rescued,” ”rescue,” “finder,” “found,” “foster,” “home,” “fostered,” or “shelter” and did not have the words “adopting,” “adopter,” “adopt,” “adopted,” “lost,” “reunited,” “missing,” “journey” or “rainbow bridge.”

• Pets with keyword “adopt” were separated into a category of their own.

• Posts with keywords “reunite” or “journey” were separated into the “re-united/journey” category. These are pets were either re-united with their original families or adopted by a family permanently

• Posts with the keywords “rainbow” and “bridge” were classified under a separate category of pets that did not survive. Rainbow Bridge is a poem that represents pet loss and as a convention many of the posts about pets that did not survive were tagged with “Rainbow Bridge.”

• Posts with no identifiable keywords were classified as unknown.

4.2.1.1 Posts generated by users (non-page-owner)

The users on the Sandy’s Pets page posted a large number of images of lost/found pets. Many users were seen to have posted information about pets from sources such as craigslist and other Facebook pages. Many of these posts were tagged as described above. Figure 4.3 shows the distribution of the posts based on the above stated tags. There were 196 posts about lost pets, 291 posts about found pets, 108
posts that contained craigslist URLs, 62 about pets being adopted or requiring adoption and 15 posts about journeys of pets joining or reuniting with a family.

![User Generated Posts](image)

Figure 4.3 Distribution of the posts generated by users between the categories Lost, Found, craigslist posts, adopted pets, pets that were re-united with families and Unknown posts.

4.2.1.2 Posts generated by page-owner

The page owners took the lost and found pet information from posts generated by the users, formatted the pictures along with their pet information into a poster and posted them back on to the wall of the page. The way Facebook works right now, posts generated by the page-owner usually reach newsfeeds of users who have liked the page. Page owners organized all the pet images into various albums on the Facebook page. The albums on the page as of 5/30/2013 are–

1. DISPLACED Sandy Pets - NEEDING HOMES
2. Timeline Photos
3. Their Journeys
Many albums such as NY - FOUND DOGS or NY - LOST CATS clearly indicate that the contents of albums are found or lost pets respectively, however for albums such as Connecticut • Pennsylvania • Maryland • Massachusetts the pet could be either lost or found. In this case the labelling of the posts into the various categories was done using keywords such as “lost” and “found.” This is similar to the labeling
used for user generated posts. The labeling was verified by manually examining approximately 50% of the posts we identified as either lost or found.

Fig. 4.4 shows a distribution of Facebook posts between its different labels. There were 175 posts about lost pets, 333 posts about found pets, 33 posts with Craigslist URLs, 37 posts with pets that required adoption or pets that were adopted and 107 posts about pets that found homes.

![Posts Generated by Page-Owner](image)

Figure 4.4 Distribution of posts generated by page owner labelled as Lost, Found, craigslist posts, adopted pets, pets that were re-united with families, rainbow bridge pets and Unknown posts.

### 4.2.1.3 Challenges in labelling the posts

Nearly 20% of the posts did not contain any textual description, making it impossible to derive any conclusions. A prominent reason was that many posts were not of lost and found pets.
4.2.2 User activities: Shares, Likes and Comments

Comments along with the user ID’s of the commenters and the timestamps at which the comments were posted were retrieved. Access to the timestamps of “like” actions was not available, however, the names of users that liked a post were all retrieved. Shares for posts generated by the page owner were retrieved, but, no user ID’s of post sharers were available. Counts of the number of shares for every post was available. Table 4.1 shows a comparison of the number of shares, likes and comments per month for all page-owner generated posts. For the purpose of this analysis, we associated shares and likes with the timestamps of the posts associated with those actions.

<table>
<thead>
<tr>
<th>Month</th>
<th>likes</th>
<th>shares</th>
<th>comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oct-12</td>
<td>8182</td>
<td>12335</td>
<td>562</td>
</tr>
<tr>
<td>Nov-12</td>
<td>96912</td>
<td>74414</td>
<td>6559</td>
</tr>
<tr>
<td>Dec-12</td>
<td>39600</td>
<td>32276</td>
<td>4092</td>
</tr>
<tr>
<td>Jan-13</td>
<td>36216</td>
<td>27649</td>
<td>3360</td>
</tr>
<tr>
<td>Feb-13</td>
<td>16325</td>
<td>16651</td>
<td>2003</td>
</tr>
<tr>
<td>Mar-13</td>
<td>9006</td>
<td>5971</td>
<td>1012</td>
</tr>
<tr>
<td>Apr-13</td>
<td>12413</td>
<td>10441</td>
<td>1106</td>
</tr>
<tr>
<td>May-13</td>
<td>19946</td>
<td>17654</td>
<td>1966</td>
</tr>
</tbody>
</table>

Table 4.1 Number of shares, likes and comments per month

Figure 4.5 shows a comparison between shares and likes on posts. The comments from this graph were excluded; as the number of comments per month is significantly lower than number of shares or likes per month. This difference is because commenting on a post on Facebook is marginally more cumbersome than liking or sharing a post on Facebook. Based on the evidence and assumptions a
conclusion that commenting on a post on Facebook is an activity that shows more commitment on the user’s part to the page than just liking the post was reached.

![Number of likes and shares vs. Time](image)

**Figure 4.5** A comparison of the number of likes and shares for every month between Oct-12 and May-13.

### 4.2.3 Patterns in Comments

#### 4.2.3.1 Sharing

An interesting observation by one of our colleagues was that a large number of users commented on posts saying “shared” or “S” notifying the others that they had shared their posts. We used to regular expressions to extract these specific comment instances. We matched the comments with the regular expression [12] “(will share)|(shared)|(sharing)|^s\b” and excluded comments that matched the text “*for sharing*” and regular expression ”(\s)|(s\\ )” because there were many comments that had a variation of the text “thanks for sharing” and these comments did not indicate that particular user sharing that post. We eliminated posts with “‘s” as we wanted to distinguish between the comment “S” for sharing and the punctuation “‘s.” Fig. 4.6 shows the top most seen words used for commenting about sharing the post.
Figure 4.6 Graph showing the frequency of the top 20 comments used to comment about sharing a post

The graph in Fig. 4.7 shows a comparison between the number of shares seen over time, the number of people who commented and the number of comments that indicated sharing. It is clear that the number of comments is far fewer than the number of shares. Also, the comments indicating sharing are a small fraction of the total comments. This may imply that the *commenting about sharing a post on Facebook is an activity that shows more commitment on the user’s part to the page than sharing the post or event commenting on a post.*
Figure 4.7 Comparison of Number of Shares, number of comments and number of comments about sharing
4.2.3.2 Matching

Many instances of matching activity on the Facebook page where people commented on a post of a lost or found pet with a possible match were observed. The comment usually includes a link to the matching pet and a phrase such as “looks like” or “possible match” where the user is alluding to the matching pet. It is clear that matching a pet on Facebook involves users sifting through found dogs in New York if he wants to match one with a Lost Dog in New York. People were seen posting links from various sources such as links from other Facebook pages (for Shelters/Rescues) and even posts from craigslist.

![Figure 4.8 Screenshot of matching activity as seen on the Sandy's Pets page](image)

Fig. 4.9 shows how the matching activity increases from October to January whereas the number of users who authored these comments did not increase drastically over this time period. Among the matching instances that were identified,
there were only about 10 unique users out of thousands who were most actively matching pets.

![Graph showing number of comments with matching activities and number of unique users who authored those comments.](image)

Figure 4.9 Graph showing number of comments with matching activities and number of unique users who authored those comments.

### 4.3 Anomalies in the Facebook data

During the analysis it was found that the comments table had many duplicate comment instances. However, each duplicate comment instance was associated with a different post. A manual look up of the posts on Facebook that were related to the duplicate comments showed that the posts that were duplicated were all about same story but had different post ID's. Every time a collection of photos was added to the album, a new story showed up on the page feed and each of these posts had the same comments and likes as the first instance of the post whereas they all had different post-ids and timestamps associated with them.
Figure 4.9 shows that even though the two posts have different timestamps and ids, the stories are about the same album to which the photos were uploaded and that their associated comments and likes were the same. To resolve this issue in our dataset, only one post instance for every set of duplicates was considered. In the Hurricane Sandy Lost and Found pets page there were 4 stories that had 31 duplicates in total.
Chapter 5

Translation of Facebook data to EPM Data

This section describes the parallels that were identified between activities on the Sandy’s Pets page and those that can be performed by users on EPM. An important step towards translating the Facebook data was the identification of patterns among the different activities that were seen on the Sandy’s Pets page. These activities included users’ attempts at posting pet information, suggesting matches and other forms of users’ involvement on the page. Also presented here is the process through which specific Facebook data, that we identified to be representative of EPM activities, was extracted and how that data was used to simulate EPM activities. The EPM log files, described in Chapter 2, were generated after this transformation.

The specific Facebook activities that we focused on were:

- Posting on the page feed
- Liking a Post
- Commenting on a Post
- Attributes and Patterns among comments
  - Commenting about possible pet matches
  - Commenting about sharing a post

These above-stated activities were translated to the following EPM activities:

- Submitting a Pet Report
- Viewing a Pet Report
- Bookmarking a Pet Report
• Viewing a Pet Match
• Creating a Pet Match
• Voting on a Pet Match

5.1 Protecting User Privacy

Though only publicly available information from the Sandy's Pets page was obtained, all user related comments and posts were associated with a user id and the user name. Even though this information cannot be used to obtain further information about the user through the Graph API, the Facebook user information was separated from the EPM data analysis. Therefore, every post, user and comment from the Facebook data were mapped to numbers from an integer sequence to differentiate between EPM data and Facebook data.

5.1 Generating EPM Log File data

The activities identified from the Facebook data were converted to EPM data and stored in the users’ activity log files and the Apache Server log files. The purpose of generating data in these log files is to analyze this log file data and derive conclusions about how this sort of log file analysis can help in supporting user’s performances on EPM.

All the translated activities for viewing pet reports and matches were logged on the Apache Server log files. Instances of pet match creation were logged on to users’ activity log files as well as the apache server log file. Activities for voting on pet matches and bookmarking pet reports were also logged in the user specific activity log files.

5.2 Pets

In Chapter 4, we discussed about how some posts represented lost and found pets. We identified 741 such posts and marked them all as Pets. Figure 5.1 shows an
example of a found Cat post on the Hurricane Sandy Lost and Found Pets page. This is similar to a Pet Report on EPM.

![Screenshot of a post about a Found Cat on the Sandy's Pets page](image)

Figure 5.1 Screenshot of a post about a Found Cat on the Sandy's Pets page

5.3 Pet report submissions

595 of the user-generated posts were labeled as lost and found. These posts were not always broadcasted to the Facebook users who liked the Sandy’s Pets page. Therefore, the page administrators converted each post to a poster with all the information pertaining to that pet and posted it on the wall of their Facebook page. Unfortunately, matching a post from the users’ posts section to its corresponding poster generated by the page owner was extremely difficult. This problem arose because the textual description of a pet that was part of its poster created by the page owner was not comparable to the user-posted content. Moreover, this matching was not required for the analysis of user activities on the comment streams. Only the comment streams on the posts generated by the page owner were analyzed. Therefore, the association of user postings of pet information with their activity on the comments stream was sufficient.
User’s action of posting pet information on Facebook was congruent with the action of submitting a new pet report on EPM. While submitting a pet report, the user is required to fill out details about a lost/found animal on a form to create a pet report. Submitting a pet report sends a “POST request” to the server to save the pet report. This action gets logged to the user’s activity logs on the EPM server. Fig. 5.2 shows an example of one such activity log for user553 submitting a pet report with ID {287}.

![Screenshot of user553's activity log showing the user's pet report submission activity](image)

**Figure 5.2** Screenshot of user553's activity log showing the user's pet report submission activity

### 5.3.1 Generating data for Pet Report Submissions

To generate activity logs showing actions of submitting pet reports, three categories of post labels that were identified were considered—Lost pet Posts, Found pet posts and Craigslist posts. Each of these posts were matched randomly with a post generated by a page owner. Users who posted craigslist links could have posted about either types of post, lost or found pets. The pet status – lost or found was not significant in our analysis of the user behavior. Hence, this labeling was randomized to be either lost or found. We matched 196 lost pets, 291 found pets and 108 craigslist posts from the user generated posts with 126 lost pets and 479 found pets posts generated by the page owner. This activity was time-stamped with the time of creation of the user's post. 525 instances of pet report submissions were generated.

### 5.4 Viewing Pet Reports

The posts generated by the page owner of the Sandy’s Pets page reached the newsfeeds of users who liked the page. If a user has liked a post, it clearly indicates that the user has seen that post.
Every time a user views a pet report on EPM, a “GET” request is sent to the EPM server to retrieve the pet report for the user to see. These “GET” requests are logged on EPM’s Apache server logs. Fig. 5.2 shows an instance of such a request.

Thus, the likes on pet posts to EPM were translated to “GET” requests that indicate pet report views.

Figure 5.3 Screenshot of Apache server logs showing "GET" requests indicating pet report views

5.4.1 Generating data for Pet Report Views

The User ID’s of users who had “liked” the labeled posts were extracted. These ID’s were then used to generate a string to add to the log file with its time-stamp to match the created time of the post. The reason behind using the time-stamp of the post is that since the time stamps for likes on posts were not available, the post’s time-stamp was used instead. People usually “like” posts when they see a post, often times on their own newsfeed. A post appears on someone’s newsfeed along with other activities that occurred in a similar time frame, it is unlikely that a user has scrolled through her newsfeed through many days’ worth activities to find a single post. Therefore, it seemed fair to assume that the timestamp of a post’s creation would not be too far off from actual timestamp for the likes.

5.5 Commenting on a Facebook Post and Bookmarking on EPM

Commenting on a post automatically allows users to get notifications about any and all future comments to that post. On the Sandy’s Pets page, people commented on posts to express their concerns for the pets, to let others know that they shared the post with their friends on Facebook and to try and find matches for pets, among other reasons. If the page owner or pet owner comment on a post with
the intention of updating others about the status of a pet, other commenters would also be informed of all future updates about a pet’s well-being.

When a user views a pet report on EPM, she has the option to bookmark the pet to receive further updates about the pet. Updates include notifications about any and all pet matches created for the pet and about a successful pet match for that pet.

Therefore both actions – commenting on a post on the Sandy’s Pets page and bookmarking pet reports on EPM serve similar purposes. Hence, commenting on Facebook was translated to bookmarking on EPM.

5.5.1 Generating data for bookmarking a pet

When an EPM user bookmarks a pet report, it gets logged on to his activity log file along with the timestamp of the bookmark and the id of the pet report he bookmarked. Fig. 5.4 shows an instance of a bookmark logged on the activity log file of a user.

![Screenshot of activity log of a user who bookmarked a pet report](image)

Figure 5.4 Screenshot of activity log of a user who bookmarked a pet report

All comments on posts that were labeled as lost and found pets were extracted. These comment instances were then converted to an action of bookmarking a pet as per the format used in user activity logs. 20,660 instances of bookmarking were thus generated.

5.6 Creating and Voting on Pet Matches.
The two specific comment patterns found during the analysis were used in the translation. These patterns were:

- Users suggesting matches for pets via comments
- Users commenting about sharing a post
The analysis of both the above activities was explained in Chapter 4.

5.6.1 Suggesting Matches via comments – Creating Pet Matches

10 users were found to suggest possible matches for pets on the Sandy’s Pets page. These users commented with a URL of a pet that’s a possible match. The match was not always from the Sandy’s Pets page. People commented with craigslist URLs and URLs pointing to pets on other Facebook pages as well. Therefore, it was impossible to identify the pet that the post was matched with. Due to this, it was only possible to identify the presence of matching activities. It was impossible to get the complete information about the matches.

On EPM, matching is done explicitly on the matching workspace where a user can select a pet to match and choose another pet to match it with, from a selection of pets available on the same page. The effort involved in creating matches on EPM is probably less compared to the effort required to find and suggest matches on Facebook. EPM data was generated by translating this matching activity on Facebook to simulate matching on EPM.

5.6.1.1 Generating EPM data for Matches

Comments that contained matching activities were identified by using specific keywords on Splunk. Only comments that contained URLs and these keywords, were analyzed, to further narrow down the number of events. The search strings we used were:

\[\text{comment=\"*is this*\" OR comment=\"*match*\" OR comment=\"*craigslist*\"}\]

The comment-timestamp and comment-author were used to quantify this activity to convert them to an EPM activity.

When a match is created on EPM, a “POST” request is sent to the EPM server. This request is logged on EPM’s Apache server logs with the timestamp of match creation,
the user information of match creator and id’s of pet reports involved in the match. An example of the activity recorded on EPM’s Apache server log is shown in Fig. 5.5. The pet match creation is also logged on the pet match creator’s activity logs as shown in Fig 5.6.

Figure 5.5 Apache Server Log instance of a match proposed by user398 between pet reports with ID’s 815 and 666

Figure 5.6 Screenshot of user14076’s Activity Log showing the user's action of proposing a petmatch with ID 127

Since this information is not available for both pets associated with a pet match (from Facebook), we used a fictional pet report for the second pet associated with the pet match. A pet report object was created by identifying the pet report with the comment instance that its activity was translated from. This object was created to assure uniqueness of every pet match created. We generated data for 130 matches.

5.6.2 Viewing Pet Matches

When a user comments on a Post with a pet match suggestion, other users who previously commented on the same post get notified about this comment. Users who comment on the post after this comment are also able to view this match suggestion. For the purposes of simulating EPM activities, it was assumed that both of the above groups of users view the comment. In reality, it may not true that they viewed the match suggestions. Since access to information about the users who liked the comment with the match suggestion was not available, we were unable to ascertain pet match views with better certainty.
Every time a user views a pet match on EPM, a “GET” request was sent to the server to retrieve the pet match. This request is logged on the apache server log with the ID of the Pet Match that was viewed. An example of this is shown in Fig. 5.7.

![Example log entry](image.png)

Figure 5.7 Screenshot of Apache server log showing a record of “GET” request when user398 viewed Pet Match with ID 216

### 5.6.2.1 Generating Data for Pet Match Views

For every comment that was a pet match suggestion, all other comments pertaining to the same post as the match suggestion, were extracted. A log-string (similar to the one shown in Fig. 5.7), that indicated that every user who commented on a post with a match would have viewed the Pet Match, was generated. The time of the viewing action was unknown. Therefore, the timestamp of the user’s comment after a match suggestion, was used as the time stamp for this action. If the user had commented before the match was suggested, the timestamp of the match suggestion was stored as the timestamp of the viewing action. About 3826 instances of viewing pet matches were generated.

### 5.6.3 Voting on Pet Matches

When a user views a Pet match on EPM, he/she have the option to either “upvote” or “downvote” the pet match. The vote is sent to EPM’s server as part of a post request. This activity can be identified by looking at the user’s activity log file. The activity is logged along with the ID of the pet match that user voted for.

Unfortunately, no voting activity on Facebook was found, as users did not have any explicit means of expressing their opinions on Pet Matches.

The “upvotes” and “downvotes” for pet matches were randomly generated. However, it was important to quantify the votes with information about specific users.
who were more actively involved on the Sandy’s Pets page than some others. Data about users who liked a post and commented on the post saying they shared the post, was retrieved. This indicates that they have liked a post and shared it as well as commented on it. To vote on a pet match on EPM a user would have to view a pet report and then view a pet match of a pet report. These two sequences of activities are in no way similar however, both indicate a deeper user interest on a single pet.

There were 26571 who liked at least one post on the Sandy’s Pets page. Out of that, 2790 users commented on the same posts they liked. 234 of those users commented about sharing the same post.

We generated data for 699 voting instances.

5.7 Challenges
Many of the activities were found using Splunk. The ID’s of the activity instances from Splunk were exported. Python scripts were run to convert these instances to complete instances with timestamps that can be used to generate log data. This was challenging and it delayed the process of generating and analyzing EPM data.

5.8 EPM data
As a result of translation of Facebook activities to EPM data, an apache log file with 93244 lines and 3606 activity log files of users was generated. The analysis of this data will be discussed in Chapter 6.
Chapter 6

Analysis of EPM Data

The current chapter is a description of the analysis done on the data generated by translating Facebook activities to EPM logs. The following sections explain which features were extracted, how they were extracted and then used to train a classifier. It also explains challenges encountered because we are analyzing simulated data and not real EPM log data.

6.1 Extracting Data from EPM log files

Data was extracted from log files and stored in a SQLite database to facilitate extraction of features from the data. Regular expression patterns were used to match the strings in the log files and each line on the log file was separated out to the different columns of the table.

The Apache Log files and the User activity log files used different regular expressions

- A line from the Apache log file matched the regular expression pattern –
  \(d+\.d+\.d+\.d+ \cdot [(f^\{|\}|:d+:d+:d+:d+) \cdot +(d{4})] (\cdot \cdot \cdot d{3}) (d{1,10})\)

- A line from the activity log file matched the regular expression pattern –
  \(([w3]\ \w2 \ \d2 \ \d2 \ \d2] \backslash (d{4}) \backslash (d+)] \ \backslash (\cdot \cdot \cdot w+) \backslash w+ .+ ID \backslash (d+)\)

Fig. 6.1 shows the schema of the tables that were used to store this data. Log strings indicating pet match creation are present on the user activity logs along with the pet match ID and on the apache server logs along with the ID’s of the pet reports that were matched. Therefore, pet match specific log values extracted from the Apache Server log files were stored in the logs_petmatch table.
6.2 Extraction of Features

Fifteen features were extracted from the EPM data. Each feature that was extracted, describes how active a user was on EPM. The features that were extracted are as follows –

- Number of pet reports submitted by a user
- Number of pet reports bookmarked by a user
- Number of pet matches viewed by a user
- Number of pet matches proposed by a user
- Number of pet matches the user has voted on
- Mean and standard deviation of a user’s time duration between any two actions on EPM
- Mean and standard deviation of time duration between any two pet matches proposed by a user
- Mean and standard deviation of the number of pet reports bookmarked by a user on a single day
- Mean and standard deviation of the ratio between the number of pet reports viewed by a user in a day to the number of pet reports generated that day
• Mean and standard deviation of the ratio between the number of pet reports bookmarked by a user in a day to the Number of pet reports generated that day

The decision to extract the above features was made based on the different activities that were available to be extracted from the log files. The activities related to the extracted features were grounded on the knowledge derived from the data on the Sandy’s Pets page on Facebook.

6.2.1 Challenges in Extracting Features

There were other features that would have been helpful in describing a user’s activity and effectiveness on EPM, such as –

• Number of successful matches
• Number of unsuccessful matches
• Number of “upvotes” on successful matches
• Number of “upvotes” on unsuccessful matches
• Number of “downvotes” on successful matches
• Number of “downvotes” on unsuccessful matches
• Mean and standard deviation of a user’s time duration between bookmarking a pet & creating a match for it
• Mean and standard deviation of time duration between a user creating a pet match and the match being rendered successful
• Number of times that a user viewed both pet reports before creating a match between them.
• Number of pet matches viewed for a single pet before voting on its matches
• Number of pet matches that a user viewed for a single pet report before creating a match for that pet
- Number of pet reports viewed by a user between navigating to the matching workspace to match a pet and creating a match for that pet

Even though the Facebook Data had parallels to many of the activities seen on EPM, it had no explicit indications of some important EPM actions such as creating and voting on pet matches. Since EPM’s tasks are centered around finding pet matches the features extracted from simulated EPM activities are not sufficient to build a model to classify users as effective matchers. However, the data was used to build a model to identify highly active users. *Highly active users spend a lot of time and energy on EPM and they are very invested in the well-being of pets displaced from homes.*

### 6.3 Classifying the Data

A supervised learning approach [7] was used to train a model to classify EPM users. A training set was constructed using the data, to classify the users as “highly active” and “moderately active” (not highly active).

#### 6.3.1 Constructing a training set

Users from the Facebook data were ranked based on their count of actions i.e., a total of their likes, comments and posts on the Facebook page. We found that there were 31153 unique users and the sum total of user-actions ranged from 1 to 1364.

The users were then ranked based on only their posting and commenting actions. There were 8936 unique users and their total actions ranged from 1 to 567. The number of actions of the Top 1% of these users ranged between 26 and 567. These users’ activities were used towards building the training and testing dataset.

Approximately 30% of the top 1% users had at least 50 actions in total. This was decided as a cut off for labelling a user as “highly active”. The rest of the users
among the top 1% were not considered to be highly active. The training data was then built for these users by extracting the feature-set for each user.

### 6.3.2 Choosing classifiers

The data was tested with two classifiers, a Naïve Bayes Classifier [7] and a Decision Tree [7] classifier. These classifiers were implemented by the NLTK library for python [14].

The Decision Tree classifier was trained with 80% of the data and tested with the remaining 20%. The accuracy of this classifier proved to be very low at 66.67%. Table 6.1 shows a confusion matrix [3] for this model.

<table>
<thead>
<tr>
<th>Actual Class</th>
<th>Predicted Class</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Highly Active</td>
</tr>
<tr>
<td>Highly Active</td>
<td>0</td>
</tr>
<tr>
<td>Not Highly Active (Moderately Active)</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 6.1 Confusion matrix for the model built using a Decision Tree Classifier

A Naïve Bayes classifier was then trained with 80% of the data and tested with 20%. The accuracy [3] of this model was seen as 88.89%. When the classifier was trained with only 70% of the data (and tested with 30%), the accuracy dropped to 80.77%. Therefore the model that performed best was the Naïve Bayes Classifier trained with 80% of the data. A confusion matrix for this model is shown in Fig. 6.2.

This classifier was tested with one highly active user that was not included in the training and testing data set, this was the page-owner of the Facebook page. The classifier labeled this user as “highly active.” The classifier also labeled six randomly
chosen “moderately active” users correctly. This sensitivity [3] of this model is 66.67%, however both its specificity [3] and precision [3] are 100%. The specificity illustrates how correctly the negatives were labeled and the precision indicates that all values labeled positively i.e., as “highly active” were labeled correctly.

<table>
<thead>
<tr>
<th>Actual Class</th>
<th>Predicted Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Highly Active</td>
<td>Highly Active</td>
</tr>
<tr>
<td>Not Highly Active (Moderately Active)</td>
<td>Not Highly Active (Moderately Active)</td>
</tr>
<tr>
<td>Highly Active</td>
<td>4</td>
</tr>
<tr>
<td>Not Highly Active (Moderately Active)</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>12</td>
</tr>
</tbody>
</table>

Table 6.2 Confusion Matrix for the model built on a Naive Bayes Classifier

6.3.3 Challenges in Building a Model to Classify EPM Users

While the model worked really well for the data set chosen, this was a really small data set with only 28 users labeled as “highly active” and 61 users labeled as “moderately active.” To model all EPM users accurately, a much larger dataset would be required. However for the current EPM dataset containing data translated from a Facebook page, this model was fairly accurate since there were no false negatives in the test set (as shown in Table. 6.2).
Chapter 7

Conclusion

This chapter provides a summary of all previous chapters. It contains a discussion that explains the initial goal of the thesis and how there were various challenges throughout the project that changed the goal of the thesis by a considerable margin. There is also a description of future work that can be done to continue research on modeling EPM users.

7.1 Summary

Chapter 2 described Emergency Pet Matcher, a crisis informatics application which enables digital volunteers to take part in submitting lost and found pet information and creating and voting on matches between pets. It is clear why an understanding of EPM users is required to make the pet matching process more effective. EPM user behavior can be analyzed using its users’ activity log files and the Apache Server log file. However, real time EPM user data was not available as the system has not yet been deployed. We therefore extracted data from the “Hurricane Sandy Lost and Found Pets” Facebook page. Chapter 3 explained how data was extracted from the Facebook page using a custom built tool.

Chapter 4 presented a detailed analysis of the data extracted from the Facebook page and the activities that were seen on the page. The posts on the page were labeled as lost or found and the comments of these posts were analyzed. The comments were seen to display patterns of user behavior such as users empathizing with lost pets, suggesting matches for pets, and commenting that they shared the posts. Chapter 5 tied these activities to EPM and presented parallels between the data. This helped in translating the Facebook data to EPM simulations. Chapter 6
explained the modeling of EPM users using a Naïve Bayes Classifier trained on the EPM data.

7.2 Discussion

The initial goal of the thesis was to identify effective user behavior that contributes towards successful pet matches. Unfortunately, we were unable to extract activities associated with successful pet matches from the Facebook page. This was attributed to the fact that there was no explicit way to create matches on the Facebook page. Since this was not an obvious activity that people could take part in, there were very few users seen doing matching activities. These matching activities were not enough to model the matching behavior of users on EPM.

On a positive note, there were a lot of activities observed on the Sandy’s Pets Facebook page that were indicators of very active users who performed a lot of actions surrounding pet reunification on the Facebook page. A model was built to classify and identify those users, who took the lead in performing several activities geared towards pet to family reunification. It is highly probable that these users would be effective in finding matches for pets on EPM.

To support these active users on EPM, users who engage in more effective behaviors could be rewarded with additional reputation points. This could also be incorporated into the threshold for a pet match. For instance, a user with greater social capital, as identified by the classifier, will have more weight given to their votes on pet matches. This could reduce the time taken for a pet match to reach its threshold of votes.

On a different note, we observed a lot of activity on the Sandy’s Pets Facebook page in relation to information posted about pets that required foster care or adoption. This activity seemed more prominent than the matching activity that was
seen on the comment threads. 62 user generated posts were identified to be related to pets being adopted or being put up for adoption. There were many posts about Found Pets that required Fostering. It is clear that the page owners of the Sandy’s Pets page had a larger picture in mind – finding homes for all pets displaced during Hurricane Sandy. Their focus was not just matching the found pets with people who lost their pets. This indicates a possible direction for a change in Emergency Pet Matcher should the need arise for this system to support pet information for pets that require fostering as well.

7.3 Contributions

The contributions of this thesis include the tool developed to extract posts and photos from a Facebook page. This tool is structured to pull data from a public page on Facebook to a SQLite database. The database makes it easy for users to retrieve simple statistics about the data, with ease. It was inferred from the data that many users of the “Hurricane Sandy Lost and Found Pets” Facebook page were eager to like and share the posts from there. However, only a third of that number participated in commenting and a fifth of the number of commenters participated in posting. The analysis of the data retrieved from this page was helpful in further understanding the behavior of those users in the process of pet reunification. It also helped in associating these activities with actions on EPM. This further validated the various functionalities of EPM that would make it easier for users to take part in pet reunification by giving them explicit means to create matches.

The contributions of this work also include the EPM data simulations that were done using the Facebook data. The simulated data played a key role in the model that was built to classify EPM users, based on their actions on EPM. This classifier is an example of how EPM user data can be used to inform the pet matching system about the users that are creating and voting on the matches. The users labeled as
active would receive a higher number of reputation points for their actions on EPM. Users' reputation points would act as a weight while calculating a “score” for a pet match. This score is a function of its votes and voters. This score can be compared with a set threshold for pet matches. Comparison of the score with the threshold would determine whether or not to inform the contacts of the pets involved in a match. This classifier would perform more accurately with the addition of features based on the real time activity of EPM users.

7.4 Future Work

Once EPM is deployed, the log data can be further analyzed to produce more features about users. Many of these features were elucidated in Chapter 6 and can be used to model user behavior more accurately. Any data available about users’ past successes or failures in matching pets would also be very beneficial in building this model.

The model used in this thesis can be used as a stepping stone to build a model using EPM users alone. The current model can identify active users from real time EPM user data. These active users can be further used to build a model using EPM data. To integrate this with EPM once it has been deployed, the classifier's inputs should be configured to automatically use EPM's log files and it should be configured to output labels for its users in a database that can be accessed by EPM's reputation point system. The reputation point system should be modified to reflect user labels identified by the classifier. The pet matching process should consequently be changed to be a function of the votes as well as the voters’ reputation points. It can be set up so that the classifier runs on a periodical basis to classify EPM users. Also, once EPM has an established set of expert users (users with a high success rate), these users can be added to train the classifier model to further optimize it.
In addition to the above, it should be possible to identify the behaviors of malicious users by identifying activities that do not contribute to successful pet matches and to automatically reduce their social capital and thus limit the effects of their behavior within the system.

On EPM proposing and “upvoting” matches is an easy task; this works great for people who are genuinely interested in reuniting pets. However, the website is open game for anyone who registers and features on EPM are open to exploitation by spammers. Therefore, this user behavior analysis can be used in an effort to reward active and effective users and to dissuade the malicious. By rewarding members who seem more effective than average we attempt to elevate the wisdom of the crowd ultimately helping families and pets impacted by mass emergencies.
Bibliography


[19] SQLite database [www.sqlite.org](http://www.sqlite.org)


[21] Twitter [https://www.twitter.com](https://www.twitter.com)
APPENDIX

A PYTHON CODE FOR TOOL DEVELOPED TO EXTRACT DATA FROM A FACEBOOK PAGE

#python libraries
import sys, time
from datetime import datetime
#third party libraries
import sqlite3 as sql
import requests, simplejson
from facepy import GraphAPI, get_application_access_token

#GLOBAL VARIABLES
ACCESS_TOKEN = get_application_access_token('168352736667694','1438d29eb39a5c6fd190ca8c9bdef97f')
page_name = raw_input("Enter the Facebook page name :")
DB_NAME = raw_input("Enter SQLite DB name: ")
num_userlikes = 1
num_posts = 0
num_albumlikes = 0
list_commentids = []

def getDBConnection(DB_NAME):
    try:
        con = sql.connect(DB_NAME)
        cur = con.cursor()
        return cur, con
    except sql.Error, e:
        print "Error %s:" % e.args[0]
sys.exit(1)

def insertPhotoFromAlbum(DB_NAME, album_id, album_name, photo_id):
    global ACCESS_TOKEN
    global num_userlikes
    photo_desc = ""
    source_link=""
    picture_link =""
    link ="
    likes_count = 0
    shares_count =0

graph = GraphAPI(ACCESS_TOKEN)
photo = graph.get(photo_id)

(cur,con) = getDBConnection(DB_NAME)
if "from" in photo.keys():
    author = [photo["from"]["id"],photo["from"]["name"]]

if "name" in photo.keys():
    photo_desc = photo["name"]

if "picture" in photo.keys():
    picture_link = photo["picture"]

if "source" in photo.keys():
    source_link = photo["source"]

if "link" in photo.keys():
    link = photo["link"]

created_time = getTime(photo["created_time"])  
updated_time = getTime(photo["updated_time"])  

likes_list = getAllInstancesOf(photo_id, "likes", type="photo")

likes_count = len(likes_list)

    cur.execute('INSERT INTO photos_info 
values(?,?,?,?,?,?,?,?,?,?,?,?,?),
[photo_id, album_id, album_name, 
photo_desc, created_time, updated_time, author[1], author[0], picture_link, 
source_link, link,likes_count,shares_count])

    print 'INSERT INTO photos_info(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)' 
%(photo_id, album_id, album_name, photo_desc,
created_time, updated_time, author[1], author[0], picture_link, source_link, 
link,str(likes_count),str(shares_count))

    if likes_count > 0:
        for like in likes_list:
            cur.execute('insert into user_likes_photos 
values(?,?,?,?)', [num_userlikes, like["id"], like["name"], photo_id])

            print 'inserted into the user-likes-photos table [%s, %s, %s]' 
% (like["id"], like["name"], photo_id)

            num_userlikes += 1

if "comments" in photo.keys():
    comments_list = 
    getAllInstancesOf(photo_id, "comments", type="photo")

    if comments_list != None:
        for comment in comments_list:
            cur.execute('insert into user_comments_photos 
values(?,?,?,?,?,?,?)', [comment["id"], comment["message"], comment["from"]['id'], comment["from"]['name'],
comment["like_count"],
getTime(comment["created_time"], photo_id))

            print 'insert into the user_comments_photos table [%s, %s, %s]' 
% (comment["id"], comment["message"], comment["from"]['id'], comment["from"]['name'],
comment["like_count"],
photo_id, getTime(comment["created_time"]))

con.commit()
def insertAlbumInfo(album_id, DB_NAME):
    global ACCESS_TOKEN

    graph = GraphAPI(ACCESS_TOKEN)
    album = graph.get(album_id)
    if "name" in album:
        album_name = album["name"]
    else:
        name = ""
    if "from" in album:
        author_id = album["from"]["id"]
        author_name = album["from"]["name"]
    else:
        author_id = ""
        author_name = ""
    if "description" in album:
        album_desc = album["description"]
    else:
        album_desc = ""
    if "link" in album:
        link = album["link"]
    else:
        link = ""
    if "cover_photo" in album:
        cover_photo_id = album["cover_photo"]
    else:
        cover_photo_id = ""
    if "count" in album:
        photo_count = album["count"]
    else:
        photo_count = 0
    if "type" in album:
        album_type = album["type"]
    else:
        album_type = ""
    if "created_time" in album:
        created_time = getTime(album["created_time"])
    else:
        created_time = ""
    if "updated_time" in album:
        updated_time = album["updated_time"]
    if "likes" in album:
        likes = getAllInstancesOf(album_id,"likes",type="album")
    else:
        likes = []
    if "comments" in album:
        comments = getAllInstancesOf(album_id,"comments",type="album")
    else:
        comments = []
    like_count = len(likes)
comment_count = len(comments)

(cur,con) = getDBConnection(DB_NAME)

#insert album info into album info
cur.execute("insert into albums_info
values(?,?,?,?,?,?,?,?,?,?,?,?,?),%s",
(album_id,album_name,album_desc,author_id,
author_name,link,cover_photo_id,photo_count,album_type,created_time,updated_time,
like_count,comment_count)

print "insert into albums_info
values(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)"
%(album_id,album_name,album_desc,str(author_id),
author_name,link,str(cover_photo_id),str(photo_count),album_type,created_time,
updated_time,str(like_count),str(comment_count))

global num_albumlikes

if like_count >0 :
    for like in likes:
        cur.execute("insert into albums_likes
values(?,?,?,?)",[num_albumlikes,like["id"],like["name"],album_id])
        print 'inserted into the albums_likes table [%d, %s, %s,
albums["data"]])
        num_albumlikes += 1

if comment_count>0:
    for comment in comments:
        cur.execute("insert into albums_comments
values(?,?,?,?)",[comment["id"],

comment["message"],comment["from"]["id"],comment["from"]["name"],
comment["like_count"],

album_id, getTime(comment["created_time"])])
        print 'insert into the albums_comments table [%s, %s, %s,

comment["message"],comment["from"]["id"],comment["from"]["name"],
comment["like_count"],

album_id, getTime(comment["created_time"])])

con.commit()
con.close()
for element in albums:
    album_id = element['id']
    album_name = element['name']
    if "Timeline Photos" in album_name:
        continue
    if album_id not in existing_albums:
        insertAlbumInfo(album_id, DB_NAME)
    photos = getPhotoIds(album_id)
    (cur, con) = getDBConnection(DB_NAME)
    for photo in photos:
        if photo not in existing_photos:
            insertPhotoFromAlbum(DB_NAME, album_id, album_name, photo)

def getPhotoIds(album_id):
    global ACCESS_TOKEN

    graph = GraphAPI(ACCESS_TOKEN)
    photos_list = []
    photos = graph.get(album_id + '/photos')
    photos_list = list(element['id'] for element in photos['data'])
    if 'next' in photos['paging'].keys():
        next_page = photos['paging']['next']
    else:
        next_page = ''
    while (next_page != ''):
        res = requests.get(next_page)  # edit next_page
        if res.status_code == 200:
            photos_list = photos_list + list(element['id'] for element in res.json()['data'])
        try:
            if 'next' in res.json()['paging'].keys():
                next_page = res.json()['paging']['next']
            else:
                next_page = ''
        except:
            if 'paging' not in res.json().keys():
                print res.json()
                next_page = ''

    return photos_list

def getObjectsList(DB_NAME, col, table):

    con = None
    postid_list = []

    try:
        (cur, con) = getDBConnection(DB_NAME)
        sql_string = 'SELECT DISTINCT "+col+" from "+table+";'
        cur.execute(sql_string)
        objectid_list = [tuple[0] for tuple in cur]
    except sql.Error, e:
        print "Error %s: %s.args[0]" % (e)
        sys.exit(1)

    finally:
if con:
    con.close()
return objectid_list

def getPageFeed(page_name,DB_NAME):
global ACCESS_TOKEN
global list_commentids

graph = GraphAPI(ACCESS_TOKEN)
response = graph.get(page_name+'/?fields=feed.fields(from)')
next_page  = response['feed']['paging']['next']
response = response['feed']
allposts = getObjectsList(DB_NAME,"postid","post_info")
list_commentids = getObjectsList(DB_NAME,"commentid","user_comments")
while next_page != "":
    for element in response['data']:
        postid = element['id']
        if postid not in allposts:
            post_authorid = element['from']['id']
            insertPostinDB(graph.get(postid),DB_NAME)
response = requests.get(next_page)
if response.status_code == 200:
    try:
        #if 'next' in response.json()['paging'].keys():
            next_page = response.json()['paging']['next']
    except:
        next_page = ""
        print 'response: '+str(response.json())
response = response.json()

#this is to be deprecated

def getNewPosts(PAGE_NAME,DB_NAME):
global ACCESS_TOKEN

graph = GraphAPI(ACCESS_TOKEN)

allposts = getPostsList(DB_NAME)
response = graph.get(PAGE_NAME+'/?fields=posts.fields(id)')
next_page  = response['posts']['paging']['next']
response = response['posts']
while next_page != "":
    for element in response['data']:
        postid = element['id']
        if postid not in allposts:
            insertPostinDB(graph.get(postid),DB_NAME)
response = requests.get(next_page)
if response.status_code == 200:
    try:
        #if 'next' in response.json()['paging'].keys():
            next_page = response.json()['paging']['next']
    except:
        next_page = ""
        print 'response: '+str(response.json())
response = response.json()

def insertPostinDB(post,DB_NAME):
global num_userlikes
global num_posts
global num_userlikes_userpost
global list_commentids

postid = post["id"]
num_posts += 1
if "from" in post.keys():
    author = [post["from"]["id"], post["from"]["name"]]
    # if author not in post_authors:
    #    post_authors.append(author)
else:
    author = ["", "]
if "message" in post.keys():
    message = post["message"]
else:
    message = ""
if "shares" in post.keys():
    shares_count = post["shares"]["count"]
else:
    shares_count = 0
if "likes" in post.keys():
    likes_count = post["likes"]["count"]
else:
    likes_count = 0
if 'link' in post.keys():
    link = post["link"]
else:
    link = ""

(cur, con) = getDBConnection(DB_NAME)
cur.execute('insert into post_info
    values(?,?,?,?,?,?,?,?,?,?)', [postid, message, author[0],
author[1], shares_count, likes_count, post["type"], link, getTime(post["created_time"]),
getTime(post["updated_time"])]
print 'inserted into post_info table [ %s, %s, %s, %s, %d, %d, %s, %s, %s, %s]' % (postid, message, author[0],
author[1], shares_count, likes_count, post["type"], link, post["created_time"], post["updated_time"])

if "likes" in post.keys():
    likes_list = getAllInstancesOf(postid, "likes")
    if likes_list != None:
        for like in likes_list:
            cur.execute('insert into user_likes
values(?,?,?,?)', [num_userlikes, like["id"], like["name"], postid])
    print 'inserted into the user_likes table [ %s, %s, %s]' % (like["id"], like["name"], postid)
    num_userlikes += 1

if "comments" in post.keys():
comments_list = getAllInstancesOf(postid, "comments")

if comments_list != None:
    for comment in comments_list:
        cur.execute('insert into user_comments
                     values(?,?,?,?,?,?)', [comment["id"],
                                 comment["from"]['id'], comment["from"]['name'], comment["message"], postid,
                                 getTime(comment["created_time"])]
        print 'insert into the user_comments table [%s, %s, %s, %s, %s, %s]' % (comment["id"], comment["from"]['id'],
                                                                               comment["from"]['name'], comment["message"], postid, comment["created_time"])
    con.commit()

def checkIfDeleted(postid):
    global ACCESS_TOKEN
    graph = GraphAPI(ACCESS_TOKEN)
    try:
        post = graph.get(postid)
    except:
        return True
    return False

def getAccessToken():
    ACCESS_TOKEN = raw_input("Enter a valid Facebook access token: ")

def createTables(DB_NAME):
    (cur, con) = getDBConnection(DB_NAME)
    try:
        cur.execute('CREATE TABLE IF NOT EXISTS photos_info(photo_id TEXT, album_id TEXT, album_name TEXT, photo_desc TEXT, created_time timestamp, updated_time timestamp, author_name TEXT, author_id TEXT, picture_link TEXT, source_link TEXT, link TEXT, likes_count INT, shares_count INT );')
        cur.execute('CREATE TABLE IF NOT EXISTS user_comments_photos(commentid TEXT, comment TEXT, userid TEXT, userName TEXT, likes_count INT, created_time timestamp, photo_id TEXT);')
        cur.execute('CREATE TABLE IF NOT EXISTS user_likes_photos(userlikesid INT, userid TEXT, userName TEXT, photo_id TEXT);')
        cur.execute('CREATE TABLE IF NOT EXISTS user_comments (commentid TEXT, userid TEXT, userName TEXT, comment TEXT, post_id TEXT, created_time timestamp )') #primary key commentid
        cur.execute('CREATE TABLE IF NOT EXISTS user_likes(userlikesid, userid TEXT, userName TEXT, post_id TEXT)') #add ID
cur.execute("CREATE TABLE IF NOT EXISTS post_info(postid TEXT, post TEXT, author_id TEXT, author_name TEXT, share_count INT, like_count INT, post_type TEXT, link TEXT, created_time timestamp, updated_time timestamp")
cur.execute("CREATE TABLE IF NOT EXISTS albums_info(album_id TEXT, album_name TEXT, album_desc TEXT, author_id TEXT, author_name TEXT, link TEXT, cover_photo_id TEXT, photo_count INT, album_type TEXT, created_time TIMESTAMP, updated_time TIMESTAMP, like_count INT, comment_count INT;")
cur.execute("CREATE TABLE IF NOT EXISTS albums_likes(num_albumlikes INT, userid TEXT, username TEXT, album_id TEXT;")

cur.execute("CREATE TABLE IF NOT EXISTS albums_comments(commentid TEXT, comment TEXT, author_id TEXT, author_name TEXT, like_count INT, album_id TEXT, created_time TIMESTAMP;")

print 'tables created successfully!'
except sql.Error, e:
    print "Error %s:" % e.args[0]
sys.exit(1)
finally:
    if con:
        con.close()

def cleandb(DB_NAME):
    (cur,con) = getDBConnection(DB_NAME)
    try:
        cur.execute("delete from user_comments;") #primary key commentid
        cur.execute("delete from user_likes;") #add ID
        cur.execute("delete from post_info;")
        cur.execute("delete from user_comments_photos;")
        cur.execute("delete from user_likes_photos;") #add ID
        cur.execute("delete from photos_info;")
        cur.execute("delete from albums_likes") #add ID
        cur.execute("delete from albums_info;")
        cur.execute("delete from albums_comments;")
        con.commit()
        print 'tables cleaned successfully!'
    except sql.Error, e:
        print "Error %s:" % e.args[0]
sys.exit(1)
finally:
    if con:
        con.close()

def getTime(timestampstring):
    timeval = time.strptime(timestampstring[:-5], '%Y-%m-%dT%H:%M:%S')
    gmt_offset_seconds = int(timestampstring[-14:])*60*60
    return datetime.fromtimestamp(time.mktime(time.localtime(time.mktime(timeval) -
        gmt_offset_seconds)))

def getAllInstancesOf(objectid, attr, type="post"):
    global ACCESS_TOKEN
attr_list = []
graph = GraphAPI(ACCESS_TOKEN)
try:
    post = graph.get(objectid)
except:
    print 'objectid deleted: ', objectid
    return None
if type == "post":
    attr_vals = graph.get(objectid+'/'+attr+'?summary=true')
else:
    attr_vals = graph.get(objectid+'/'+attr)
attr_list= attr_vals['data'] #list of [id name pairs]
next_flag = False
if 'paging' in attr_vals.keys():
    if 'next' in attr_vals['paging'].keys():
        next_flag = True
if next_flag:
    next_page = attr_vals['paging']['next']
else:
    next_page = ""
#expected_count = post[attr]['count']

while (next_page != ""):
    #print next_page
    res = requests.get(next_page)#edit next_page
    if res.status_code == 200:
        attr_list = attr_list + res.json()['data']
        try:
            if 'next' in res.json()['paging'].keys():
                next_page = res.json()['paging']['next']
            else:
                next_page = ""
        except:
            if 'paging' not in res.json().keys():
                print res.json()
                next_page = ""
    return attr_list

def pullFacebookDataFromTextFile(DB_NAME):
    (cur,con) = getDBConnection(DB_NAME)
    try:
        while file_count <= 3:
            sandy_file = open('sandyspets6-8'+str(file_count)+'.txt','r')
            #out_file = open('tmp','w')
            # deleted_posts_file = open(deleted_info,'w')
            print 'reading from SandysPets2.txt...' + str(file_count) + '...

            file_count += file_count
            for line in sandy_file:
                json_line = simplejson.loads(line)
                #print line
                #out_file.write(line)
                deleted_flag = False
                for element in json_line["data"]: postid = element["id"]
deleted_flag = checkIfDeleted(postid)
if deleted_flag == True:
    deleted_counter += 1
    insertPostInDB(element, deleted_flag)
    cur.execute('insert into objects_list
values(?,?)', [postid, ('y' if deleted_flag else 'n'))]
    print 'inserted into deleted_objects table (%s,%s)' % (postid, ('y' if deleted_flag else 'n'))
    con.commit()
except sql.Error, e:
    print "Error %s:" % e.args[0]
    sys.exit(1)
finally:
    if con:
        con.close()
    print 'num deleted objects: ' + str(deleted_counter) + ' out of ' + str(num_posts) + ' posts'

# create all tables if tables don't exist in DB
createTables(DB_NAME)
# print "page_name: "+page_name
# insert all photos from sandyspets page to the sqlitedb
insertAlbumsAndPhotosInDB(page_name, DB_NAME)
# pull Timeline Data
getPageFeed(page_name, DB_NAME)