Destination Finder

TEAM: Noor Fazli, Isra Sunhachawi

ABSTRACT: Selecting a travel destination can be difficult given the overwhelming amount of choices, information and reviews available on the Internet. To simplify this process, we introduce an intelligent recommender system that operates on the premise that a person tends to put more weight on a friend's recommendation rather than reviews by other people. Using data made available through Facebook, the Destination Finder provides a ranked list of recommended destinations that are vetted by friends and supplements this list by mining data from review sites such as Yelp.

1. INTRODUCTION

When it comes to selecting a new place to visit, the Internet opens doors to a seemingly unlimited amount of information leaving a person inundated with choices. A user can search Yelp or Urban Spoon to find places to eat, Travelocity or Expedia for hot travel deals or comb the latest Groupon or Living Social deals to find a last-minute getaway or spa treatment.

Through our experience, we believe that it is the feedback from a person's friends that plays the heaviest influence on their decision making because a person will know if they have matching interests with a particular friend and is more likely to trust their judgment than a stranger who is potentially posting biased reviews.

Instead of creating a site that simply consolidates data from other sites or that collects its own reviews, we embarked on utilizing this observation to provide a concise, ranked list that would save the user time and simplify the decision-making progress.

As social media has become an integral part of the daily lives of Internet users, the amount of data collected on places we or our friends visit (or "check in" at) is quickly multiplying and this project seeks to drive value out of this data. With over 1 billion users, we chose Facebook as the most reliable option for gathering data on a user's friends compared to other applications such as foursquare.com which has a much smaller user base of around 45 million.

In this paper, we describe how we developed a web application to extract "check in" data from Facebook in order to provide a customized, ranked list to the user. As additional decision aids, we performed sentiment analysis on associated comments and sought to retrieve related data from other review sites using a web crawler. In order to make the Destination Finder effective globally and across multiple destination categories, we would have to create a system that interfaces with multiple services (e.g. flight reservations, hotel bookings, reviews). This would entail creating a large-scale system, which would be outside the scope of this course. Instead, we decided to create a baseline application that can be further expanded to help solve this problem.

2. RELATED WORK:

The concept of providing recommendations for a destination whether it is a vacation spot, restaurant or other venue is not new. Websites such as Yelp provide reviews for all sorts of businesses and allows you to connect with friends to see the places they go and their opinions. Another related website is FourSquare which is a social-based check-in site that encourages users to share their location by granting perks such as discounts or even the ability to become the virtual mayor of a place by collecting the most check-ins there.

What makes the Destination Finder novel is that it utilizes data posted by a user's friends from the popularly used Facebook application to provide a customized recommendation list by analyzing information such as comments and likes (i.e. number of people who liked a post). Most other sites tend to show where friends are going or provide recommendation lists based on all users but do not seek to find inferences and patterns to provide a customized list of this sort.

3. <u>PROBLEM DEFINITION</u>

The primary challenge was to extract location-related data from Facebook as well as associated information that could be used for ranking. With this information an algorithm for calculating a score for each location needed to be defined to return a customized destination ranking.

The input for the Destination Finder application comes from the Facebook Graph API which provides access to data contained within Facebook. There is a large number of data available in what Facebook refers to as "nodes" and "edges", where a "node" represents an object such as a Facebook post and an "edge" represents connections associated to that object such as its comments.

The expected intermediate output of the calculation is a listing of locations visited by all of a user's friends. For each of these locations, the following information associated to the check-in needed to be captured:

- Place Name
- Address
- Message – what the user entered as a comment when they checked in
- Likes – how many other users liked the original poster's check-in
- Comments and associated likes – comments that others made about the check-in and the number of likes for the comment
- ID of poster – User ID of the person that created the check-in post
- Creation Date – the date the check-in was recorded

The final output of the application uses the intermediate output and calculates a score for each location by utilizing these attributes which is then displayed to the user in descending score order.

Once the top-k (e.g. top 10, top 20) destinations are determined the next step is to associate related review information crawled from other sites which could then be consolidated and displayed with each destination recommendation.

This application should be accessible using any web-enabled device such as a smartphone.

4. METHODS

The high-level workflow that was used to produce destination recommendations is presented in Figure 1 and described in further detail in the following section. The application logic was implemented using PHP even though we did not have extensive experience with the language. We chose PHP over Java since we found plenty of documentation regarding interfacing with Facebook, Apache Solr and other useful libraries. It also seemed more lightweight for the purposes of quickly developing a prototype application for this course.

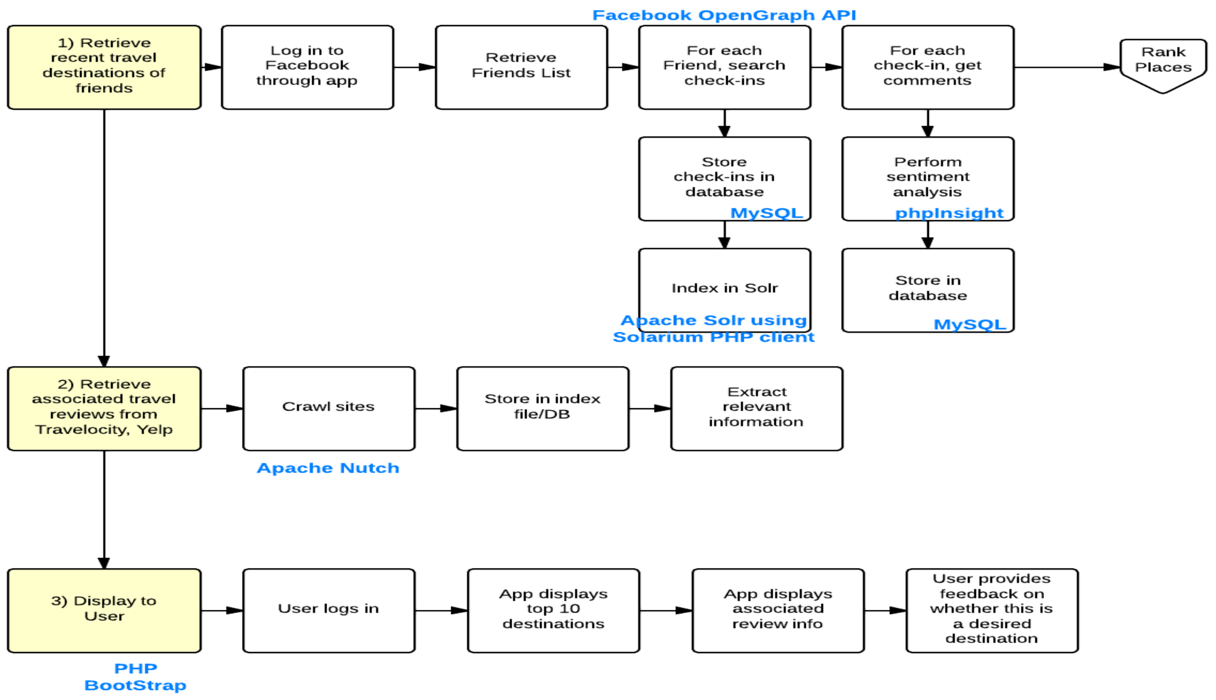

Facebook OpenGraph API
1) Retrieve recent travel destinations of friends
Log in to Facebook through app
Retrieve Friends List
For each Friend, search check-ins
For each check-in, get comments
Rank Places
Store check-ins in database
MySQL
Perform sentiment analysis
phpInsight
Index in Solr
Apache Solr using Solarium PHP client
Store in database
MySQL
2) Retrieve associated travel reviews from Travelocity, Yelp
Crawl sites
Apache Nutch
Store in index file/DB
Extract relevant information
3) Display to User
PHP BootStrap
User logs in
App displays top 10 destinations
App displays associated review info
User provides feedback on whether this is a desired destination

Figure 1: Destination Finder Workflow

4.1 GATHERING DATA

There were two sources that we needed to retrieve data from:

- Facebook
- Travel review sites

For Facebook, the first step was to gather the list of all of a user's friends in order to retrieve all of the user's friends' posts. We used this method instead of relying solely on the logged in user's newsfeed or wall data because the data available through these paths was more limited. Using the PHP SDK, we were able to perform the following actions:

1. Prompt the user to log into Facebook if they weren't already logged in so that the application had the required Facebook access tokens to retrieve desired data
2. Using the Graph API we requested all of the user's friends, parsed the JSON response and stored it in a PHP array.
3. For each friend in the array, we used another Graph API get request for "posts" submitted by that friend

4. If a post had location information available this served as an indication that this could be a potential destination to the user. This was a more direct method of identifying destination-related information than trying to mine comment text for keywords related to travel.
5. The location information and associated attributes such as the number of likes were stored in a MySQL database for subsequent retrieval and also indexed in an Apache Solr instance using a Solr library for PHP called Solarium. In retrospect, storing the data in MySQL may have been unnecessary given that the same information is in Solr but we had used that as the initial storage repository until we understood how to configure Solr.
6. For each location, we also retrieved the comments that were made on the check-in at the location. We considered using various techniques to squeeze information out of these comments such as summarization or topic analysis but due to the low density of comments we chose to provide sentiment analysis to give a general feel of whether comments were positive, negative or neutral overall in order to aid in the recommendation process. This was accomplished by passing comment text into a library called phpInsight which returns a response indicating the dominating result class of positive, negative or neutral. This information was also stored into the MySQL database.

For retrieving data from travel review sites, various methods were explored. We tried to use a developer widget provided on TripAdvisor.com to retrieve information related to a destination but it did not provide crawling functionality. We also tried using the "wget" UNIX/Linux command to crawl data from Yelp but were only able to get the index file and robot.txt file. Using this same command on other static sites provided more results. The use of Apache Nutch was also explored but we were unable to overcome the configuration issues of trying to get this running on a Windows machine. We were able to get prerequisites such as JRE, Tomcat, Solr, Cygwin and Nutch working find independently but the Cygwin interface for Apache Nutch ran into multiple configuration issues where it was not able to locate the basic Java files and was not able to run properly.

Due to these configuration issues, we were unable to incorporate review site information into the recommended list of destinations displayed on the front-end.

4.2 SEARCHING/DISPLAYING DATA

With the data stored and indexed in Solr, our next task was to determine algorithms that would be useful in ranking the destinations. The following rules/assumptions were considered:

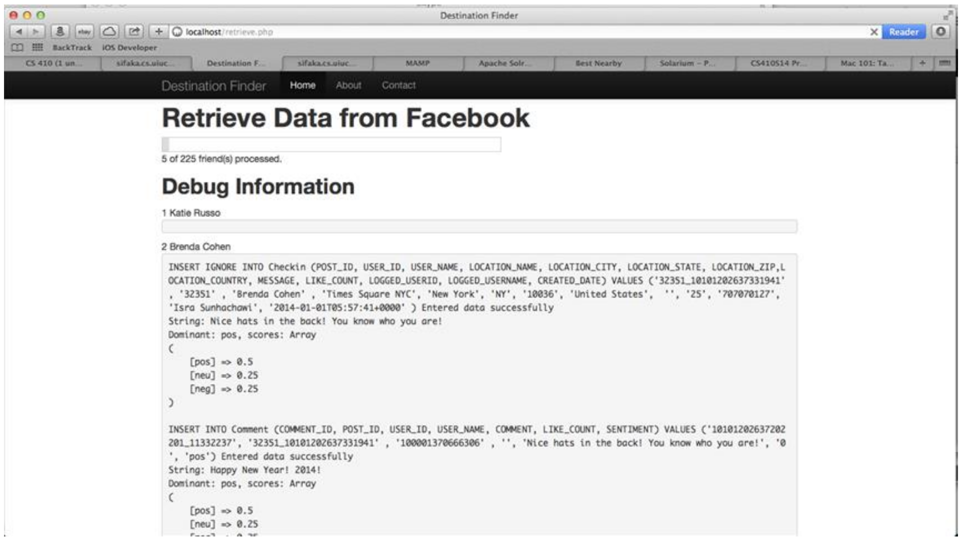
- A check-in which has many likes on the original post indicates that this is more likely to be a desired destination and should therefore increase the ranking
- If the logged in user has liked many of another friend's check-ins it is likely that they would be interested in other check-ins made by that user. Other check-ins made by that friend should be weighted higher in the ranking.
- Check-ins with a higher amount of comments can be assumed to be an indication of "buzz factor" (i.e. a lot of people are talking about it so it is possible something interesting is going on here) and should receive a slight boost in ranking
- If the proportion of negative comments exceeds a certain threshold, the ranking should be decreased

Due to time expended through troubleshooting and ramping up the learning curve in the previous steps for activities such as configuring a Facebook application, developing the user interface and learning PHP enough to communicate to a MySQL database and Apache Solr instance, we were not able to implement all the ranking rules we had defined. The recommendation list currently only utilizes the number of likes on a post to determine its relevance. The top 10 results were retrieved from Solr in a JSON format and displayed to the user.

Using built-in functionality with Solr called Solritas we also had available a simple search interface driven by Solr (which provides facilities such as detecting misspellings or removing stopwords) that could retrieve data from the indexed location information to present to the user.

5. EVALUATION/SAMPLE RESULTS

Our application performs well in retrieving check-in based information from Facebook and presenting a recommended destination list to the user however the ranking of results stills needs more work. A screenshot of debug information captured as the application processes Facebook posts and comments for a user's friends is shown in Figure 2.



Retrieve Data from Facebook

5 of 225 friend(s) processed.

Debug Information

1 Katie Russo

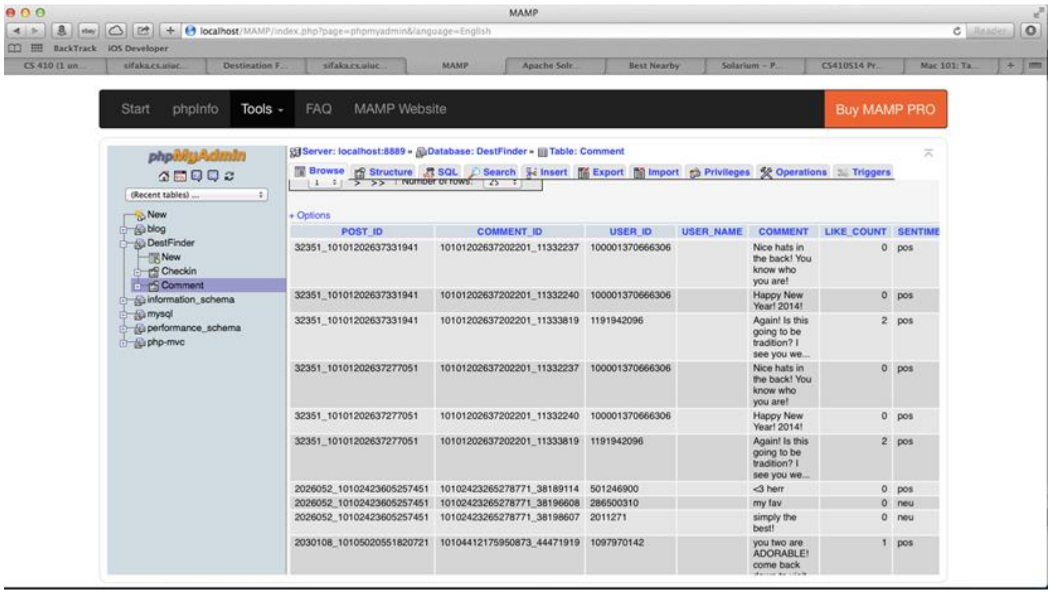
2 Brenda Cohen

```
INSERT IGNORE INTO Checkin (POST_ID, USER_ID, USER_NAME, LOCATION_NAME, LOCATION_CITY, LOCATION_STATE, LOCATION_ZIP,L
OCATION_COUNTRY, MESSAGE, LIKE_COUNT, LOGGED_USERID, LOGGED_USERNAME, CREATED_DATE) VALUES ('32351_10101202637331941'
, '32351' , 'Brenda Cohen' , 'Times Square NYC', 'New York', 'NY', '10036', 'United States',  '', '25', '707070127',
'Isra Sunhachawi', '2014-01-01T05:57:41+0000' ) Entered data successfully
String: Nice hats in the back! You know who you are!
Dominant: pos, scores: Array
(
    [pos] => 0.5
    [neu] => 0.25
    [neg] => 0.25
)

INSERT INTO Comment (COMMENT_ID, POST_ID, USER_ID, USER_NAME, COMMENT, LIKE_COUNT, SENTIMENT) VALUES ('10101202637202
201_11332237', '32351_10101202637331941' , '100001370666306' , '', 'Nice hats in the back! You know who you are!', '0
', 'pos') Entered data successfully
String: Happy New Year! 2014!
Dominant: pos, scores: Array
(
    [pos] => 0.5
    [neu] => 0.25
```

Figure 2: Check-in Information and Comments

Comments data were stored in MySQL and associated to the original post using the POST_ID as a foreign key reference. Other information such as likes for the comment and sentiment analysis is shown in Figure 3.



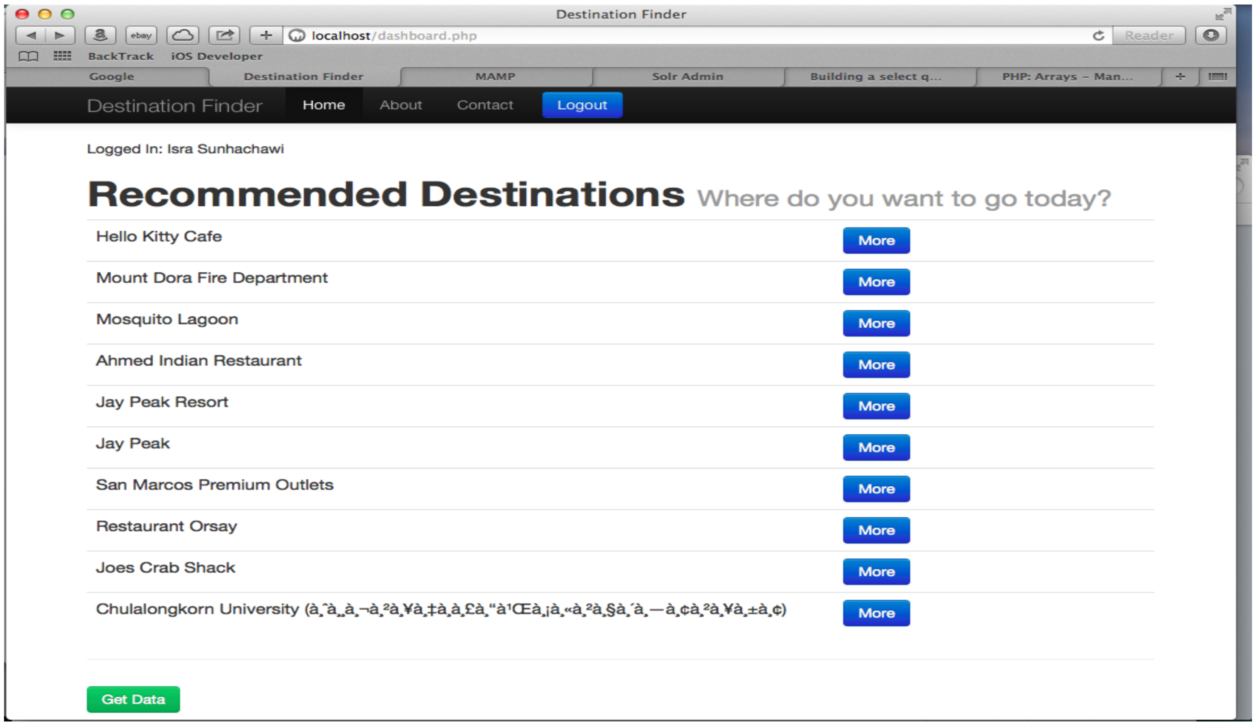
Server: localhost:8889 » Database: DestFinder » Table: Comment

POST_ID	COMMENT_ID	USER_ID	USER_NAME	COMMENT	LIKE_COUNT	SENTIME
32351_10101202637331941	10101202637202201_11332237	100001370666306		Nice hats in the back! You know who you are!	0	pos
32351_10101202637331941	10101202637202201_11332240	100001370666306		Happy New Year! 2014!	0	pos
32351_10101202637331941	10101202637202201_11333819	1191942096		Again! Is this going to be tradition? I see you we...	2	pos
32351_10101202637277051	10101202637202201_11332237	100001370666306		Nice hats in the back! You know who you are!	0	pos
32351_10101202637277051	10101202637202201_11332240	100001370666306		Happy New Year! 2014!	0	pos
32351_10101202637277051	10101202637202201_11333819	1191942096		Again! Is this going to be tradition? I see you we...	2	pos
2026052_10102423605257451	10102423265278771_38189114	501246900		<3 herr	0	pos
2026052_10102423605257451	10102423265278771_38196608	286500310		my fav	0	neu
2026052_10102423605257451	10102423265278771_38198607	2011271		simply the best!	0	neu
2030108_10105020551820721	10104412175950873_44471919	1097970142		you two are ADORABLE! come back	1	pos

Figure 3: MySQL Database

Sample recommended destinations for Isra are shown in Figure 4. His assessment is that the "Hello Kitty Café" would likely be the last place he would ever frequent but destinations such as the Mosquito Lagoon and Joe's Crab Shack could be potential destinations he would go to.

Similar feedback was provided from other friends who tested the system. Although there were some results that did seem interesting, it was only a small proportion and often not ranked as desired. To improve upon these results, some avenues to explore is to tag each location with a category type to filter out certain types of destinations that a user might not be interested in such as check-ins at a university. Also, ranking based on the number of likes alone is not sufficient to provide robust results so the ranking function would need to incorporate more of the business rules/assumptions discussed in Section 4.



Logged In: Isra Sunhachawi

Recommended Destinations Where do you want to go today?

Hello Kitty Cafe
Mount Dora Fire Department
Mosquito Lagoon
Ahmed Indian Restaurant
Jay Peak Resort
Jay Peak
San Marcos Premium Outlets
Restaurant Orsay
Joes Crab Shack
Chulalongkorn University (à¸ˆà¸„à¸¬à¸²à¸¥à¸‡à¸à¸£à¸“à¹Œà¸¡à¸«à¸²à¸§à¸´à¸—à¸¢à¸²à¸¥à¸±à¸¢)

Figure 4: Recommended Destinations

The web crawling portion of the application is not as efficient and needs more time and effort in order to make it functional. As for technical issues, some fixes in our queries would be to decode html whitespace and exclude it from our search query to better extract keywords. Our search engine does not do a great job of finding relevant data when there is not a direct match with the search query, so adding a way to match to related keywords would improve search results tremendously. We also had some issues converting some of the exported data from some websites to JSON format, so an improved way to extract that data would make our data set more comprehensive. Other sites ended up being very poorly designed for use with Nutch and as such we were unable to scrape them.

6. CONCLUSIONS AND FUTURE WORK:

Sharing information socially has experienced a large expansion in the recent years and there is a large amount of data available to perform analytics to meet a particular need. For this project, we accomplished the majority of our proposed goals in extracting data from Facebook and using it to provide a recommended destination list to the application user. We see a lot of value in this project because it gives a user a quick one-stop shop where they can get ideas on where to go in this world of numerous choices. Before it becomes truly useful, we would need to make it better by improving the ranking function, because right now destination is ranked high if it has more likes or more of the positive or neutral comments. This ranking function could be made more robust by applying further algorithms such as putting more weights on posts made by close friends. To provide more supplemental data we would also like to use data from other social media sites such as Twitter and Google+.

From this project we have learned a great deal about the Facebook API such as how to request and parse information. We have also learned enough to be comfortable with PHP and developing prototype sites rapidly in combination with front-end frameworks such as Bootstrap. We gained familiarity with various utilities for crawling websites to gather data and gained experience setting up an Apache Solr instance on a Tomcat server locally. Armed with new knowledge on these tools as well as concepts learned during the course we believe we can apply it to other interesting ideas in the future.

The PHP code for this project is stored in a UIUC SVN repository located here:
https://subversion.ews.illinois.edu/svn/sp14-cs410-destfinder/

7. ACKNOWLEDGEMENTS

This research was supported by the Computer Science Information Text Systems class at the University of Illinois at Urbana-Champaign Spring 2014.

8. Appendix:

Individual Contributions:

- Noor Fazli: web crawling, indexing returned data, presentation, report.
- Isra Sunhachawi : Facebook data retrieval, indexing, sentiments analysis, user interface, presentation review, report

9. References:

http://lucene.apache.org/solr/

https://nutch.apache.org/

https://developers.facebook.com/docs/graph-api