733 Final Project Report - Travel++

Hanhan Wu Shruthi Mohan

Motivation and Background

Everyone likes traveling.
Traditional search engine like Google and Bing cannot show you real current hot tourism spots.
When answering traveling questions, we want to make it simple and interesting.
Wanted to learn NLP, algorithms used in search engine and data visualization.

Main Features

- **Dr. Q** - Automatic Traveling Answers
- **Gossip Queen** - Real Time Traveling Trends
- **Map Attentive** - Map that allows people to see tourism spots cater for their own style

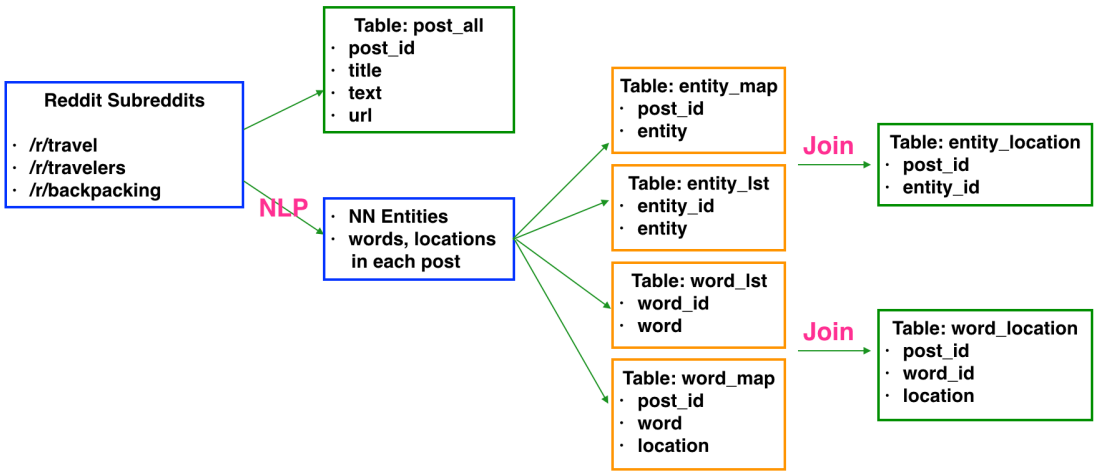
As for related work, having built light weighted search engine and did other social media mining in leisure time before, but those all have been done on Eclipse with Python, Sqlite and MongoDB. This project, we are using totally new development platform and databases, social media sources are also new.

Challenges

Questions Want to Answer	Why Challenge
How to get data from Twitter, Flickr, Instagram, Reddit and take advantage on them for all three features?	Each social media provides quite different ways to get data. It takes time to figure all them out, and whether we could get the data to help achieve the goal is a real challenge, because of privacy setting, especially geo-location data.
How to use Databricks Cloud (Spark Cluster) to do the implementation? Whether we need to use any other IDE or tools at the same time? How do we store the data?	Databricks Cloud is totally new to us, although on the user interface, we can find it has tables to store data, it has Notebook that allows you to write Python, Scala, R and Spark Sql in the same Notebook. How to connect all the development segments together was unknown.
To implement Dr.Q needs to build a small search engine, how to calculate the score to match Reddit or Wiki pages, what kind of data should be stored? When building the tables, what are the relationship between those tables?	At the very beginning, the plan to store and match Reddit and Wiki pages were the same, but during the implementation and testing, Databricks Cloud has some limitations, and the text from Reddit and Wiki need to use different score calculation methods to get more accurate results for each. It took long time to do many experiments and compare the results.
At that time, news about GraphFrame suddenly appeared at that time, and it should be a good opportunity to try it. GraphFrame helps calculate PageRank score but PageRank is just one of the methods to calculate the scores, how to combine other score calculations to get more accurate results while storing the data into GraphFrame vertices?	According to previous experiments, when building lighted weighted search engine for a specific topic like traveling, PageRank cannot play significant role although it is still useful. So, have to use other score calculation methods at the same time, most of these methods need to use words and their locations in each page, however, GraphFrame only stores urls as vertices in this case, it does not make sense to save all the words and locations with each url, meanwhile according to the experiments, the vertices forms a dataframe, it can not save so much data.

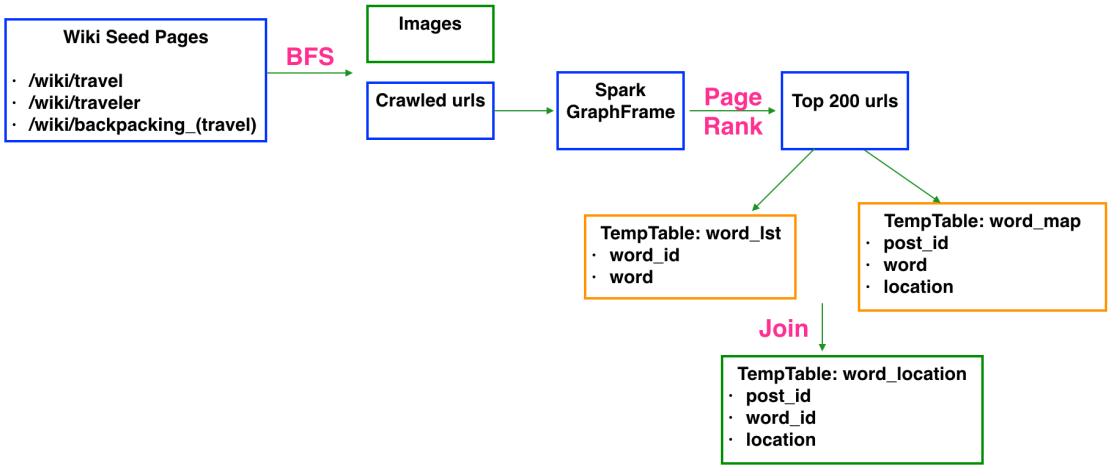
Data Processing Pipeline

Dr.Q Part 1 - Reddit Posts Match



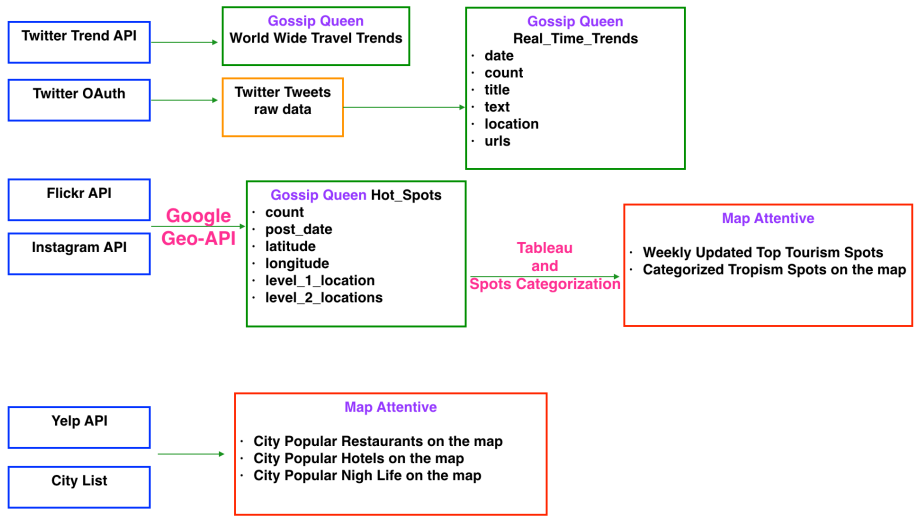
- Get raw data from Reddit
- Extract post data and store them into table **post_all**
- Using NLP extracts all the NN entities, words and their locations in each post. Store the data into tables **entity_map, entity_lst, word_lst, word_map**
- Join entity_map and entity_lst to get table **entity_location**
- Join word_map and word_lst to get table **word_location**
- Table **post_all, entity_location, word_location** will be used in post matching

Dr.Q Part 2 - Light Weighted Traveling Search Engine, Wiki References



- Get raw data from Wiki seed pages
- Crawl inbound urls from seed pages using BFS, save images at the same time
- Save all the crawled urls in Spark GraphFrame, sort them in PageRank descending order
- Because of the memory limitation of Spark dataframe, choose top 200 urls
- Do NLP on each url and generate temporary tables **word_lst, word_map**
- Join word_lst and word_map to get table word_location which is used for later calculations and queries

Gossip Queen and Map Attentive



- Green squares are Gossip Queen results, red squares are Map Attentive results
- Using Twitter Trend API get world wide trends, the trends have longer time span. Twitter OAuth allows to get tweets raw data, using keywords like “travel” to narrow down trends and finally generate real time trends by sorting the retweet count
- Using the raw data generated by Flickr API and Instagram, with Google Geo-API to get accurate locations of photo posts and track real hot tourism spots in each city
- Gossip Queen Hot_Spots provides data for Map Attentive tourism spots map visualization
- Using raw data from Yelp API to visualize popular restaurants, hotels on the map

Methodology

Tools

- Development Platform - Databricks Cloud (Community Edition)
- Databases - Tables in Databricks Cloud, or temporary tables using Spark Sql
- Language - Python, Spark Sql
- Visualization - Tableau, d3, Databricks Cloud charts

Reasons for Choosing the Tools

1. Want to learn more cloud development experiences. Databricks Cloud is a great tool for big data development and data analysis. It provides Notebook that supports Python, Scala, R, Spark Sql, you can write all these languages in just one Notebook, very convenient. The Notebooks functions quite similar to iPython Notebook add has makes development even more smooth. Since there is no need to import Spark built-in libraries, the dataframe can be generated into charts automatically, you can also do Spark Sql operation in the Notebook cell by starting with "%sql".
2. Tableau is the major visualization tool used in this project, for Map Attentive. It provides more chart choices, including maps. It is flexible that we did NOT use Tableau built-in maps, but instead used more colorful maps from other sources, but Tableau still can generates dots on our own maps. The best part is, it allows users to publish all the visualization work for free.

Major Data Methodologies

1. Dr.Q Part 1 - Reddit Posts Match

Level 1 Method - Match user query with post title, using Levenshtein Distance

Just calculate the Levenshtein Distance between posts titles and user query, choose the top 5 posts that have shortest distances. Since the code is using Spark dataframe udf, it is fast for the calculation.

Findings: Levenshtein Distance works better when 2 strings share similar length and both of them are not very long. In this case, Levenshtein Distance did not give accurate results, because the post titles are longer than user queries.

Level 2 Method - Find matched posts using NN entities

Extract NN entities from user query first. Then find posts that contain all these NN entities.

Findings: This method sets strict constraints that the Reddit posts have to contain all the NN entities appeared in a user query. This method is more accurate than level 1 method, but returns very few results and did not return the most matched results.

Level 3 Method - Find matched posts using words, locations in each post

Extract non-stopwords from user query first, then calculate scores using the following 4 approaches:

- **Score calculation approach 1** - Using words locations

Assuming words in user query appear closer to the head of a post, this post gets higher score.

- **Score calculation approach 2** - Using words distances

Assuming words in user query are closer to each other in a post, this post gets higher score.

- **Score calculation approach 3** - Using words frequency

Assuming words in user query appear more frequently in a post, this post gets higher score.

- **Score calculation approach 4** - Using all the above 3 approaches together but set different weights

After doing experiments using the above 3 approaches one by one, for Reddit posts text, words distances plays a significant role in accuracy, words locations also contributes much to the accuracy but a little less important than words distances, words frequency is useful but lead to lowest accuracy compared with the other 2 approaches. So, giving words distances highest weight, words locations gets a little lower weight and words frequency gets the lowest frequency.

Findings: Level 3 method is the most accurate method, especially approach 4, it can find the top 5 most matches results for a user query.

2. Dr.Q Part 2 - Light Weighted Traveling Search Engine, Wiki References

Some user queries cannot find satisfied Reddit posts to match, as a thoughtful search engine, we still have to give them a reasonable answer. Wiki resources are the best choices.
So, when the highest score a user query got from Reddit Post Match did NOT pass the threshold, Dr.Q will provide top 5 matched wiki urls and top matched images.

Step 1: Crawl inbound urls from a set of seed pages, using BFS. Here, just using 2 levels search has already crawled many distinct urls. Save inbound images at the same time.

Step 2: Save all the crawled urls in Spark GraphFrame, the urls are vertices and their relationships are the edges. Then use GraphFrame to calculate PageRank score for each url, sort them in PageRank score descending order.

Step 3: Choose the top 200 sorted urls, since even if using Spark dataframe udf is the fastest way, later storing each word and it location of each page occupies huge memory, Spark became terribly slow and the cloud disconnected several times. After several experiments, choose the top 200 urls for later score calculation is the most acceptable way for returning satisfying results.

Step 4: Extract words and locations from those top 200 urls and store them in Spark temporary tables. Here using temporary tables because different from fixed Reddit Posts, Wiki resources will vary when user queries are different.

Step 5: Do similar score calculations used in Reddit Posts Match Level 3 Method, to find most matched urls. Using Levenshtein Distance to find most matched images.

Findings:

- In Wiki image match case, Levenshtein Distance works very well.
- When doing score calculations for Wiki text, words frequency plays the most significant role for higher accuracy. So set words frequency the highest weight in score calculation, words locations and words distances are also important for the accuracy. This is different from Reddit Posts Match, which means, for different text sources, it is important to do experiments using the first 3 approaches one by one and observe the results first, then choose those approaches played positive role, using them together but may need to set different weights based on how important they are for the higher accuracy.

Note!

Before using score calculation approach 4, all the scores from approach 1 to 3 have been rescaled into range of [0,1]. All the scores represent the same trends, which means higher score indicates matching more, lower score indicates matching less.

Data Products

- **Dr.Q**

When a user types a query, it returns matched Reddit post urls or matched Wiki urls + images

For example:

User Query = "Could you give me some advice for post college trip? Like a trip to Europe."

Output:

```
+-------+--------------------------------------+-----------------+
|post_id|url                                   |ranking          |
+-------+--------------------------------------+-----------------+
|4b46hd |https://www.reddit.com/r/travel/4b46hd|7.633333333333334|
|4arqzu |https://www.reddit.com/r/travel/4arqzu|5.3              |
|48uu2h |https://www.reddit.com/r/travel/48uu2h|5.291666666666667|
|4atds4 |https://www.reddit.com/r/travel/4atds4|4.014285714285714|
|2ltqv3 |https://www.reddit.com/r/travel/2ltqv3|3.623076923076923|
+-------+--------------------------------------+-----------------+
```

- **Gossip Queen**

It shows world wide trends, real time trends, hot tourism spots automatically, and stores daily data into Spark tables. The data tables can be created into charts automatically.
For example:
Real Time Trends

count	title	text	location	urls
8196	ThetravelVibe	In the mood to travel	wanderlust (st. a)	https://t.co/81ALbXRE61,
4603	ThatBucketList	I want to travel. I want to feel ocean water on my skin. I want my feet to feel the dewy grass. I want to lay in flower…	Sherwood Park, AB	
4032	ericaandersenn	wish i could just drop everything & travel for the rest of my life	San Diego, CA	
3043	ThetravelVibe	I can't wait to travel the world with the one i love 😍		https://t.co/j1ZYpT9HQ9,
2250	NatGeo	The photographer of this image won the NatGeo Travel Photo Contest in 2015. Where is he now?		https://t.co/Qp7Rq3nAgU,

Hot Tourism Spots

count	post_date	latitude	longitude	level_1	level2_locations
5	2016-03-28	35.577824	-5.357379	Tétouan, Tanger-Tétouan, Morocco	باب العقلة, Café Oum Al Adouak, Agh-Tech, مجوهرات الاميرة, Maroc, Pompa Cheayri, Cafe Sara Drems, MED5, FONDO NORTE, Tetouan-Tanger, Oum El Adwak, A La Casa, HOTEL A44, La joie de Déguisement, Frescos Chicken, Bab El Okla, C.E.T Volley ball minime cadet, Bab brad, Folar zoubae, Oum Adwak Tetouan, ELI Tetouan, Académie Régionale d'Education et de Formation Tanger-Tétouan, بيع مواد البناء الطوب الإسمنتي للبناء و التسقيف البورديرات للرصيف - تجاري, MomoPhone - مومو فون, MAROC, The Moon, Marjane, Safir, Les beaux arts, Traiteur Pâtissier El Mofadal, ChickenFrescos, Omo Adwak, Saniat Rmel,
1	2016-03-28	36.732911	138.463338	Yamanouchi-machi, Shimotakai-gun, Nagano-ken, Japan	地獄谷後楽館, 後楽館, 地獄谷温泉後楽館, Jigokudani Yaen-koen, 地獄谷温泉 後楽館, Snow Monkey Park, Snow Monkeys of Nagano, 地獄谷野猿公苑 (Jigokudani Snow Monkey Park), 地獄谷温泉, 地獄谷野猿, 地獄谷野猿公苑, 渋の地獄谷噴泉, Jigokudani Monkey Park, 野猿公苑,

- **Map Attentive**

Published Visualization URL
This URL contains **3 tabs** (from left to right)
  - Yelp recommendations map for hotels, restaurants and nightlife
  - Categorized hot tourism spots on the map
  - Weekly top tourism spots and user preferences chart

Lessons Learned and Summary

- Became familiar with Databricks Cloud, its advantages out-weight AWS, as well as its limitations for larger amount of data. Got deeper insights about when to choose which platform for big data development and data analysis. Also learned how to make the Spark code runs more efficient.
- Spent longest time to do many experiments in getting satisfied results for Dr.Q 2 matching parts. Learned how to get satisfied results under the limitations of Spark. Learned deeper NLP techniques.
- Learned Spark new feature GraphFrame.
- Learned a lot to get data from varies social media and data cleaning.
- Became familiar with Tableau.

Summary: In this project, we chose the best big data development platform we have used so far which allows us to do learn its features and focus more on keeping improving the output results. It also has very high values for travelers.
Map Attentive - Very convenient for users to find weekly hottest tourism spots on the map, choosing the spots cater for their own style and find restaurants, hotels on the map
Dr.Q - Answers traveling questions in a simple but effective and interesting way
Gossip Queen - Now you will find real hottest tourism spots just like local people. And you will have much topics to share with traveling partners on the way by just checking Gossip Queen real time updates.