TREADS: A Safe Route Recommender Using Social Media Mining and Text Summarization

Kaiqun Fu, Yen-Cheng Lu, Chang-Tien Lu
Department of Computer Science, Virginia Tech, USA
{fukaiqun,kevinlu,ctlu}@vt.edu

ABSTRACT
This paper presents *TREADS*, a novel travel route recommendation system that suggests safe travel itineraries in real time by incorporating *social media data resources* and points of interest *review summarization techniques*. The system consists of an efficient route recommendation service that considers safety and user interest factors, a transportation related tweets retriever with high accuracy, and a novel text summarization module that provides summaries of location based Twitter data and Yelp reviews to enhance our route recommendation service. We demonstrate the system by utilizing crime and points of interest data in the Washington DC area. *TREADS* is targeted to provide safe, effective, and convenient travel strategies for commuters and tourists. Our proposed system, integrated with multiple social media resources, can greatly improve the travel experience for tourists in unfamiliar cities.

Categories and Subject Descriptors
H.2.8 [**Database Management**]: Database Applications—*Data Mining*

Keywords
Route recommendation, Text summarization, Social media

1. INTRODUCTION
Tourists are usually attracted by cities with strong political or economic influence, places with stunning scenery or towns with profound histories. However, tourist attractions are not always accompanied by good public security. Take Washington D.C. as an example: although it has served as the nation's capital since 1790, according to recent online statistics[1], the city suffers from unusually high crime rates with a crime rate 3 times higher than the national median, placing it in the lowest 5% overall in the United States. This presents tourists with a dilemma as they try to visit the places that they have come to see while avoiding those areas with high crime rates. With the growth of social networks, many user experiences are shared on platforms such as Twitter and Yelp. In fact, the reviews and comments on those social media platforms often contain useful information about the city, such as security instructions and recommended spots. However, due to the information deluge of comments and reviews, people are seldom able to read through all of the messages posted and identify those that are relevant to their needs.

SIGSPATIAL '14, Nov 04-07 2014, Dallas/Fort Worth, TX, USA

Inspired and motivated by these problems, we developed the *TREADS* system, a user friendly mobile application that provides an optimal routes recommendation service. It mines these pervasive social media resources to customize the service to suit an individual user's particular points of interests (POI) and leverages the historical crime record data to extract the spatial distribution pattern of all crimes reported, thus promoting the safety of both commuters and tourists. Furthermore, the text summarization services provided by *TREADS* focus on utilizing Twitter data to extract summaries of transportation related events and Yelp reviews. These functionalities will allow tourists to make the best use of their time.

The major contributions of *TREADS* can be summarized as follows:

- **Route recommendation service:** *TREADS* provides an optimal route for an individual user based on his or her stated preferences. Implementation of this feature is based on the POI distributions and crime records data.
- **Query expansion for transportation tweets:** The system applies query expansion techniques to collect transportation related tweets from Twitter. A NoSQL database is established to maintain the large amount of transportation related tweets.
- **Yelp reviews and tweets summarization:** Two different extractive text summarization techniques are utilized for both Yelp reviews and transportation related tweets. This feature reduces the time consumed by unnecessary reading.
- **Integrated mobile application:** A user friendly mobile application interface was developed to combine all the above functions. It utilizes several state-of-the-art web technologies to provide an efficient and convenient user experience.

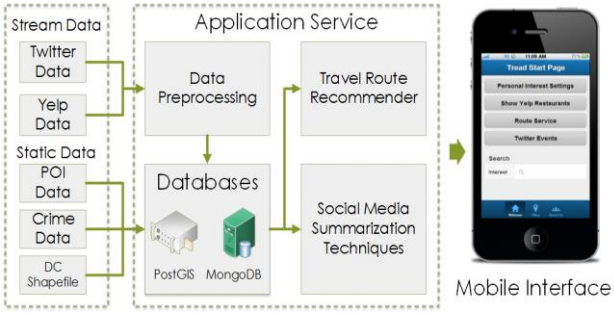


Figure 1: System Architecture

2. SYSTEM ARCHITECTURE
This section describes the system architecture for *TREADS*. At the higher level, the system consists of three main components: data pre-processing, application services, and the user interface. Figure 1 shows the high level system architecture.

2.1 Data Processing

The main function of this component is to parse, pre-process, and store the three types of data into appropriate databases, i.e., Twitter, POI, and crime records. For Twitter data, we utilized Twitter Search API to crawl real-time tweets, and maintained a MongoDB for storing tweets. The detailed process is described in Section 3.1. For POI data, we downloaded 3518 points of interest in the Washington DC area from *OpenDataDC*[2], including both popular restaurants and tourist attractions. In order to verify the original POI data in *OpenDataDC*, a Yelp POI verifier was implemented to ensure that all the POIs were accessible on Yelp. After verifying each POI by calling up the Yelp search and business API, we obtained 1443 Yelp POIs. For crime data, a .NET based data preprocessor was implemented for information extraction and database insertion for local crime record data[3].

2.2 Application Services

Two types of databases are maintained in the backend of *TREADS*: the relational database consists of a traditional relational PostGreSQL with PostGIS database for spatial datasets such as crime records data and shape files for the Washington DC area, while the NoSQL database contains Twitter data, Yelp POI data and Yelp review data. MongoDB was utilized for the NoSQL database due to its flexibility and scalability. These two databases were chosen because in the Python Flask environment they are highly collaborative with each other, which enables us to move on to further processes more smoothly.

Based on city roadway data from the PostGIS database, along with route ranking factors such as crime records and Yelp POIs from the MongoDB, we implemented the optimal route recommendation service according to the individual user's stated interests. A user interest extraction algorithm was implemented to learn and generate appropriate POIs for each user.

MongoDB was leveraged by summarizing the transportation related tweets and Yelp POI reviews. Although these are both extractive textual summarization techniques, they utilize different document structures[4]. The detailed algorithms are described in Section 3.3.

2.3 Mobile Interface

User interactions and operations such as sending requests and receiving results or feedback sent back from the backend server are performed by this module. This web-based mobile application utilizes Sencha Touch to construct the framework. All communications to the backend Python Flask web server were implemented by Ajax technology. Google Maps APIs were leveraged to enable location based services such as route recommendations, crime heat maps, and Yelp POI visualizations.

3. FEATURES

TREADS is capable of crawling and extracting transportation related tweets from the Twitter server. The dynamic crawling process developed can be applied to multiple cities. The optimal route recommendation service is the most important feature of the new system. It takes user's preferences as inputs and suggests the optimum travel route that combines the most interesting places with the least risk of dangerous encounters or traffic delays. Text summarizations for tweets and Yelp reviews are also provided by the proposed system. As they are both extractive summarization techniques, adopting this approach enables us to eliminate redundant readings before integrating the traffic status from Twitter data with the tourist attraction characteristics from hundreds of Yelp user reviews.

3.1 Tweets Query Expansion

A data collector utilizes the Twitter timeline API to retrieve the latest tweets by a set of predefined influential users under the transportation topic. Four influential users were initially selected by the system, namely, *WTOPtraffic*, *VaDOT*, *drgridlock* and *DCPoliceDep*. These users are active under the topic of transportation in the Washington DC area. Figure 2 illustrates the process of data collection and tweet query expansion.

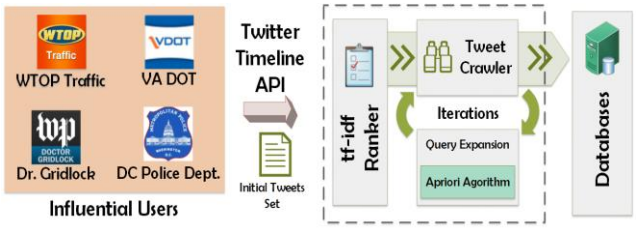
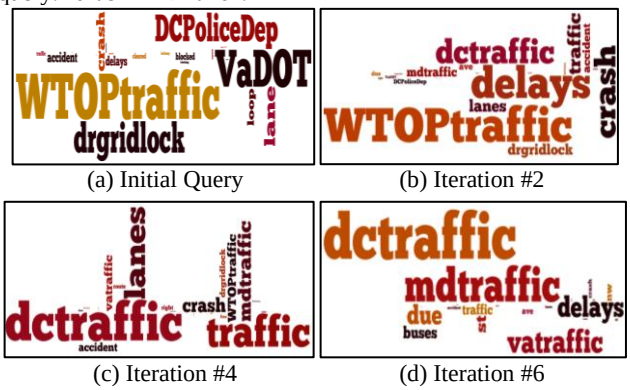


Figure 2 : Data Collection and Tweets Query Expansion

A collection of initial transportation related tweets set T_I was extracted from the influential users, after which a *tf-idf* ranker was applied to the set of extracted tweets T_I, defining words with high *tf-idf* scores as keywords in list Q because we are confident that the words that frequently occur in their tweet content are more likely to be transportation related. Then, this new keywords list Q became the expanded query to be inputted into the Twitter Search API to crawl all the tweets that match the queries.

The Apriori algorithm was leveraged to determine the association rules between the words in T_I. The reason we chose to utilize association rules algorithms was to avoid generating a long query with single words. If not applied, our data crawler would only send one word at a time as the query for the Twitter Search API, but by applying this algorithm any one of *"crash", "right", "lane"* entered separately as queries will retrieve all tweets containing the word *"crash"*. There are several obvious problems with this method, however: not only would the noise in these results be enormous, the high number of web requests sent means that the data crawler will rapidly reach its rate limitation.

In order to solve these shortcomings, the Apriori algorithm was applied to identify the minimum support *wordsets* in the vocabulary lattice. The concept of "*wordsets*" is equivalent to "*itemsets*" from items transaction mining. One of the minimal support *wordsets* serves as the new query, e.g., *["crash", "lane"]*, and logic connections provided by Twitter search APIs join these two or more words together as a more complicated query: "*crash AND lane*".



(a) Initial Query (b) Iteration #2

(c) Iteration #4 (d) Iteration #6

Figure 3 : Word Cloud for Query Expansion.

Figure 3 shows the word clouds for the initial query and resulting queries after iterations of expansions. Figure 3(a) shows the word cloud for the original query generated by the influential users' tweets sets T_I. Figures 3(b), (c), and (d) show the queries generated by iterations two, four and six, respectively. These results indicate that the query is changing gradually from a specific topic (set of keywords) that focuses solely on the influential users to a more general transportation related topic. Based on our observations, the query will eventually converge after an average of 6 query expansion iterations has been performed. The expanded query broadens the searching space while maintaining appropriate filtering keywords. It thus helps to retrieve more traffic related data while preserving the data quality.

3.2 Routing Schemes

Users provide the locations of their origin and destination as input parameters for the route recommendation service. The system then suggests an optimal route between the two locations that considers three factors: length, safety, and user intention. Users are allowed to customize the weights (factor distribution) of these factors to adjust the relative influence of each. Inspired by the existing shortest or fastest routing algorithms, we developed our routing algorithms based on the Dijkstra and A* shortest path algorithms. The Dijkstra algorithm is a special case of the A* algorithm with the heuristics set to zero. As the Dijkstra algorithm outperforms the A* algorithm when the target graph is relatively small, we selected the Dijkstra algorithm as the default routing algorithm since the roadway network for Washington DC area is relatively small.

The problem of finding the optimal route was projected onto an optimization problem that minimizes the cost of the route [5]. The edge cost weighting was determined from three perspectives: *distance factor*, *crime factor* and *POI factor*. By adjusting the weights of these three factors apprpriately, we propose four routing schemes: shortest path, safest path, POI path and optimal path.

A. Shortest Path Scheme

Distance is the basic factor for most routing systems. By considering this factor, the edge is weighted only by the length of the trajectory. The objective function is:

$$\underset{S\to D}{\overset{E}{MIN}}\{\sum_{e\in E} length(e)\} \quad (1)$$

where *S* and *D* are the source and destination predefined by the user. The function identifies a set of trajectories with the shortest total lengths.

B. Safest Path Scheme

The ability to incorporate crime rate data is an important aspect of *TREADS*. Since an assumption has been made that users of our application will treat safety as a relatively important factor while they are traveling, the system takes this into consideration and recommends the safest route, thus improving the user experience. This factor weights the trajectories by the number of crime reports within its distance-d buffer area. The objective function is:

$$\underset{S\to D}{\overset{E}{MIN}}\{\sum_{e\in E} crime(e)\} \quad (2)$$

which identifies a set of trajectories with the fewest accumulated crime incidents.

C. POI Path Scheme

This factor takes the numbers of points of interest or attractive nodes into account. The objective function here is:

$$\underset{S\to D}{\overset{E}{MAX}}\{\sum_{e\in E} POI(e)\} \quad (3)$$

This identifies a set of trajectories that maximizes the number of points of interest for that specific user.

D. Optimal Path Scheme

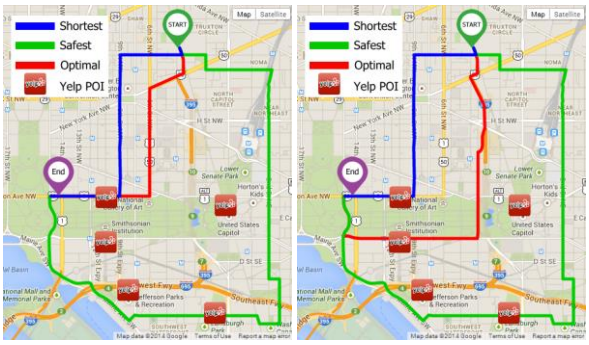
This scheme combines the above three edge weighting schemes by specifying a *safety level* β. The objective function for this scheme is:

$$\underset{S\to D}{\overset{E}{MIN}}\{\sum_{e\in E}\{\alpha\cdot nlenght(e)+\beta\cdot ncrime(e)+\gamma\cdot(1-nPOI(e))\}\} \quad (4)$$

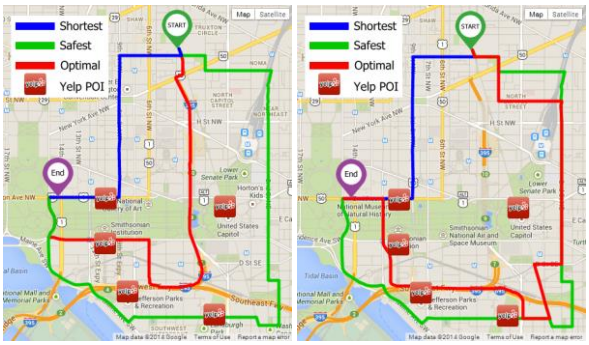
where *nlength(e)* is the normalized edge length, *ncrime(e)* is the normalized crime number, and *nPOI(e)* is the normalized POI number. That is to say:

$$nlength(e) = length(e)/max_{e'\epsilon E}\{length(e')\} \text{ and}$$
$$ncrime(e) = crime(e)/max_{e'\epsilon E}\{crime(e')\} \text{ and}$$
$$nPOI(e) = POI(e)/max_{e'\epsilon E}\{POI(e')\}$$

The *factor distribution* $[\alpha, \beta, \gamma]$ is a three dimensional vector which is located on the plane $\alpha+\beta+\gamma=1$, where $\alpha,\beta,\gamma \in [0,1]$, and the previous three schemes are special cases when $\alpha=1$, $\beta=1$, and $\gamma=1$ respectively.



(a) $[\alpha, \beta, \gamma] = [0.8,0.1,0.1]$ (b) $[\alpha, \beta, \gamma] = [0.3,0.3,0.4]$



(c) $[\alpha, \beta, \gamma] = [0.1,0.1,0.8]$ (d) $[\alpha, \beta, \gamma] = [0.1,0.8,0.1]$

Figure 4. Routing Results for Different Factor Distributions

In Figure 4, the green and purple pins represent the start and end locations respectively. The blue, green, and red lines are shortest path, safest path, and the optimal path, respectively. The red markers are locations for the top ranked Yelp POI. The vector $[\alpha, \beta, \gamma]$ represents the factor distribution and the value for this vector can be predefined by the users. By default, the value of the vector is set to *[0.3,0.3,0.4]* and the corresponding result is shown in Figure 4 (b).

Although the blue path is the shortest, it does not necessarily take the user through safe or interesting areas. The green path goes a roundabout way to reaching the end destination, but

although it has the smallest number of crime incidents along the path, its total length is long and it may not necessarily pass by the POIs that the user is interested in. The red path represents the optimal path that is the result of the tradeoff between length, safety, and POIs. By changing the factor distribution vector of the optimal path, it approaches the shortest path when length factor α is large (0.8 in Figure 4(a)), the safest path when β is large (0.8 in Figure 4(d)), and as many POI locations as possible when γ is large (0.8 in Figure 4(c)).

3.3 Text Summarizations

In order to provide a better and more convenient user experience, *TREADS* offers two types of text summarizations. The motivation for implementing these text summarization techniques is to reduce and eliminate the need to read multiple similar texts by replacing the individual objects with a summary of the Yelp reviews and transportation topic related tweets. The two techniques involved both fall within the scope of the extractive summarization algorithms[6]. Based on these algorithms, sentences in each document are represented by vertices in a graph, and the cosine similarities are represented as the edges between the vertices. Those edges with a cosine similarity less than a predefined threshold will be removed. Then, a ranking algorithm such as *PageRank*[7] or *LexRank*[6] is applied to this graph. The proposed system applies *LexRank*, since the *PageRank* is more generally used for directed graphs. The topic ranked sentences are regarded as the summary for the documents.

Figure 5(a) shows the summarized information for one user selected Yelp POI. Inside the balloon-sized information window, the POI profile image and name are displayed at the top, while the summary and Yelp link are placed at the bottom. In the case shown in Figure 5(a), the description for this POI summarized 1411 reviews. Figure 5(b) shows the result for a real time transportation related tweets summarization. The result was generated on *May 24th 2014* at *10:05 am*, from which a user could learn about the latest and most important traffic events.

A. Tweets Summarization

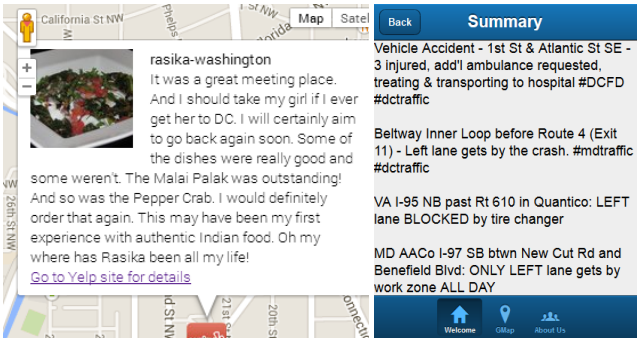
Since tweets have very different characteristics from regular articles, traditional text summarization methods are no longer adequate. There are two main challenges: 1. Twitter content is highly random and casual, with many abbreviations and informal expressions; and 2. It is hard to determine the Twitter data document size.

Twitter context trees can be constructed to bind several tweets together as documents with appropriate sizes. A corresponding context tree will be formed for each topic initially proposed by one of the influential users. This Twitter context tree is constructed based on the reply or retweet relationship, with the root of the context tree being the original tweet posted by the influential user. The structure of the context tree is then used to present the document in the traditional text summarization.

B. Yelp Review Summarization

Yelp reviews are not limited by character constraints. Most have an appropriate length so that we do not need to construct a special structure for them and can directly apply centroid-based text summarization techniques.

The main challenge in developing this function is that Yelp APIs do not provide any functions for user review retrieval. Thus, a web crawler for the Yelp reviews must be implemented. *Urllib* and the *BeautifulSoup* Python package are generally utilized for basic html page parsing. In each Yelp page, the total number of review pages can be found under the CSS class labelled as "*pagination-links*". Each review page contains a maximum of 40 user reviews and the *href* attribute for each list item element indicates the other review pages. The div tags for user reviews are labeled by the "*review_comment ieSucks*" CSS class. By seeking every review div tag under every user review page, all the user reviews for the POIs can be retrieved.



(a) Yelp Review Summarization (b) Twitter Summarization

Figure 5. Summarization Techniques Results

4. CONCLUSION

TREADS is designed to meet the needs of an end user who wishes to find the most convenient travel itinerary between desired locations that avoids high crime areas. The optimal router feature helps the user schedule a safe, convenient and enjoyable travel plan. *TREADS* implements text summarization techniques to generalize and refine redundant online textual information. This feature minimizes time wasted on trivialities. *TREADS* is envisioned to eventually go beyond city level coverage and evolve into a collaboration platform that functions nationwide to improve the overall quality of life for travellers.

5. REFERENCES

[1] *Washington DC crime rates and statistics - NeighborhoodScout*. Available from: http://www.neighborhoodscout.com/dc/washington/crime/.

[2] *Welcome - OpenData DC*. Available from: http://www.opendatadc.org/.

[3] *Data Catalog*. Available from: http://data.dc.gov/.

[4] Chang, Y., Wang, X., Mei, Q. and Liu, Y., *Towards Twitter context summarization with user influence models*. In Proceedings of the 6th ACM International Conference on Web Search and Data Mining, ACM, 2013, pp. 527-536.

[5] Shah, S., Bao, F., Lu, C.-T. and Chen, I.-R., *Crowdsafe: crowd sourcing of crime incidents and safe routing on mobile devices.* In Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM, 2011, pp. 521-524

[6] Erkan, G. and Radev, D. R., *LexRank: graph-based lexical centrality as salience in text summarization*. Journal of Artificial Intelligence Research, 22 (2004), pp. 457-479.

[7] Page, L., Brin, S., Motwani, R. and Winograd, T., *The PageRank Citation Ranking: Bringing Order to the Web*. In Proceedings of the 7th International World Wide Web Conference, 1998, pp. 161-172.

[8] Wang, B., Dong, H., Boedihardjo, A. P., Lu, C.-T., Yu, H., Chen, I.-R., Dai, J."*An Integrated Framework for Spatio-Temporal-Textual Search and Mining*," 20th International Conference on Advances in Geographic Information Systems, 2012, pp. 570-573.