OSNI: Searching for Needles in a Haystack of Social Network Data

Shiwen Cheng, James Fang, Vagelis Hristidis, Harsha V. Madhyastha†, Niluthpol Chowdhury Mithun, Dorian Perkins, Amit K. Roy-Chowdhury, Moloud Shahbazi, and Vassilis J. Tsotras

University of California, Riverside, Riverside, CA, USA
†University of Michigan, Ann Arbor, MI, USA

schen064@cs.ucr.edu, amitrc@ece.ucr.edu, jfang003@cs.ucr.edu, vagelis@cs.ucr.edu, harshavm@umich.edu, nmithun@ece.ucr.edu, dperkins@cs.ucr.edu, mshah008@cs.ucr.edu, tsotras@cs.ucr.edu

ABSTRACT

This paper presents the Online Social Network Investigator (OSNI), a scalable distributed system to search social network data, based on a spatiotemporal window and a list of keywords. Given that only 2% of tweets are geolocated, we have implemented and compared various state-of-art location estimation techniques. Further, to enrich the context of posts, associations of images to terms are estimated through various classification techniques. The accuracies of these estimations are evaluated on large real datasets. OSNI's query interface is available on the Web.

1. INTRODUCTION

The amount of user generated data increases every year, as more social interaction tools like Twitter, Instagram and Facebook are being created and more users use them to share their everyday experiences. Most research on analyzing social data has focused on detecting trends and patterns such as bursty topics [11] and popular spatiotemporal paths [10], event extraction [7], studying how information is spread [12], or analyzing properties of the social graph.

In contrast, in this paper, we study how to search social network data items, posts and images, based on spatiotemporal keyword queries. That is, we created methods to find the right needles (social data items) in the haystack (social networks), which we refer to as *investigative search*, to contrast it to the trending queries studied by previous work. Investigative search can also be viewed as exploring the currently untapped long tail of the distribution of topics in social networks. We use law enforcement as our focus application. The system capabilities and user interface were created in consultation with the University of California Police Department.

Figure 1 shows the user interface of the developed OSNI, available at `http://dblab-rack30.cs.ucr.edu/IARPA/`, where a user may select a spatial area on the map, a time range, and specify keywords. OSNI returns a list of posts (tweets) ranked by their relevance to the query. The relevance is computed using a combination of the relevance of the text (based on an Information Retrieval function) and of the image (based on the confidence of the relevance of the query to an image) to the query. Only posts that belong to the specified spatiotemporal window are returned.

A key challenge is that only about 2% of the tweets are geolocated (have GPS location). Another challenge is how to associate images with terms. For example, if a tweet's image shows a "bike" we would like to return this tweet for the query "bike" even if it does not contain this word in its text. And a third key challenge is how to scale OSNI to a throughput of millions of posts per day, and how to faciliate interactive query response times.

This demo paper has the following contributions:

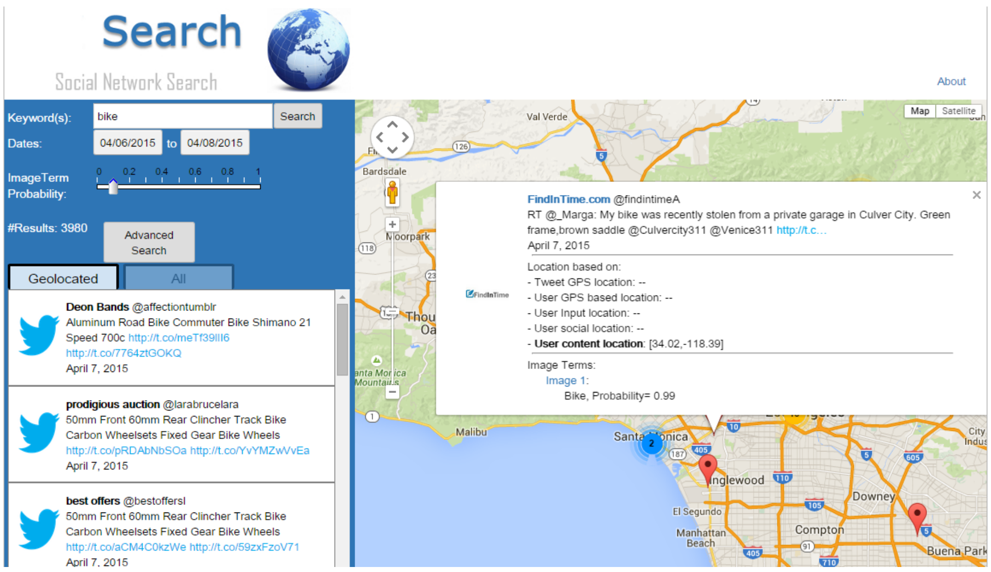
- We have adapted and implemented several social posts location estimation methods, and we have evaluated them on a dataset of millions of posts.
- We implemented a method to classify images based on the terms they are relevant to. We evaluated this classifier for various datasets.
- We built a scalable overall architecture, by combining several leading big data technologies. We experimentally evaluate the throughput and distributed performance of the system. We make the system publicly available on the Web.

2. ARCHITECTURE

The architecture of OSNI is shown in Figure 2. OSNI was written in approximately 8K lines of Java code. The whole OSNI, including all modules in the figure, in deployed on a cluster of two machines, which host instances of several systems (Cassandra, ElasticSearch, Spark). The Web server runs on one of the two servers.

The OSNI uses the Twitter Steaming API to collect tweets from Twitter. We specify a keyword-based filter on the Streaming API to only retrieve tweets that contain at least one of a collection of 64 keywords (provided by the law enforcement agency). This makes our data more focused to our domain, given that a single machine can only receive up to about 1% of Twitter's traffic. Matching records are stored in a Cassandra [5] database cluster of two nodes with replication factor 2 (each tweet is copied in both machines).

The preprocessing module continuously queries Cassandra for unprocessed records and runs a distributed job on a


Search

Social Network Search

About

Keyword(s): bike

Dates: 04/06/2015 to 04/08/2015

ImageTerm Probability:

#Results: 3980

Deon Bands @affectiontumblr
Aluminum Road Bike Commuter Bike Shimano 21 Speed 700c http://t.co/meTf39lIl6 http://t.co/7764ztGOKQ
April 7, 2015

prodigious auction @larabrucelara
50mm Front 60mm Rear Clincher Track Bike Carbon Wheelsets Fixed Gear Bike Wheels http://t.co/pRDAbNbSOa http://t.co/YvYMZwVvEa
April 7, 2015

best offers @bestoffersl
50mm Front 60mm Rear Clincher Track Bike Carbon Wheelsets Fixed Gear Bike Wheels http://t.co/aCM4C0kzWe http://t.co/59zxFzoV71
April 7, 2015

FindInTime.com @findintimeA
RT @_Marga: My bike was recently stolen from a private garage in Culver City. Green frame,brown saddle @Culvercity311 @Venice311 http://t.c…
April 7, 2015

Location based on:
- Tweet GPS location: --
- User GPS based location: --
- User Input location: --
- User social location: --
- **User content location**: [34.02,-118.39]

Image Terms:
Image 1:
Bike, Probability= 0.99

Figure 1: User Interface

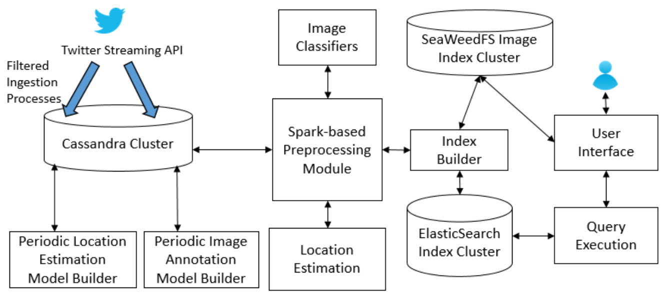

Twitter Streaming API
Filtered Ingestion Processes
Cassandra Cluster
Periodic Location Estimation Model Builder
Periodic Image Annotation Model Builder
Image Classifiers
Spark-based Preprocessing Module
Location Estimation
SeaWeedFS Image Index Cluster
Index Builder
ElasticSearch Index Cluster
User Interface
Query Execution

Figure 2: Architecture

Spark [1] cluster to process these records. For each record, we perform location estimation to determine the approximate location of the user at the time the tweet was posted (Location Estimation module in Figure 2, details in Section 3). Further, if the tweet contains an image, we also apply an image classifier, which downloads the image(s) from the Internet (only a URL is stored in the tweet record), and then runs the classification algorithm (Image Classifiers module in Figure 2, details in Section 4). If desired, images can be stored locally in SeaweedFS [4], a distributed storage system tuned towards storing images.

The enriched posts (enriched by location and image terms) are inserted into Elasticsearch [3], a distributed document search system. The Web-based query user interface, shown in Figure 1, is built using Google's Web Toolkit (GWT), which is a Java-based Web toolkit. It is hosted on Apache Tomcat.

3. LOCATION ESTIMATION

We found that only about 2% of tweets are geolocated (e.g., using the GPS of a mobile phone). Hence, we need effective ways to estimate the location of the majority of tweets. We consider several methods to assign location:

User Location String. Here we use the user's location as an estimate of the tweet's location. Twitter allows users to enter a location on a text field. Many users specify their city, e.g., "Riverside, CA", but others specify imaginary addresses, e.g., "On the moon." We use Google's Map API to map a location string to latitude and longitude.

Places. Users may check-in or otherwise specify a place in the tweets (there is a *place* field in the JSON of a tweet), which can be something like "UCR Campus, Riverside, CA." Again we use Google's Map API to get coordinates.

User GPS. As in *User Location String*, we use the user's location as an estimate of the tweet's location, but here we compute the user's location as the median of that user's GPS locations in the last 30 days. Only users with at least 3 geotagged tweets are assigned a location using this method.

Social Network Approach. This approach is based on Compton et al. [9], where the assumption is that users are often located close to their friends. Users A and B are considered "friends" if User A has tweeted at least 3 times to User B and User B has tweeted at least 3 times to User A. We first calculate GPS medians for the users that have at least three geotagged tweets in the past and store the information in the user table. Afterwards, we build a social graph and estimate the locations that are still unknown using an error minimization technique. Our approach differs from the original approach [9] in a few ways. First, the number of tweets needed to be a "friend' is decreased from 3 to 2 to increase the size of the graph because our graph was too sparse. The graph size increased by 21% and the accuracy went down only by 2%. Second, we are making use of the places field used in the user's tweets to assign more locations in the graph.

Content-based. This method looks for "local" words – e.g., "howdy" in Texas or "White House" around the White House in Washington, DC – in the body of the tweets to assign location to them. We build upon the method proposed in Cheng et al. [8], where instead of using cities, we use zipcodes (over 31,000 zipcodes in the US), that is, we assign terms to zipcodes. Further, we use a much larger number of tweets to increase the accuracy. Specifically, we use over

100 million geotagged tweets to calculate the focus and dispersion for each word to decide if it is a local word. Then, we build an index that maps terms to zipcodes with a probability, and assign to a tweet the zipcode with the highest probability among its words.

Implementation considerations. The *social network approach* has two parts. The preprocessing and the graph processing. The first step of the preprocessing is to create a unidirectional graph based on who tweets to whom so we can later build the friendship graph. The second is to collect tweets that have GPS locations and places location. In the graph processing, we first calculate the GPS medians for users with more than 3 geotagged tweets. Next, we build the graph and estimate the locations of their friends. The graph processing takes more than one day to run on a single machine, and hence we execute it once a week. The preprocessing is performed once per day. Similarly, for the *content-based approach* the local words are calculated once per day using gathered data from the last week.

Dataset and Evaluation. To measure the *accuracy* of our approaches, we used as ground truth 100 million geotagged tweets and compared the estimations of the various methods to these GPS locations. The *coverage* is the percentage of tweets that are assigned a location using a method; note that a tweet may be assigned a location based on multiple methods.

Type	**10 miles**	**30 mile**	**50miles**	**coverage**
Tweet GPS	100%	100%	100%	1.97%
User String	71%	73%	74%	27.6%
Places	81%	82%	83%	2.3%
User GPS	92%	95%	96%	1%
Social	82%	83%	85%	3.31%*
Content-based	24.7%	25.9%	27.3%	76%

Table 1: Tweet Location Estimation Accuracy and Coverage Evaluation. In Social Network approach, we only consider posts that have not already been assigned a location through Tweet GPS, User String, Places or User GPS.

4. IMAGE ANNOTATION

In this section we study how to extract keywords from the images of tweets to enhance the accuracy and effectiveness of our query interface. That is, a tweet that shows a picture of a bike, should be returned for query 'bike' even if it does not contain the word 'bike' in its body. We found that about 25% of Tweets have images.

Specifically, the image annotation module inputs an image and outputs a set of (term, confidence) pairs, e.g., (bike, 0.72), where the confidence denotes how probable it is that the image is related to the term. We use a vocabulary of terms that we want to detect in images and for each term we build a classifier.

After experimenting with various classifiers and image feature extraction methods, we chose Support Vector Machine (SVM) as classifier and SURF Detector [6] based Bag-of Words Model [13]. We found that single-class classifiers are more accurate than multi-class ones.

The method for training the classifier works in two major steps. At the first step, SURF features are extracted from the training set (described below) that contains all images of all classes. A visual vocabulary of features (Bag of Words) is created by reducing the number of features through quantization of feature space using K-means clustering. The resulting space has 5000 features. Then, in second step, occurrences of all visual features in the vocabulary are calculated from each of the images. A histogram of features is created per image, which is a 5000-entries long feature vector, which is a reduced representation of the image. Using these feature vectors and corresponding known labels for images in the training set, the SVM classifier is trained (offline, once per week). In the online system, when a new image comes, SURF features for the image are calculated and the image is represented as a feature vector. The label of the image and the confidences are estimated by feeding the feature vector into the SVM models created during the training phase.

Search Engine	Sample Images for query 'bike'
Google	
Twitter	

Table 2: Examples of a few top ranked and relevant images collected from Google and Twitter based on textual query 'bike'.

Data Collection and Evaluation. We consider 12 terms, and we build a training set as follows. We query Google Images and Twitter Images interfaces for each of the terms. Then, the retrieved images are checked manually to remove false matches. In total, we keep 200 images from Google and 200 from Twitter per term, that is, we have a total of 4800 images. The query words are: bike, gun, robbery, crowd, car, concert, murder, drunk, fire, helmet, family and friends. These words were chosen based on the law enforcement focus of the project.

The reason behind collecting training images from Google and Twitter is twofold. First, there is no existing image dataset, that contains images corresponding to all possible query words. Second, carefully chosen images from web search and social media search will make the training set both diverse and relevant. Table 2 shows examples of diverse images collected from Google and Twitter for 'bike.' We see that the Google images are generally more "clean", whereas the Twitter images offer more diversity. As we will see later the combination of the two leads to better accuracy.

K-fold ($k = 10$) cross-validation is used to test the classification performance. Table 3 shows the performance of the image classifier significantly varies, when trained with different training sets. The results demonstrate that incorporating images from both Google and Twitter for training improves classifier accuracy, as more diverse and informative set of images are included.

Our classifier has high accuracy (more than 80%) on categories like bike, gun, crowd etc. On the other hand, the accuracy on complex categories like drunk and murder was lower (around 50%). Some interesting examples of correctly identified and incorrectly identified images for category 'bike', 'friends' and 'gun' are shown in Table 4.

Testing / Training	Google	Twitter	Google & Twitter
Google	63%	36%	49.5%
Twitter	43%	31%	37%
Google & Twitter	79%	55%	67%

Table 3: Accuracy of Image Classifier with different sets of training and test images

JavaCV (Java interface to open computer vision library) is used to extract the image features. When running as a stand-alone application, the image classifier takes 5.98 millisecond on average to classify an image for a term on a system of quad-core Intel Core i5-4200M 2.50GHz CPU, 8 GB DRAM, 500 GB 7.2K RPM HDD.

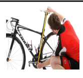
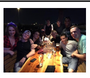
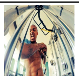
True Positive	Bike(0.652)	 Bike(0.879)	 Friends(0.437)	Friends(0.673)	Gun(0.828)	Gun(0.99)
False Positive	Gun(0.805)	Gun (0.89)	Friends (0.374)	Friends (0.332)	Bike (0.722)	 Bike (0.247)

Table 4: A few True-Positive and False-Positive examples from Image Classifier. The detected class is given below the images; confidence is in parenthesis.

5. SCALABILITY

Our OSNI is currently deployed on a cluster with two servers, each with two 6-core Intel Xeon E5-2630 2.30GHz CPUs, 96 GB DRAM, 4x 4 TB 7.2K RPM SATA HDDs, and connected via gigabit Ethernet.

Total records	3.46M	100%
Records w/ image URL	863.62K	24.96%
Records w/ non-image URL	1.32M	38.15%
Records w/ no URL	1.59M	45.95%

Table 5: Tweet record statistics for July 2015.

Latency. Based on our current set of 65 keywords used by the Twitter Streaming API to scrape Twitter's data, we store approximately 3.5 million tweet records per day. On average, it takes 18 hours to process 1-day worth of tweet records, or 18.5 ms per record. Table 5 shows the average daily composition of these records over a one-month period (July 1, 2015 to July 31, 2015). We see that about half of the posts have no URL, which support our intuition that there are many non-news related posts, which can be uses for investigative exploration (viewing a user as a witness).

P	Location	Image	Index	Total
1	17.42	10.32	534.45	6.68
8	168.0	67.79	1622.67	38.44

Table 6: Throughput, in records per second, of the OSNI indexer for different levels of parallelism (P).

Throughput. We evaluate the throughput of our OSNI indexer on a sample set of 50K records to understand the performance of the location and image classifiers, and inserting into the Elastic Search index. We compare performance using parallelism level 1 and 8. In Spark, parallelism determines the number of shards the data is split into before being distributed for processing. We show the throughput of each component and the full indexer in Table 6. We show clear speedup improvement when increasing the parallelism, and expect further improvements on a larger-scale cluster.

6. CONCLUSIONS AND FUTURE WORK

We presented OSNI, an interdisciplinary system to facilitate searching in social networks. The key contributions are the location estimation, the terms extraction from images, and the scalable architecture. To scale to more terms in the image annotation phase, we will study how multi-class classifiers can be effectively applied – single-class classifiers performed better in our experiments. Further, we are working on building a system that can collect informative training example images without human effort in filtering out irrelevant images. Finally, instead of utilizing different modules (eg. Cassandra, Spark), we are examining how to implement OSNI over a unified framework like AsterixDB[2].

7. ACKNOWLEDGMENTS

This research was supported by the Intelligence Advanced Research Projects Activity contract number 2014-14071000011. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA or the U.S. Government.

8. REFERENCES

[1] Apache Spark. `http://spark.apache.org/`.
[2] AsterixDB. `https://asterixdb.incubator.apache.org/`.
[3] Elasticsearch. `https://www.elastic.co/products/elasticsearch`.
[4] SeaweedFS. `https://github.com/chrislusf/seaweedfs`.
[5] The Apache Cassandra Project. `http://cassandra.apache.org/`.
[6] H. Bay, T. Tuytelaars, and L. V. Gool. Surf: Speeded up robust features. In *ECCV*, pages 404–417. Springer Berlin Heidelberg, 2006.
[7] H. Becker, D. Iter, M. Naaman, and L. Gravano. Identifying content for planned events across social media sites. In *ACM WSDM*, pages 533–542, 2012.
[8] Z. Cheng, J. Caverlee, and K. Lee. You are where you tweet: a content-based approach to geo-locating twitter users. In *ACM CIKM*, pages 759–768, 2010.
[9] R. Compton, D. Jurgens, and D. Allen. Geotagging one hundred million twitter accounts with total variation minimization. In *Big Data*, pages 393–401. IEEE, 2014.
[10] M. De Choudhury, M. Feldman, S. Amer-Yahia, N. Golbandi, R. Lempel, and C. Yu. Automatic construction of travel itineraries using social breadcrumbs. In *ACM Conference on Hypertext and hypermedia*, pages 35–44, 2010.
[11] T. Lappas, M. R. Vieira, D. Gunopulos, and V. J. Tsotras. On the spatiotemporal burstiness of terms. *Proceedings of the VLDB Endowment*, 5(9):836–847, 2012.
[12] K. Lerman and R. Ghosh. Information contagion: An empirical study of the spread of news on digg and twitter social networks. *ICWSM*, 10:90–97, 2010.
[13] J. Yang, Y.-G. Jiang, A. G. Hauptmann, and C.-W. Ngo. Evaluating bag-of-visual-words representations in scene classification. In *Workshop on multimedia information retrieval*, pages 197–206. ACM, 2007.