# MoDisSENSE: A Distributed Platform for Social Networking Services over Mobile Devices*

Ioannis Mytilinis, Ioannis Giannakopoulos, Ioannis Konstantinou, Katerina Doka and Nectarios Koziris

*Computing Systems Laboratory, National Technical University of Athens*

{gmytil, ggian, ikons, katerina, nkoziris}@cslab.ece.ntua.gr

***Abstract*—In this work we present MoDisSENSE, a distributed analytics platform for social networking services over mobile devices. MoDisSENSE collects and stores various types of data from heterogeneous sources, such as GPS traces from cell phones, user profile information and comments from social networks connected to the platform. These are combined through spatio-temporal and textual analysis, performed in a distributed fashion, in order to extract knowledge, make smart suggestions and leverage user experience. The datastore follows a hybrid approach to handle both raw and processed data, simultaneously covering the need for scalability and fast query processing. Thus, the platform is able to resolve complex, multi-parameter, socially charged queries over Points of Interest in the order of milliseconds even under heavy load.**

***Keywords*—*Social Networks, Geo-location Services, Sentiment Analysis, Points of Interest***

## I. Introduction

With the advent of powerful programmable mobile devices, online social networking sites, such as Facebook, Twitter and Foursquare, have become increasingly popular. In June 2014, Facebook had on average 654 million mobile daily active users [1]. The introduction of social network APIs has also caused a viral growth to third-party social applications. Third-party developers launch their own applications that utilize social networks' data and offer services that leverage user experience.

To this end, we present MoDisSENSE, a social network based, geo-location service that exploits spatio-temporal and social data. It enables personalized and semantically rich search of Points of Interest (POIs) based on various criteria such as user location, preferences, sentiment of social media friends or a combination of the above. Sample, high level queries that the platform is able to resolve could be "list all meat restaurants offering lamb near Acropolis that my friends have visited and commented positively about in the last month" or "show all places near my current location where most of my friends are gathered right now". Different users are expected to get different results for the same query, according to their social profile.

Moreover, MoDisSENSE analyzes GPS traces from mobile devices to automatically discover new POIs and trending events. Their combination with background information such as maps, check-ins, user comments, etc., results in the inference of a user's semantic trajectory, representing her activities within the day.

*This work has been funded by EU and GR Resources under the Hellenic (GSRT) "COOPERATION 2009" National Action "09SYN-72-881" MoDisSENSE Project.

The aforementioned functionality is supported by a hybrid, highly scalable platform architecture, that handles all these different types of data in massive volumes, processes them and allows for complex queries upon them, achieving response times in the order of milliseconds even when multiple users concurrently utilize the offered services.

MoDisSENSE is currently available as a web application [2] supporting connections to Facebook, Foursquare and Twitter. It is intended to be released as an open source project in the near future.

## II. Architecture

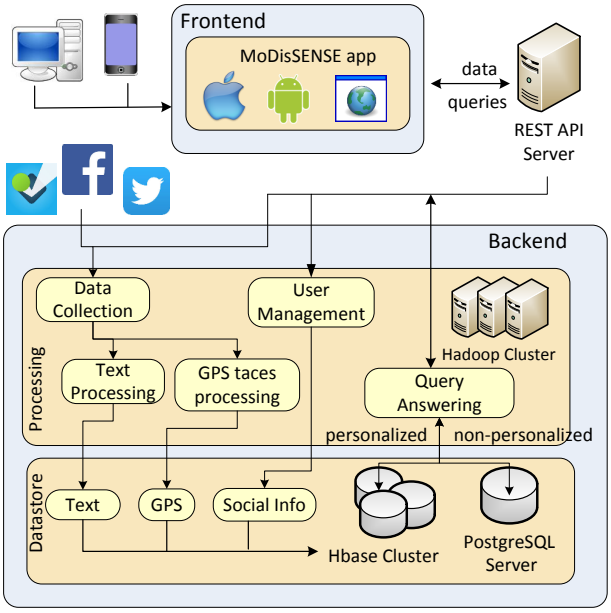


Fig. 1: MoDisSENSE platform architecture.

Figure 1 depicts the overall architecture of the MoDisSENSE platform. MoDisSENSE follows a modular design, consisting of separate, interconnected components. Thus, new components can be added to drive additional functionality without affecting the rest of the system. The two major components that MoDisSENSE is divided into are the frontend and the backend, which communicate through a REST API.

The frontend comprises of the web and mobile applications that the user interacts with. More specifically, MoDisSENSE is offered as a web application supported by the most prominent web browsers and as a native application both for Android and iOS mobile devices.

978-1-4799-5666-1/14/$31.00 ©2014 IEEE

The backend constitutes the platform where data are being stored and processed. The different data types handled by the system are (i) the structured, user-related information provided by the user through the application GUI and enriched through her social media accounts (e.g., friends, POIs visited, etc.), (ii) spatio-temporal data provided by the user's mobile devices (GPS traces) and (iii) textual information extracted by the connected social networks (e.g., status updates, comments, etc.).

On one hand, the structured user-related data require a storage schema that facilitates their efficient retrieval through advanced indexing techniques. On the other hand, the high generation rate of GPS traces and the sheer volume of textual data call for scalability and high read/write throughput. Therefore the platform's datastore follows a hybrid architecture, comprising of both a central database and a distributed NoSQL cluster.

For the former, a PostgreSQL Server [3] is selected, while for the latter an HBase cluster, the open-source implementation of Google's BigTable [4], is deployed. This hybrid approach allows for flexibility and performance: Queries based solely on geo-location and keywords, which demand multiple indices and a large number of joins, are executed in PotstgreSQL, whereas complex queries that delve into the massive amounts of social information are handled by efficient searching techniques applied to the HBase cluster.

These techniques take advantage of the HBase coprocessors feature [5] in order to increase scalability by equally splitting and forwarding the amount of query processing to the servers that contain the respective data. Furthermore, the distributed nature of the NoSQL cluster enables the system to handle large rates of data insertions, without compromising the platform's performance, while ensuring scalability and fault tolerance. The availability of the data is guaranteed by the HBase inherent replication mechanism.

The data processing is performed by a distributed compute cluster, which runs the *Data Collection*, *Text Processing*, *GPS Traces Processing*, *Query Answering* and *User Management* modules. These modules represent the major functionalities of the platform and are thoroughly described subsequently. An important target of the platform is to provide a vivid user experience regardless of the data volume. To this end we rely on distributed techniques to provide high performance, fault tolerance and scalability. As MapReduce [6] is the dominant framework for large-scale distributed processing, we opt for a Hadoop [7] cluster as a substrate for data processing.

### A. Data Collection

The Data Collection module periodically receives GPS traces and collects data from social networks. The GPS traces are generated from the mobile clients of the platform and are temporarily stored in the HBase cluster until they are processed and indexed. These raw traces are indexed with respect to their timestamp and coordinates to allow for fast retrieval based on those fields during their processing.

Moreover, as MoDisSENSE provides social geo-location services, it needs to keep track of information like user check-ins and the accompanying comments, status updates and reviews from social networks to gain the ability of answering socially charged queries. Thus the platform monitors and collects all such information from its users' connected social media accounts. Data collection from social networks is a parallel process, since each HBase node collects data for the users it is responsible for.

### B. Text Processing

The textual data collected from social networks are processed online in real time by the Text Processing module. This type of data includes text that accompanies a check-in, friends' comments, reviews, etc., and is of great importance for MoDisSENSE as it expresses the user's sentiment for a given place. The dominant sentiment of a user's friends about a place is taken into consideration when answering socially charged queries. In order to extract this sentiment, we employ Sentiment Analysis algorithms that make use of Machine Learning classification techniques.

As a classification algorithm, we choose the Naive Bayes classifier that the Apache Mahout [8] framework provides. Naive Bayes is a supervised learning method that needs a previously annotated dataset for its training. In our case, a dataset crawled from Tripadvisor [9] containing reviews for hotels, restaurants and attractions is used for training. The chosen dataset offers two key advantages: First, it is semantically close to our application data and thus results in a high quality training and second, Tripadvisor comments are annotated with a rank from 1 to 5, that can be used as a classification score. After an extensive experimental study and algorithm fine-tuning, we managed to achieve a classification accuracy ratio of more than 90%.

Textual processing is carried out in real time during data collection. This is feasible since the employed classification model is small enough to fit in main memory. Contrarily, the training of the algorithm is an offline process performed in the distributed filesystem due to the massive volume of crawled data.

### C. GPS Traces Processing

The mobile devices that have the MoDisSENSE application installed transmit their GPS traces to the platform. The GPS Traces Processing module runs periodically and processes the traces in order to discover trending events and produce semantic trajectories.

Specifically, the platform utilizes clustering algorithms to identify dense gatherings of GPS traces that could signify events, spontaneous or not, such as concerts, traffic jams etc. Those gatherings may happen to a POI already known to the platform or to an unknown location. Trending events in already known POIs are discovered by considering users' concentration deltas. Large gathering of people in a POI, where a few traces/check-ins had been previously reported, indicates a trending event. For example, traces are not usually reported from football stadiums. However, in the case of a football game or a concert increased trace concentration may be noticed. When an unknown location is the case, dense gatherings imply the existence of a new POI. The identification of the POI may be accomplished through cross-checking with social-network-derived information. If the identification fails, the user is suggested to manually create it through the application UI.

The actual clustering algorithm that the MoDisSENSE platform currently uses is DBSCAN [10], a popular density-based clustering algorithm. To be able to handle millions of concurrent GPS traces, as expected to be the common case, a distributed implementation of DBSCAN over Hadoop [7] is adopted.

Knowing a user's trajectory, MoDisSENSE attempts to semantically annotate it, that is, infer the user's activities during the day in a semi-automatic way. The timestamps of a user's traces help distinguish the ones that may represent an activity, i.e., places where the user spent a considerable amount of time. By comparing the coordinates of such traces and existing POIs, the platform identifies the important POIs along a user's trajectory. If a trace cannot be mapped to an existing POI, the user is prompted for input.

### D. Query Answering

The Query Answering module is responsible for executing user queries, meaning search queries over the collected and processed data. As query input, the user specifies a set of parameters. These parameters can be:

- geographic POI location,
- keywords characterizing a POI (e.g., bar, burger),
- type of POI
- overall expressed sentiment on a given POI,
- only the sentiment expressed by friends on a given POI,
- time window

A search query may contain all of the above parameters or a subset of them. The user can set the desirable geographic location by zooming in and out and adjusting a bounding-box map through the application UI. Only the POIs inside the defined region are considered for the query. A list of keywords characterizing the POI and a list of friends, whose opinion should be taken into consideration, can also be provided. Time constraints may optionally be applied. Typical examples of such queries are the following:

- Which bars do my friends prefer in Athens?
- What is my friends' opinion for this POI?
- How did my friends' opinion for a POI change during last week?

These multi-parameter queries need an adaptive indexing scheme which allows for the efficient execution of such differently configured, concurrent queries. When the query is not socially charged (non-personalized), meaning that friendships in social networks are not taken into account, it is executed in the PostgreSQL database, where we can quickly search for POIs in a specific location and/or apply filters based on their global popularity. Otherwise, when friends' sentiment matters, the NoSQL datastore is employed.

The sentiment-related data kept in the NoSQL cluster are indexed according to the id of each user's friend. This choice enables the parallel execution of the query: A number of workers is launched, each of which searches for the friend ids assigned to it in parallel. Each worker scans all the recorded check-ins of each friend, eliminates the ones that do not fulfill the posed criteria and returns them to a master worker which aggregates the respective POIs. Sorting criteria such as the overall expressed sentiment about a POI or the number of visits can also be used. In such a case, a list of top-k POIs is finally returned to the user. To speed up the process, each check-in (of the same friend id) is stored in chronological order, enabling the platform to quickly scan the data for the necessary interval.

### E. User Management

The User Management module is responsible for authenticating the users to the MoDisSENSE platform and granting access to social networks' APIs. The user is registered to the platform either through a mobile client or the MoDisSENSE website [2]. In order to register, only the social network credentials are used. The registration workflow follows the OAuth protocol [11]. The OAuth authorization framework enables a third-party application to obtain access to an HTTP service on behalf of a resource owner. When the authentication is successful, an access token is returned to the MoDisSENSE platform. From that time on, the MoDisSENSE application can interact with the connected social network on behalf of the end user. It can monitor user's activity, user's friends activity, it can make posts etc. Being an authorized member of the platform, the user can connect to the MoDisSENSE account more social networks. In this case, MoDisSENSE joins information from different social network accounts in order to infer more knowledge and make smarter suggestions.

## III. Conclusion

In this work we present the architecture of the MoDisSENSE platform, a social-network based geo-location service, where users can efficiently execute personalized and semantically rich queries over POIs. Real-time, distributed algorithms are implemented to collect various types of information from heterogeneous data sources and process them to derive useful information. The processing, which includes textual and spatio-temporal analysis, is performed in a distributed environment, perfectly scaling to meet demand. A hybrid datastore approach is followed to properly store and index the diverse data in order to allow for scalability due to their vast amount, provide high write throughput to cope with their rapid generation rate and guarantee fast retrieval to resolve complex queries.

## References


[1] "Facebook Stats," http://newsroom.fb.com/company-info/.

[2] "MoDisSENSE Web App," http://modissense.gr/.

[3] "Postgresql," http://www.postgresql.org/.

[4] F. Chang *et al.*, "Bigtable: A Distributed Storage System for Structured Data," *ACM Transactions on Computer Systems (TOCS)*, vol. 26, no. 2, p. 4, 2008.

[5] "HBase Coprocessors," http://hbase.apache.org/book.html#coprocessors.

[6] J. Dean *et al.*, "MapReduce: Simplified Data Processing on Large Clusters," in *CACM*, 2008.

[7] "Apache Hadoop," http://hadoop.apache.org.

[8] "Apache Mahout," https://mahout.apache.org/.

[9] "Tripadvisor," http://www.tripadvisor.com/.

[10] A. Y. Ng, M. I. Jordan, Y. Weiss *et al.*, "On Spectral Clustering: Analysis and an Algorithm," *Advances in Neural Information Processing Systems*, vol. 2, pp. 849–856, 2002.

[11] "OAuth," http://oauth.net/2/.