Recommendation service for big data applications in smart cities

Georgios Palaiokrassas, Vassilios Charlaftis, Antonios Litke, Theodora Varvarigou
Electrical and Computer Engineering Department
National Technical University of Athens
Athens, Greece
Corresponding author: geopal@mail.ntua.gr

***Abstract*—In this paper we present a recommendation service based on Neo4j graph data base which is applied in the context of smart cities application and leverages the potential of big data. The current work studies a modelling approach for user generated data combined with open big data and proceeds with the appropriate reference implementation and experimentation to validate recommendation services for innovative citizen-centric applications. Moreover, we study and validate performance issues of this Neo4j based recommendation service and evaluate it as a useful appliance for real-time big data application.**

Keywords- Smart cities; Internet of Things; Big Data; Graph data bases; Neo4j; recommendation systems.

I. INTRODUCTION

Modern smart cities are offering innovative applications to their citizens, which are based on the deployed Internet of Things (IoT) infrastructures, the social networks as well as on citizens' initiatives to create mobile applications based on various open data repositories. This new paradigm of innovative smart city applications comes with a wide set of technical concerns and limitations that brings barriers to the developers. Among others the various issues that hinder the development of such applications is how to tame the big data provided by the IoT infrastructures, how to make meaningful added value services for the citizens based on these data. The current paper provides a solution which is based on smart city open data provided by its IoT infrastructure (network of sensors) and enables the creation of recommendation services for outdoors activities to the citizens based on the preferences of similar users. In order to handle the complexity of the data and deliver recommendations based on dynamic criteria provided by the users in real time, the proposed solution is utilizing the Neo4j graph data base.

A graph database is a database that uses graph structures for semantic queries, with nodes and edges to represent and store data. A basic concept of these bases is the graph (with nodes and edges), which directly correlates data items in the store. These relationships allow data to be linked together directly, and in most cases, are retrieved in a single operation. This contrasts with conventional relational databases, where relationships between data stored in the data itself, and the search queries for this data using the concept of JOIN tables for the recovery and collection of relevant data. The graph databases allow, by their design, fast retrieval of complex hierarchical structures that are difficult to model in relational systems. For this reason, they have begun in recent years to be widely used in both social network applications and recommendation systems as they better reflect their loose structures and enable them to evolve and escalate with more flexibility.

In this paper, we build a graph database application using Neo4j and we study ways to implement queries and traversal techniques that enable value-added services based on machine learning. Specifically, we build a Neo4j graph database that stores data from sensors located in a smart city (air pollution concentrations, temperature, relative humidity and other open data sources) and we will implement recommendations queries for potential users of applications using the database.

The paper is structured as follows: section II provides an analysis of related and background work upon which this paper has been based. Section III provides an overview of a smart city context (focusing on the BigClouT project) that forms the basis for modern smart cities which have demanding requirements on deploying big data based applications offered to the citizens through smart phones. Section IV presents a suggested graph modelling approach for big data, while sections V and VI give the details of the followed implementation for the recommendation service on top of the Neo4j graph data base and the experiments/validating test that have been performed.

II. RELATED WORK

In the recent years, Graph Databases have been applied in many domains such as semantic web, social networking and recommendation systems. They can efficiently handle big data problems, involving complex interconnected information, capture semantics, store, retrieve and manipulate big volumes of complex data [1]. By supporting the interrelation of data, graph databases permit the fast traversals along the edges between vertices and as models optimized for processing dense, interrelated datasets allow the detection of correlations and patterns [2][3][4]. A key design characteristic for a database is its compliance to ACID (atomicity, consistency, isolation, durability) properties. Neo4j Graph Database is a

978-1-5386-3250-5/17 $31.00 © 2017 IEEE
DOI 10.1109/HPCS.2017.41

fully ACID transactional database [1], guaranteeing reliable transactions and data integrity .

In [5] a data model was introduced for time-varying social networks represented as graphs in the Neo4j graph database and collected data using wearable sensors. They found graph databases very well suited to their use case due to the support for the property graph data model, the persistent, transactional storage of graphs, the support for deep graph analytics via efficient many-hop traversals and the usefulness of Cypher [6][7] declarative query language as well. As they observed Neo4j proved to perform well when querying the sample dataset, performing exploratory data analysis, and research-oriented data mining.

Most graph-based methods create only an edge between nodes to describe the interaction between a user and an item. However these methods ignore the contextual information which may be crucial in some applications. It is observed that a user under different circumstances shows different preferences, so a recommendation system should take into consideration the decision context and the available additional information such as time, weather and location [8][9].

A variety of techniques have been proposed as the basis for recommender systems: i) collaborative filtering approaches build a model from a user's past behavior (ratings, items purchased) and predict items (or ratings for items) that the user may have an interest in; ii) content-based filtering approaches utilize a series of discrete characteristics of an item in order to recommend additional items with similar properties; iii) demographic approaches provide recommendations based on a demographic profile of the user; iv) Hybrid recommender systems combine multiple techniques together to achieve some synergy between them [10][11]. Towards the direction of Hybrid system is the technique of BellKor's team which won the Netflix Prize and combined 107 different algorithmic approaches recognizing that accuracy is substantially improved when blending multiple predictors [12].

Much research work has been conducted in the area of recommendation systems and their convergence with other domains such as Smart Cities [13][14] and in their applications as well. The proposed methods exploit the provided information to deliver a recommendation [8]. In [15] authors developed a graph-based recommender system in a digital library environment which contains rich information of books' content, users' usage history and demographic data. They combined content-based and collaborative recommendation approaches to make recommendations and provide value-added services to users [16]. In [9] user's personal and social information were incorporated to improve the quality of ranked list of items. Applied in Smart Cities and tourism, different systems have been designed considering different kinds of interfaces, recommendation approaches and functionalities [17]. Luberg, Tammet and Jarv [18] developed a personalised recommendation system using probabilistic reasoning. Tourists provided their profile with interests and the suggestion engine created a plan for visiting interesting objects and events on the trip. In [19] a personalized city transport advisory system was introduced. Recommendations for personalized paths were using a knowledge-based recommendation technology, and for each user the suggestions were computed according to their travel-related preferences. The concepts of travel and user profiles were introduced in order to rank different routes in the city and provide the top ranked to a user. Each time the user requested a recommendation for a travel he was asked to specify his preferences (walking, public transportation, sightseeing, time etc.).

III. SMART CITY CONTEXT: THE EXAMPLE OF BIGCLOUT

The goal of a smart city is the satisfaction of various social needs of its citizens and the improvement of their life conditions in the urban environment. Modern cities, by their nature, generate vast amount of data (e.g. traffic data, air pollution data, weather conditions, energy consumption metering, etc.). This comprises what we call today as Big Data: collections of complex and dynamic data that are gathered by various sensors or services across the smart city. The value of such Big Data relies on the fact that they can be managed and analyzed in ways that offer significant knowledge for the decision making processes of the various involved stakeholders in the smart city (e.g. public authorities, citizens, visitors etc.). The intelligent and effective usage of these data is a significant challenge for every smart city.

A first step for the exploitation of Big Data is to let them be open while enabling the free, public access to them, creating thus what we call Open Data (OD). Smart city's OD enhance innovation by enabling the creation of smart city applications by citizens, entrepreneurs, open source community and others. Especially the combination of different OD sources can lead to new, creative applications that contribute to the safety, easiness, health and prospect of the smart city inhabitant (for example, air pollution OD can be combined with OD of transportation, road traffic, weather conditions, or social network applications).

An example of such an approach for smart cities is the BigClouT project [20]. BigClouT will in particular make use of today's three key technologic enablers, IoT, cloud computing and big data, for the objective of increasing the efficiency in using urban infrastructure, economic and natural resources shared by the increasing population. BigClouT will offer an analytic mind to the city by creating distributed intelligence that can be implanted throughout the whole city network either for large or smaller urban areas. The project leverages the results of the EU-Japan ClouT project [21] and will bring them beyond, by adding for instance distributed intelligence with edge computing principles, big data analytics capability, in addition to self-awareness and dependability properties towards a programmable smart city platform.

The BigClouT project brings together four smart cities (Grenoble-Alpes, Bristol, Tsukuba and Fujisawa) and six use cases, such as monitoring of economic impacts of events, smart energy, mobility prediction of citizens, providing traffic and environmental information in real time to visitors to Tsukuba and others.

[1] https://neo4j.com/developer/graph-database/

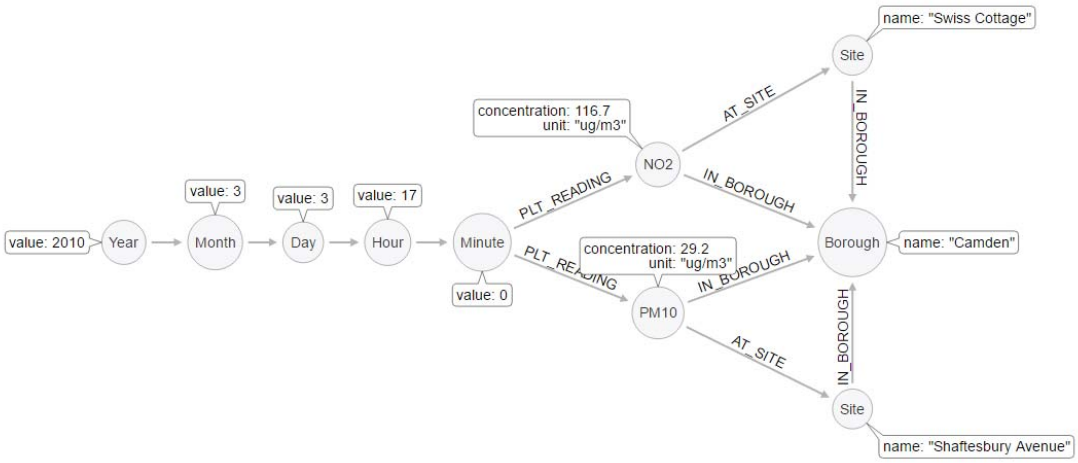


Figure 1. Linking boroughs, sites and air pollutants in the Neo4j time tree.

IV. Graph Database Modelling Approach for Big Data

A. The smart city use case adopted

The implementation in the current paper is based on a state-of-the-art smart city application, which is based on open data captured by the city IoT infrastructure and user generated content in terms of user preferences for various activities. Such an application is regarded as a lifestyle application for a particular city context. The open data that are being deployed refer to: i) air pollution measurements as collected in particular boroughs of the city, along with their characterization in scales (e.g. low, moderate, high, very high); ii) weather conditions as collected by the weather stations (measuring wind speed, rain fall, humidity and temperature); iii) user profile data and iv) user preferences in terms of activities (cycling, walking, gardening and others) along with a dynamic rating mechanism where users give current feedback giving a score (from 0 to10) according to if they find the current conditions appropriate for the specific activity.

B. Overall moddeling approach

A significant advantage of graph models is the fact that they depict entities and relations in the same way as we think of them. Moreover, these models are forming a view on the queries we would like to implement in the graph data base. The modelling process and approach in graph databases can be regarded as equivalent to the approach of creating graph structures that reflect the queries we would like to answer. "Users" or "Activities" for example comprise the nodes of the graph, "names" are the attributes of the nodes, verbs such as "likes" depict the relations that link the nodes, and whatever refers to such verbs is regarded as the attributes of the relations.

An interesting way to model time in Neo4j is through the use of time trees. In this approach nodes represent years, months, and days, etc. while every node contains an attribute "value". By forming the time trees, we can link particular events or other measured data on these trees. In our example, we link measurements of air pollution (comprising of the five pollutants NO2, O3, SO2, PM10 and PM25) which have been collected in the Camden borough of London and are provided as open data [22][23]. The data have been fed in our graph data base and linked together in an efficient and constructive way allowing the user to ask a multitude of questions such "what is not the level of air pollution", "what are the weather conditions" etc. At the same time users have been added in the data base along with their preferences with respect to various activities.

V. Graph Database Implementation

A. Database Population

Based on the data modeling previously presented, we begin the implementation of our system using Neo4j's query language Cypher. It is a declarative, SQL-inspired language for describing patterns in graphs visually. It allows us to state what we want to select matching a specific pattern, insert, update or delete data from our graph without requiring us to describe exactly how to do it [6][7].

We initiate the implementation by creating the time trees, which constitute the keystone of our model. In order to achieve faster data retrieval and improved performance we use indexes, a feature provided by Cypher, created once over a property for all nodes that have a label. Cypher automatically manages the update by the database whenever the graph is changed.

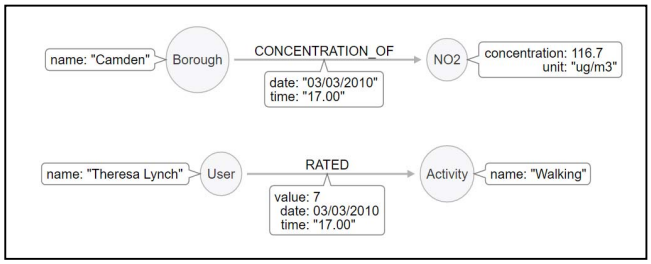


Figure 2. Example of modelling in Neo4j (model for the concentration of air particle in a specific borough and time as attribute in relations between nodes.

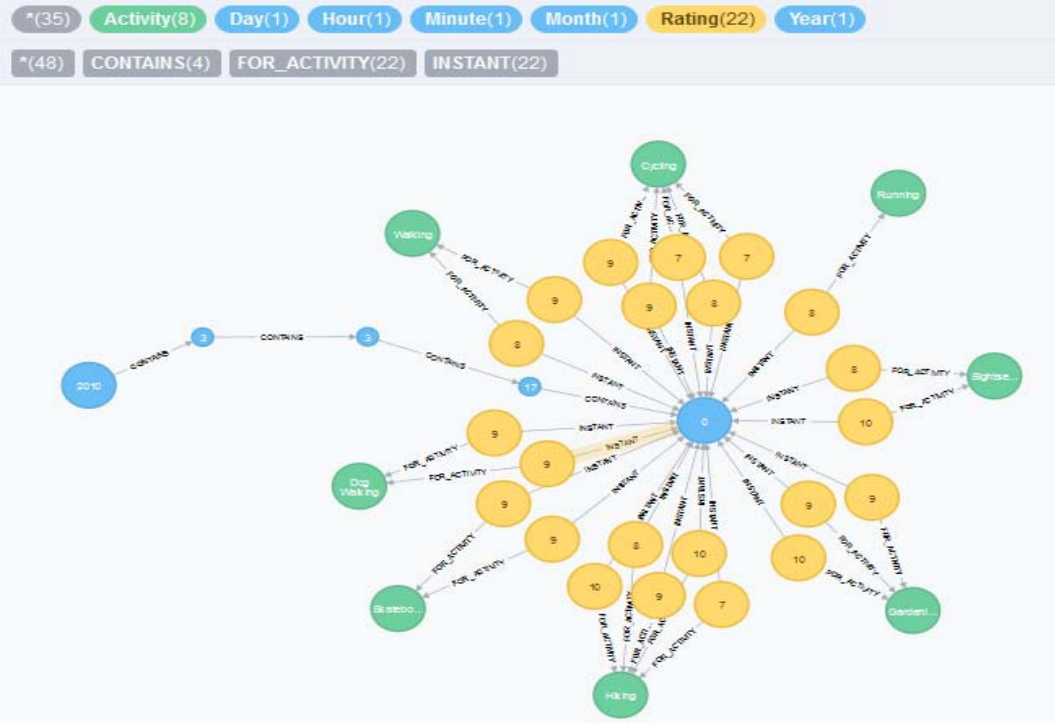


Figure 3. Ratings of activities provided by users of the system.

Followingly, we populate our model by importing air pollution concentration data into the graph database, as shown in Fig 2. In a similar way we import the temperature and weather data linked to the suitable time nodes, adding additional integration layers when required e.g. an additional layer is required for wind level and speed. We should note that Neo4j offers great data visualization options allowing us to view all the stored nodes, relations and information, giving real-time insights into how data nodes are related, facilitating the development and evaluation process.

B. User Registration

The system provides a user friendly way for users' registration in order to deliver them recommendation services. When a user is registered to the system a new unique node is created with his demographic data and characteristics stored as properties. Afterwards the user inserts his preferences regarding environmental conditions, indoor, outdoor activities and the corresponding nodes and relations are created in the graph database. Our model includes a weight scale for all preferences from 0 to 10 instead of a simple binary field. This way the user gives a more detailed profile and in the end a more personalized recommendation is delivered.

The next step is the addition of evaluations regarding past and real time activities. Every evaluation node is connected to the user who provided it, the corresponding node in the time tree and borough. This way a specific rating is connected to all the related information of our model.

VI. EXPERIMENTAL TESTS - RECOMMENDATIONS

Our proposed system executes queries in the database to receive answers to simple or complex questions and produces a recommendation based on stored data and dynamic information, supporting different types of recommendation.

One of the most remarkable features of a graph database, which is also crucial for our implementation, is its effectiveness into handling big volumes of information with real time response to queries. Our system based on a highly scalable graph database processes in real-time simple and complex questions such as "What is the temperature on a specific day/time?", "What is the level of air pollution and weather on a specific timestamp?" by rapidly traversing the graph while reading and processing values in nodes, relations and properties.

Similarly, it handles other questions related to the preferences of users regarding outdoor activities and environmental conditions. Questions about ratings on past activities the user has attended are effectively handled, by traversing the graph as shown in Fig. 4, where the activities' ratings of users are depicted.

Recommendation systems produce suggestions through collaborative, content-based filtering or personalized approaches. There are also approaches taking into consideration social and demographic information. In many cases recommender systems not only take into account the preferences of users but they also analyze the dynamic context.

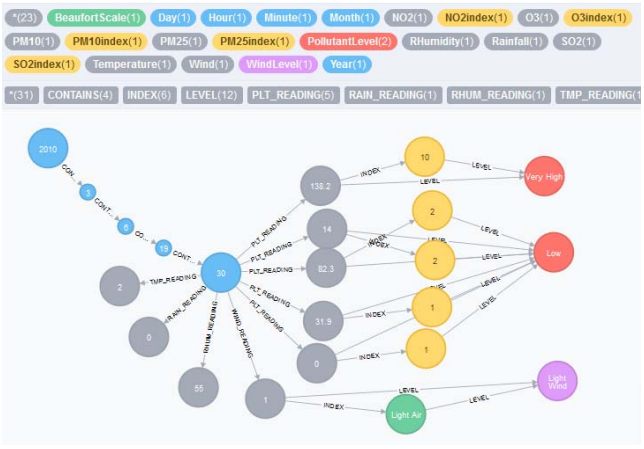


Figure 4. Graph representation for measurements of air pollution and weather. The nodes of the time tree specify a timestamp related to the current values of air pollution concentrations and weather.

The context can include aspects like the location, time and weather [17]. There are also methods which combine different approaches together called hybrid. In all cases the recommendation systems establish relations among users and target objects. The production of effective recommendations is based on the comprehension of these relations and their quality as well. Graphs are perhaps the most suitable structure to represent dense connected data structures which come as an outcome of the previous mentioned requirements. By storing and studying these data using graph databases, an application is allowed to exploit and demonstrate in real time the impact of users' actions and not just take advantage of predefined results of pre-existing data. The system delivers real time recommendation following different approaches as will be presented below.

A. Extracting current trends and popular activities

Our first recommendation approach is quite useful in cases where there are not sufficient data for a user such as his profile or previous activities he participated in and scored. Therefore a personalized suggestion is not possible and a more generic technique is followed: the clustering of users' rating and the computation of average score for each activity. As a result, the most popular activities among all users at the current moment are returned.

B. Personalized recommendation based on user's feedback and preferences

Despite the usefulness of the previous approach, its ability to extract the public disposition, the "average" user's preferences and effectively tackle the lack of data, this recommendation is not personalized. For example a particular user has scored "Outdoor Photography = 0" so the suggestion "Outdoor photography" is inefficient.

Personalization techniques aim to provide customized information to users based on their preferences, restrictions or tastes. They are particularly relevant in recommender systems, whose objective is to filter irrelevant options and to provide personalized and relevant information to each particular user[8][17]. Our second approach takes into consideration user preferences, excluding activities he has down-voted and promote results closer to his preferences. This technique is useful when users want to receive information specifically related to their designated preferences.

C. Using demographic data and social information

Despite the fact that the returned results of the previous method are certainly close to user's references there is a drawback: the recommendations are limited to a particular subset of activities and we can't suggest new ones that he might find satisfactory. In order to solve this problem and produce a more targeted recommendation we exploit the available demographic data, provided by the users of the service during the registration. One application is to take into consideration only the ratings registered by users of a certain age group e.g. who have 5 years difference with our target user.

D. Exploiting similarity metrics and vector representations to deliver recommendations

A widely applied technique in collaborative filtering recommendation systems is identifying the closeness among different users with similarity metrics. To this direction we model application users as vectors, by specifying the criteria and then compute the distance between them. This way we locate users who are closer to the target-user, study their behavior and make corresponding recommendations. We have used several similarity metrics and users modeling techniques. We will present two similarity metrics (Euclidean and Cosine) and a user representation technique as a vector based on his activities preferences. We will also evaluate the effectiveness of this approach by executing queries in our graph database.

In the following example a user is represented as a vector taking into consideration the rating on outdoor activities he has provided: User = <$Activity_1$, $Activity_2$, …, $Activity_N$> where $Activity_i$ could be outdoor photography, walking e.t.c. taking values from 0 to 10, according to the scores provided. For example a user could be represented with the vector: User1=<10, 0, …, 7>. In figure 5 that follows the scoring of three outdoor activities leads to vectors with three values which represent users.

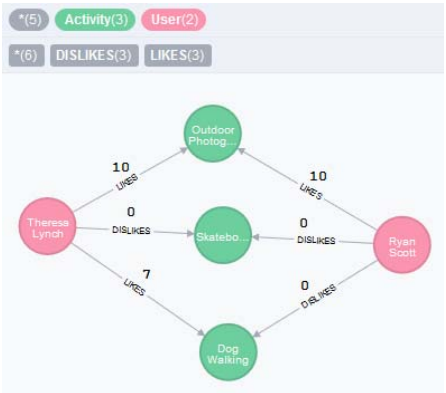
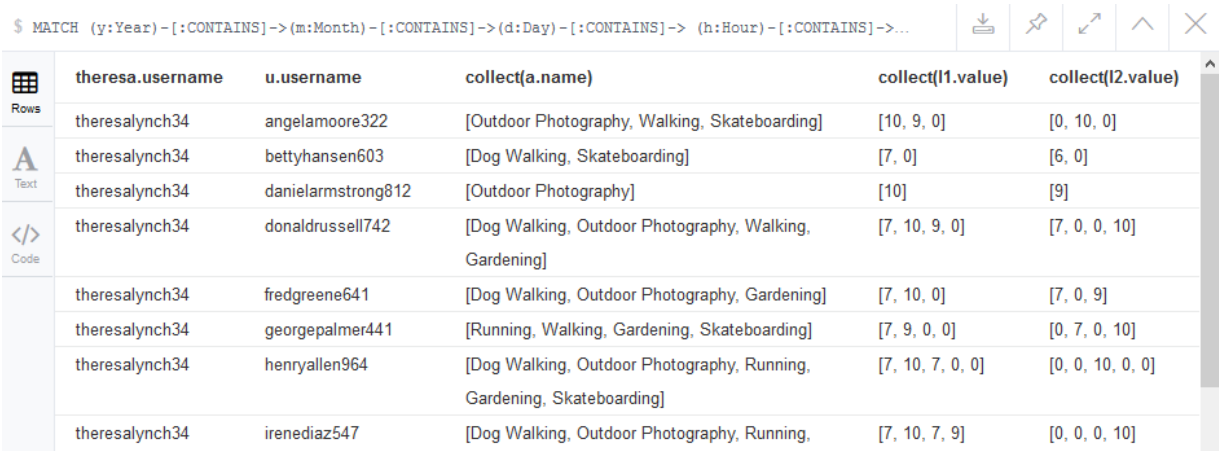


Figure 5. Example of user preferences regarding outdoor activities



theresa.username	u.username	collect(a.name)	collect(l1.value)	collect(l2.value)
theresalynch34	angelamoore322	[Outdoor Photography, Walking, Skateboarding]	[10, 9, 0]	[0, 10, 0]
theresalynch34	bettyhansen603	[Dog Walking, Skateboarding]	[7, 0]	[6, 0]
theresalynch34	danielarmstrong812	[Outdoor Photography]	[10]	[9]
theresalynch34	donaldrussell742	[Dog Walking, Outdoor Photography, Walking, Gardening]	[7, 10, 9, 0]	[7, 0, 0, 10]
theresalynch34	fredgreene641	[Dog Walking, Outdoor Photography, Gardening]	[7, 10, 0]	[7, 0, 9]
theresalynch34	georgepalmer441	[Running, Walking, Gardening, Skateboarding]	[7, 9, 0, 0]	[0, 7, 0, 10]
theresalynch34	henryallen964	[Dog Walking, Outdoor Photography, Running, Gardening, Skateboarding]	[7, 10, 7, 0, 0]	[0, 0, 10, 0, 0]
theresalynch34	irenediaz547	[Dog Walking, Outdoor Photography, Running,	[7, 10, 7, 9]	[0, 0, 0, 10]

Figure 6. Execution of Cypher code snippet produces results with users, activities and representation vectors

We transform the problem of finding similar users into the equivalent problem of finding the closest vectors representing users. To this direction we compute distances between vectors using metrics. The first presented metric is Euclidean distance and we compute the distance of two points in the n-dimensional space:

$$d(\boldsymbol{x},\boldsymbol{y}) = \sqrt{(x_1-y_1)^2+(x_2-y_2)^2+\cdots+(x_n-y_n)^2} = \sqrt{\sum_{i=1}^{n}(x_i-y_i)^2} \quad (1)$$

The equivalent computation in Cypher programming language is written as follows:

```
MATCH
  (u1:User)-[x:LIKES|DISLIKES]->(a:Activity),
  (u2:User)-[y:LIKES|DISLIKES]->(a)
WHERE u1.username = "use1" AND
  u2.username = "user2"
RETURN sqrt(sum((x.value - y.value)^2));
```

We now form complex Cypher queries delivering recommendations in real time using the Euclidean distance to compute the similarity among a target-user and all users in the graph database who have ranked activities. The following Cypher query refers to Fig. 4 and goes through the steps: i) find the time node for this particular moment; ii) aggregation of all users who have scored an activity in the time window under study; iii) production of vectors representing users; iv) computation of distance between users using Euclidean distance; v) sort results in an increasing order; vi) choose the closest users; vii) deliver recommendations based on these users activities. Because of space limitations and since our source code is available[2] we present only the third point and the results obtained:

```
MATCH
  (y:Year)-[:CONTAINS]->(m:Month)-
  [:CONTAINS]->(d:Day)-[:CONTAINS]->
  (h:Hour)-[:CONTAINS]->(min:Minute)-
  [:INSTANT]-(r:Rating)<-[:GAVE]-(u:User)
WHERE
  y.value=2017 AND m.value=3 AND d.value=3 AND
  h.value = 17 AND min.value = 0
WITH u, collect(r) AS ratings
MATCH
  (user1:User {username: "user1"})-
  [l1:LIKES|DISLIKES]->(a:Activity),
  (u)-[l2:LIKES|DISLIKES]->(a)
RETURN
  user1.username, u.username,collect(a.name),
  collect(l1.value), collect(l2.value)
ORDER BY u.username;
```

After executing this code snippet the output with users, activities and vectors is returned, which is later filtered and sorted as described above resulting in three activities as a recommendation.

The second metric is cosine similarity. We compute the cosine of the angle between the vectors in the n-dimensional space using the mathematical form:

$$similarity = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \quad (2)$$

The cosine between two vectors takes values in the range[-1,1] and we follow a similar approach to produce a recommendation as in Euclidean distance.

Fig. 6 depicts a part of the results after finding the users, activities and representation vectors. The distances among them are calculated using these vectors and then sorted in an increasing order. For the example user of this figure who has shown particular interest in walking and outdoor photography the closest user is currently hiking and this is the activity suggested. This method is very effective since it allows us to produce recommendations based on the current popular activities of the closest users to the target one.

Contrariwise the first described approach (popular activities) captures the current trends among all the users delivering a less personalized suggestion, but informing the user regarding current popular events such as an outdoor

[2] https://drive.google.com/open?id=0B8RfqpH7xrOKeTlpTTVNaXJHQW8

concert or a gathering. For the specific user one or two out of the top three recommended activities were disliked by her during her initial registration form. However, under a certain context she might consider them interesting choices. Exploiting social and demographic information we noticed that certain activities are recommended to people of all ages (e.g. hiking, dog walking), while some activities like skateboarding are mostly recommended to young people.

E. Real time response of the suggestion service

It is important for a recommendation service to maintain a reasonable response time and particularly the suggestion calculation to have a limited running time [18]. We should note that our system works in real time. We have extensively tested the efficiency of our system for increasing number of users, data and complex queries and it returns high speed results. Exploiting the features of graph data bases, it achieves for the big majority of queries responses in less than 100 msec.

VII. CONCLUSIONS

The current work presented in this paper, successfully combines open big data with user generated data captured through smart phone applications in order to implement an efficient recommendation service on top of the Neo4j graph data base. The exploitation of resulting big aggregated datasets poses multiple challenges, with timely-efficient analysis being the most important. Focusing on effective data modelling, management and multiple techniques for recommendation services on top of Neo4j in the experiments performed, we can validate our approach as an efficient one for building innovative applications for citizens in smart cities. In the future the authors plan to use even larger and more complex datasets, further leveraging on the effectiveness of the social networking services and experiment additionally with new traversal techniques on top of graph data bases.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Commission under the H2020 Programme's project BigClouT (grant agreement nr. 723139).

REFERENCES

[1] J. J. Miller, "Graph database applications and concepts with Neo4j," in Proceedings of the Southern Association for Information Systems Conference, Atlanta, GA, USA, 2013, vol. 2324, p. 36.

[2] M. A. Rodriguez and P. Neubauer, "Constructions from dots and lines," Bull. Am. Soc. Inf. Sci. Technol., vol. 36, no. 6, pp. 35–41, 2010.

[3] A. Silvescu, D. Caragea, and A. Atramentov, Graph Databases. Citeseer, 2002.

[4] M. A. Rodriguez and P. Neubauer, "The graph traversal pattern," ArXiv Prepr. ArXiv10041001, 2010.

[5] C. Cattuto, M. Quaggiotto, A. Panisson, and A. Averbuch, "Time-varying Social Networks in a Graph Database: A Neo4J Use Case," in First International Workshop on Graph Data Management Experiences and Systems, New York, NY, USA, 2013, p. 11:1–11:6.

[6] https://neo4j.com/developer/cypher-query-language/

[7] I. Robinson, J. Webber, and E. Eifrem. *Graph databases: new opportunities for connected data*. " O'Reilly Media, Inc.", 2015.

[8] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions," IEEE Trans. Knowl. Data Eng., vol. 17, no. 6, pp. 734–749, Jun. 2005.

[9] W. Yao, J. He, G. Huang, and Y. Zhang, "SoRank: Incorporating Social Information into Learning to Rank Models for Recommendation," in Proceedings of the 23rd International Conference on World Wide Web, New York, NY, USA, 014, pp. 409–410.

[10] F. Ricci, L. Rokach, and B. Shapira, Introduction to recommender systems handbook. Springer, 2011.

[11] https://en.wikipedia.org/wiki/ Recommender_system

[12] Y. Koren, "The bellkor solution to the netflix grand prize," Netflix Prize Doc., vol. 81, pp. 1–10, 2009.

[13] F. Aisopos, A. Litke, M. Kardara, K. Tserpes, P. Martinez , T. Varvarigou, "Social network services for innovative smart cities: the RADICAL platform approach. Journal of Smart Cities" , 2016 vol.2(1): 26–40.

[14] E. Psomakelis, F. Aisopos, A. Litke, K. Tserpes, M. Kardara and P. Martinez Campo, "Big IoT and Social Networking Data for Smart Cities: Algorithmic improvements on Big Data Analysis in the context of RADICAL city applications.", Workshop on Towards Convergence of Big Data, SQL, NoSQL, NewSQL, Data streaming/CEP, OLTP and OLAP, 6th International Conference on Cloud Computing and Services Science (CLOSER 2016).

[15] Huang, Z., Chung, W., Ong, T. H., & Chen, H. (2002, July). A graph-based recommender system for digital library. In *Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries* (pp. 65-73). ACM.

[16] W. Yao, J. He, G. Huang, J.Cao, Y. Zhang, "A Graph-based model for context-aware recommendation using implicit feedback data." *World wide web* 18.5 (2015): 1351-1371.

[17] J. Borras, A. Moreno, and A. Valls, "Intelligent tourism recommender systems: A survey," *Expert Syst. Appl.*, vol. 41, no. 16, pp. 7370–7389, 2014.

[18] A. Luberg, T. Tammet, and P. Järv, "Smart city: a rule-based tourist recommendation system," in Information and Communication Technologies in Tourism 2011, Springer, 2011, pp. 51–62.

[19] G. Tumas and F. Ricci, "Personalized mobile city transport advisory system," Inf. Commun. Technol. Tour. 2009, pp. 173–183, 2009.

[20] http://bigclout.eu/

[21] http://clout-project.eu/

[22] http://www.londonair.org.uk/LondonAir/Default.aspx

[23] https://uk-air.defra.gov.uk/air-pollution/daqi