International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 08 Issue: 06 | June - 2024 SJIF Rating: 8.448 ISSN: 2582-3930

# Location Based Personalized Restaurant Recommendation System

Priyanka Prakash
2000300100167
Indraprastha Engineering College
Ghaziabad, India
priyankaprakash410@gmail.com

Anurag Kumar Kaushal
2000300100040
Indraprastha Engineering College
Ghaziabad, India
anshrag101@gmail.com

Ankit Kumar Maurya
2000300100028
Indraprastha Engineering College
Ghaziabad, India
mauryaankit804@gmail.com

***Abstract*—Restaurant recommendation system is a very popular service whose accuracy and sophistication keeps increasing every day. With the advent of smartphones, web 2.0 and internet services like 5G, this has become accessible by every consumer. In this paper, we present a personalized location based restaurant recommendation system integrated in mobile technology. It ubiquitously studies the user's behavioral pattern of visiting restaurant using a Machine Learning algorithm. We also address the issues faced by today's recommendation systems and propose methods to rectify it.**

***Keywords—Machine Learning; Recommendation System; Location Based Service; M-commerce; foursquare;***

## I. INTRODUCTION

Mobile has become the most common means of introducing new and developing technologies to the masses. The current technologies present elsewhere are also being transferred to mobile, with addition of mobile technology features like ubiquity, personalization and localization [1].M-commerce has also found its way into mobile from the success of e-commerce, along with the adaptation of most of the mobile technologies to provide a more enhanced shopping experience.

Among some of the mobility features of M-commerce, recommendations systems also play a very important role as they are an indispensable part of personalization. The term 'recommender systems' or 'recommenders' has extensively evolved from the term 'collaborative filtering' since the latter is a specific algorithm for recommendation. Recommender systems take into account the reviews and experiences of a community to produce for the user a mapping of scores to items, a ranking of items, a set of recommended items, or these three combined [2]. They can dramatically reduce the amount of information a user has to go through to search a particular item. Recommendation systems have a come a long way since they were developed, as they work upon item ranging from shoes, clothes to dining places.

In this paper, we introduce a new mobile [illegible] that infuse personalization, ubiquity [3] and location based service to take a user's dining experience to the next level. We would be developing a recommendation system that would use machine Learning algorithm to study the user's behavior as he keeps checking in restaurants through any popular social network. Information extraction (IE) is the task of locating specific pieces of information from a document, thereby obtaining useful structured data from unstructured text [4]. We would use Foursquare to extract all our data and post the user's visits in it. That is because, firstly, a social driven data set solves most of the problem of missing or unreviewed data, and secondly people find it comfortable if they know and can access the source of data and how their information is being used. Foursquare is one of the most dominant social-driven location sharing application with its users having several reasons to use it, ranging from elements of fun, exploration, coordinating with friends to using history of places and checking-ins [5].

This paper is structured as follows: Section 2 presents the current drawbacks of the systems and how we eradicate it in our system. Section 3 highlights the basic system architecture. Further, sections 4 and 5 elaborate the important aspects of the architecture. The implemented work and the functional flow of system are elaborated in Section 6. Section 7 outline future work and minor improvements that can be made and the contribution are finally concluded in section 8.

## II. ISSUES RELATED TO RECOMMENDATION SYSTEMS

### A. *Cold Start*

In current Content Based recommendation systems, when a user first starts using it, the system knows nothing about his preferences [6]. Consequently, the system is unable to present him any personalized recommendations. This is called the cold-start problem of recommender. There are cold start problems for both new users and new items. In this paper, aim at eradicating this problem by retrieving the user's past visits of restaurants using his foursquare account. If the user doesn't have an account on foursquare, then he

978-1-4673-6217-7/13/$31.00c 2013 IEEE

can login through his Facebook account, and the system can retrieve his friends' foursquare check-ins, though this then becomes a Collaborative filtering in the beginning, but slowly the system would again move to Content-based.

*B. Unclassified Restaurants*

Usually, recommendation systems extract their content from a trusted online source. These sources, in most cases, are not able to locate very small and road-side food joints, which might, in fact, be very popular among the local people. Thus, we use a social location sharing source, where people can declare restaurants at certain coordinates. These new declarations are visible by other people too on the network, and number of check-ins for the place increase, though it is not officially recognized. So our system takes account of these places while gathering venues near the user's location.

*C. Constant user participation*

Most of the recommendation systems, especially, Knowledge Based Recommendation Systems, prompt the user constantly for category and other attributes of the restaurant. Also the user has to rate and review items, and share his location on social networks. The user has to manually start the system whenever he needs the service. We have tried to eradicate all these issues and let the user sit back and relax after just logging in once through his Foursquare credentials. The system would use push notification service to constantly recommend places to user, when he is on move. The system would automatically share his location on social networks, depending on the settings he has provided.

## III. THE SYSTEM ARCHITECTURE

The motive of this paper is to present an implemented design of the recommendation system. The system architecture presented in Figure 1 can be simply divided into two sections, one which has online activity, and the other which processes data offline. When the user is in motion, i.e., his geo-position changes notably, the system goes online and recommendation module becomes active, retrieving nearby and restaurants and ranking them, based on their properties, according to the scores generated offline. The offline part generally remains in a non-functional mode when the user is stationary The work of the offline system is to generate a user interest profile, using a Machine Learning algorithm, from the data set that keeps getting modified whenever the user checks-in a restaurant. This autonomous switching of the system between online and offline modes guarantee that there is no wastage of power and bandwidth when it is not required.

Based on the type of functionalities [7, 8], the system can be divided into different modules, i.e., database layer, recommendation engine, user profile generator and online interaction layer.

The user will be tracked through Web 2.0 technologies [9] namely HTML5 and the data set will be retrieved simultaneously from Foursquare. Based on the user's check- in to any restaurant, the data will be added to the restaurant data set and will then be evaluated for user interest profile. This application will mostly be running in the background, as participation from user is avoided as much as possible.

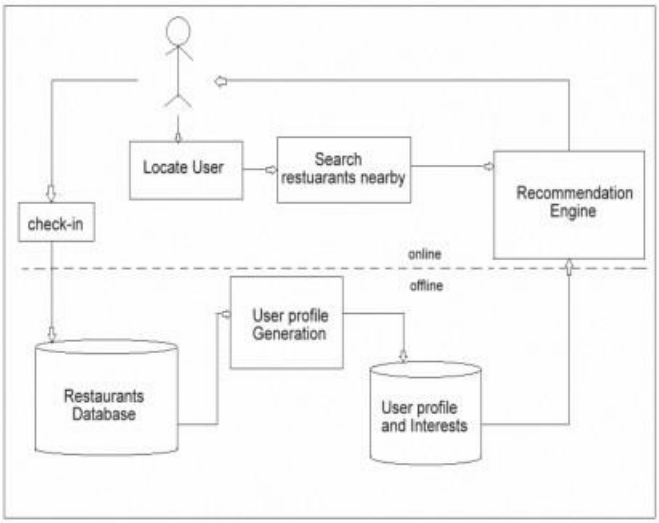


Figure 1. System Architecture

## IV. LEARNING USER PROFILE

The system solely focuses on the user's interests and does not take into account the reviews of other people. Since if there may exist such a case that a restaurant may not be preferred by the majority but only cater to the needs of a few, including the user, then the system may give a negative remark for the place, thus misguiding the user.

Restaurants are classified as 'like' or 'dislike', depending on the taste of the user [11]. We build this taste by evaluating the list of restaurants already liked by the user. Here, we make an initial assumption that the user likes all those restaurants if he makes a check-in in Foursquare. Each restaurant would account for n entries, where n is the number of check-ins made. Further, if he likes two restaurants, which one would he favor more? We solve this problem by using Naïve Bayes Classifier algorithm to recognize the factors that the user likes about a restaurant and to what degree does he like them. The simplicity and performance of this algorithm, as compared to other classification methods, is far superior [6]. Among the two formulations of Naïve Bayes, we have found out that multinomial Bernoulli model would be more efficient in this context, as its calculations are based on word frequency information [10,15].

Let each restaurant object have feature vectors or attributes. These are the properties of the object according to which the classification is done. The standard assumption made is that the object or document contains a bag of words [4], i.e., all the words are considered independent. But we modify this assumption in our system and consider each feature as a single word or token. Because in some cases, a feature may describe a restaurant with several words, and

splitting them would not make sense. For example, the category of a restaurant may be 'South Indian' or a location might be 'Mahatma Gandhi Road'. With the independence assumption, we can come up with various inductions related to the user's behavior.

Let the feature vector set be X = {category, location} and the set of classification classes be C = {*like*, *unlike*}. The total weightage of all the features present in the training data is calculated as follows

$$P(X|\{like\}) = P(x_1|\{like\}) \cdot P(x_2|\{like\}) \cdot \ldots \cdot P(x_n|\{like\})$$

$$\text{where } P(x_i|\{like\}) = \frac{N(xi,\{like\})}{N(\{like\})}$$

where N(*i*) is the number of instances of *i* in the dataset. Since, we retrieve only the list of restaurants liked by the user, we already assume that all of the features are liked by him. Now to avoid the maximum likelihood problem [16], we have to redefine the previous equation as

$$P(x|\{like\}) = N(x_i, \{like\}) + \frac{1}{N(\{like\})} + V$$

where V is the total number of keywords occurring in the database. These probabilistic values are calculated and stored in the database, so that when the new dataset is retrieved, these values can be easily used for generation for recommended list. The calculations involving these equations are iteratively done on every check-in.

## V. Recommendation Engine

In this module, a newly retrieved dataset of restaurants is classified based on the study in the previous module. The study of user's tastes is not a one-time process and has to be done iteratively. The whole process is taken further by collecting more datasets and adding them to our training data. Recalling that the application is always functioning, when it detects a change in the user's geo-located position, it gets his coordinates and finds the restaurants around those coordinates. We only collect those feature vectors or attribute profile [12, 13] of the object, which we think would describe the object as a whole.

Each restaurant from the retrieved list is passed onto calculate its probabilistic value. Therefore, the probability that a user likes a restaurant R is

$$P(\{like\}|R) = P(\{like\}) \prod_{i=1}^{n} P(X_i|\{like\})$$

$$P(\{like\}) = \frac{\text{Number of Restaurants liked}}{\text{Total number of restaurants}}$$

and P (X| {*like*}) is the user profile that is already stored in our database. Now since we have a training data which consists of the user's favorite restaurants only,

P ({*like*}) = 1 is always true. This turns out the final equation as

$$P(\{like\}|R) = \prod_{i=1}^{n} P(X_i|\{like\})$$

This is to say that the recommendation value of a restaurant is nothing but the summation of the weightage of its features. Here we apply some modification to the recommendation algorithm to produce a more efficient restaurant recommendation. These changes would be more prevalent in the final recommendation list.

Before proceeding with the calculations, the system would check if any restaurant in the list shares the same name and category, but a different location, with that in the training data. This is done to recognize a restaurant chain (e.g. KFC, Mc Donalds) that the user may like. In this case, this restaurant is exempted from calculations and is directly prioritized to the top of the list under the header: "Branches of places you've been to". Otherwise, if the restaurant shares all the features with anyone in the training data, even then it makes to the top of the list with the header: "You've been here". These headers precede "Other Exciting Places" header, which is the calculated list of the remaining restaurants. This last list is sorted according to the recommendation value.

With the coming of high resolution screens and sizes like 4 inches in smart phones, we produce a visualized result of the recommendations on a map like interface [14] with the user's position along with all the venues plotted.

## VI. Implementation

We have developed conceptual design for a mobile web app, "My Eat!" that would use some very advanced features in designing and development of mobile and web apps. Though this app is primarily made for mobiles, it uses Responsive Layout Design that would enable the app to run on any mobile environment like a tablet or a laptop. This application only needs to be started once, and then it runs as a background process.

As seen in Figure 2, the system would continuously keep on checking the user's location depending on certain events. As seen from the flow chart, there is minimum user intervention after logging in. The user only needs to interact with the application if he needs further details of a restaurant, otherwise, he can just glance at the report.

The user sees his own location in the screen after logging in. Figure 3(a) clearly depicts that we have used Google's geolocation api and map to track the user. The steps shown in Figure 2, from locating the user to displaying the report, are being carried on at this time. When the result is ready, the top left button would pull down a slider that shows the report under three drop down headers in Figure 3(b). Now the user has to scroll down to see the full data. Selecting any restaurant would display a static information page about that restaurant including its description, snapshot and rating taken from Foursquare page of that restaurant. Figure 3(d) shows that now the map has also been populated by the locations of

the restaurants. The markers use the serial number of the restaurants from the list to display them on map. Restaurants not in the map have their markers hanging on the edge of the map.

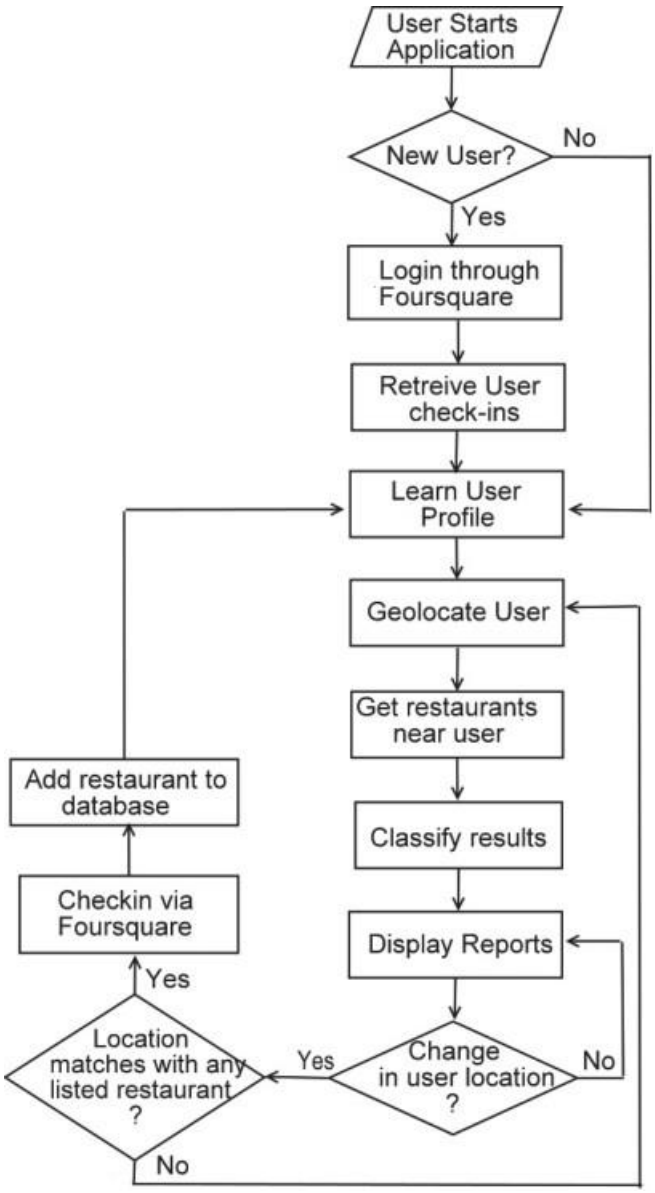


Figure 2. Work Flow Chart

My Eat! is not a native app for any mobile platform, so the app is hosted on a cloud server and all the user data is also stored in a cloud database. The app uses the user Foursquare credentials to store his data on the cloud.

The only overhead time that the app requires is the time to calculate the recommendation list of the restaurants. The remaining overheads such as, data retrieval and map navigation directly depends upon the cellular data bandwidth and may vary from place to place. The app has a light architecture and thus it will not be a problem for low end phones to run this app too.

## VII. Future work

We are trying to add extra features to the mobile app like voice-based notifications and results, number of people already present at the venue, online reservation of restaurants. We also want to integrate 'Facebook places' to get user's past visits data. More current information of recommended restaurants would also be searched online to lookout for offers and discounts. Locating the user's friends, if nearby, and providing recommendations for same venues. Further, we are also working to show a more analyzed result through graphs.

## VIII. Conclusion

This study proposes a model that combines localization, personalization and content-based recommendation in a dynamic and ubiquitous environment. A different and un-social form of personalization that is only derived from the user's behavior and caters to his needs is designed. The common problems have also been addressed and to dealt with. The architecture is kept minimal and light and common algorithms are fused in a unique way. Recommender systems continue to evolve on a daily basis, and we believe we have contributed to this evolution.

## References

[1] Ding Xizojun, Iijima Junichi and Ho Sho. Unique Features of Mobile Commerce. In 4th Asian eBusiness Workshop, Journal of Electronic Science and Technology of China, University of Electronic Science and Technology of China, Vol 2, No 3, pp 205 - 210, Aug. 2004.)

[2] Christos K. Georgiadis and Athanasios Manitsaris. Personalized Recommendations In Mobile Commerce. In 7th Hellenic European Conference on Computer Mathematics & its Applications, HERCMA 2005, Athens, Greece, September 2005.

[3] Daniele Querciaxy, Neal Lathiaz, Francesco Calabresey, Giusy Di Lorenzoy and Jon Crowcroftx. Recommending Social Events from Mobile Phone Location Data. In Proceedings of ICDM. 2010, 971-976.

[4] Raymond J. Mooney and Loriene Roy. Content-Based Book Recommending Using Learning for Text Categorization. In Proceedings of the SIGIR-99 Workshop on Recommender Systems: Algorithms and Evaluation, Berkeley, CA, August 1999.

[5] Janne Lindqvist, Justin Cranshaw, JasonWiese, Jason Hong, and John Zimmerman. I'm the Mayor of My House: Examining Why People Use foursquare - a Social-Driven Location Sharing Application. In CHI: Location Sharing, Vancouver, BC, Canada, 2011.

[6] Mustansar Ali Ghazanfar and Adam Prugel-Bennett. An Improved Switching Hybrid Recommender System Using Naive Bayes Classifier and Collaborative Filtering. In IAENG International Conference on Data Mining and Applications, 17-19 March, 2010.

[7] Wan-Shiou Yang, Hung-Chi Cheng and Jia-Ben Dia. A location-aware recommender system for mobile shopping environments. In Expert Systems with Applications 34, 2008.

[8] Mu-Hsing Kuo, Liang-Chu Chen and Chien-Wen Liang. Building and evaluating a location-based service recommendation system with a

preference adjustment mechanism. In Expert Systems with Applications 36, 2009.

[9] Fan Yang and Zhi-Mei Wang. A Mobile Location-based Information Recommendation System Based on GPS and WEB2.0 Services. In WSEAS Transactions on Computers, 2009, Pages 725-734.

[10] Michael J. Pazzani and Daniel Billsus. Content-Based Recommendation Systems. In The Adaptive Web 2007: 325-341.

[11] Chumki Basu, Haym Hirsh and William Cohen. Recommendation as Classification: Using Social and Content-Based Information in Recommendation. In AAAI-98 Proceedings, 1998.

[12] Souvik Debnath, Niloy Ganguly and Pabitra Mitra. Feature Weighting in Content Based Recommendation System Using Social Network Analysis. In WWW '08 Proceedings of the 17th international conference on World Wide Web, Pages 1041-1042.

[13] Moon-Hee Park, Jin-Hyuk Hong, and Sung-Bae Cho. Location-Based Recommendation System Using Bayesian User's Preference Model in Mobile Devices.In Ubiquitous Intelligence and Computing (2007), pp. 1130-1139.

[14] L. Martínez, R.M. Rodríguez and M. Espinilla. REJA: A Georeferenced Hybrid Recommender System For Restaurants.In Web Intelligence and Intelligent Agent Technologies, 2009.

[15] Andrew McCallum and Kamal Nigam. A Comparison of Event Models for Naive Bayes Text Classification. In AAAI/ICML-98 Workshop on Learning for Text Categorization, pp. 41-48.

[16] Jason D. N. Rennie, Lawrence Shih, Jaime Teevan and David R. Karger. Tackling the Poor Assumptions of Naive Bayes Text Classifiers. In Proceedings of the Twentieth International Conference on Machine Learning, 2003.


Figure 3(a)

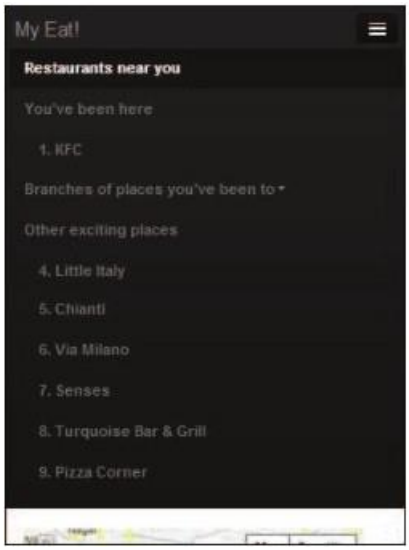


Figure 3(b)

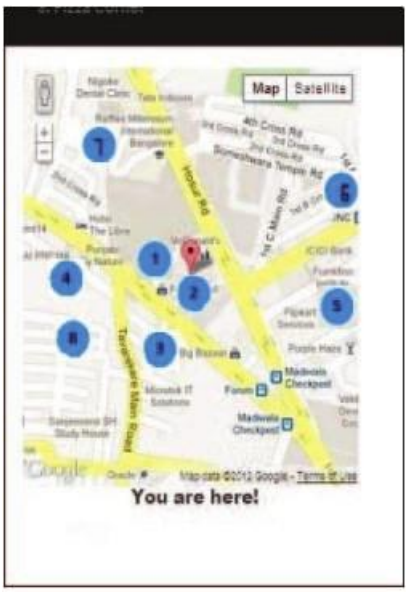


Figure 3(c)


Figure 3(d)