



# **A Proactive Multi-Type Context-Aware Recommender System in the Environment of Internet of Things**

**Yassmeen A. Salman**

**Supervisor: Dr. Abdellatif Abu-Issa**

**A thesis submitted to the faculty of Engineering and Technology  
in fulfilment of the requirements for the  
degree of Master of Science (M.Sc.) in Computing.**

**Faculty of Engineering and Technology**

**Birzeit University**

**February 2016**

# ABSTRACT

In this age of information overload, people use a variety of strategies to make choices about what to buy, how to spend their leisure time, and even with whom to connect. Recommender systems automate some of these strategies in order to predict user's interest in information, products, and services among the tremendous amount of available items.

Recommender systems have been developed in parallel with the web. Currently, these systems are incorporating context and social information of the user, producing context aware recommender systems. In the future, they will use implicit, local and personal information of the user from the Internet of things. The Internet of things paradigm encapsulates the concept of connectivity with anyone and anything at anytime and anywhere. That's why it will offer a lot of information about the user, thus knowing more about his context, and consequently enables high quality and satisfactory recommendations.

Most recommender systems follow the request-response approach. This means that the recommendations are provided to the user upon his request. Recently, a proactive recommender system - that pushes recommendations to the user when the current situation seems appropriate, without explicit user request - has been introduced in the research area of recommender systems.

In this thesis, a design of a context aware recommender system that recommends different types of items proactively in the Internet of things environment is proposed. A major part of this design is the context aware management system which decides if the context is proper to push a recommendation or not, and what type of recommendations to push. As a proof of concept of the proposed system, we have designed a three-type proactive recommender system. The three types are: gas stations, restaurants and attractions. We assume that the context aware management system inputs are derived virtually from the Internet of things, and its output is a score that determines if the context is appropriate for a recommendation and identifies the recommendation type. In this context aware management system, we have used a neural network to do the reasoning of the context. The results of 5000 random contexts were tested. For an average of 98% of them, our trained neural network generated correct recommendation types in the correct times and contexts.

Moreover, A prototype for the system as an application has been developed and a user's acceptance survey has been conducted to 50 users. The results of the survey were very satisfactory and showed high users' interest in such application.

## ملخص

في هذا العصر من التوافر الزائد للمعلومات، يستخدم الناس استراتيجيات متنوعة للتمكن من اتخاذ القرارات حول ماذا يشتررون، وكيف يقضون أوقات فراغهم، وحتى مع من يتصلون. أنظمة التوصية (Recommender Systems) أتمت بعض من هذه الاستراتيجيات من أجل التنبؤ بالمعلومات والمنتجات والخدمات التي تهم المستخدم من بين الكم هائل من المنتجات والمعلومات المتاحة.

طوّرت أنظمة التوصية بالتوازي مع شبكة الإنترنت. حالياً، هذه الأنظمة تدمج السياق والمعلومات الاجتماعية للمستخدم، مما أنتج أنظمة التوصية المدركة للسياق. في المستقبل، سوف تستخدم المعلومات الضمنية والمحلية والشخصية للمستخدم بعد الحصول عليها من إنترنت الأشياء. إنترنت الأشياء هو مستقبل الإنترنت الذي يتضمن مفهوم الاتصال مع أي شخص وأي شيء في أي وقت وفي أي مكان. ولهذا بيئة إنترنت الأشياء سوف تقدم الكثير من المعلومات عن المستخدم، وبالتالي ستزيد من معرفة السياق له، وبالتالي ستستطيع أنظمة التوصية تقديم توصيات أكثر دقة ومرضية أكثر للمستخدم.

معظم أنظمة التوصية تتبع نهج الاستجابة للطلب. وهذا يعني أن يتم تقديم توصيات للمستخدم بناء على طلبه. في الآونة الأخيرة، ظهر مفهوم نظام التوصية المبادر والذي يزود المستخدم بالتوصيات عندما يبدو الوضع الراهن مناسباً، دون طلب صريح من المستخدم.

في هذه الرسالة، قمنا بتصميم نظام توصية تلقائي متعدد الأنواع مدرك للسياق في بيئة إنترنت الأشياء. الجزء الأبرز في هذا التصميم هو نظام إدارة السياق الذي يقرر إذا كان السياق مناسباً لتقديم توصية أو لا، وأي نوع من التوصيات عليه أن يقدم. لاثبات مفهوم النظام المقترح، افترضنا ثلاثة أنواع من أنظمة التوصية المبادرة وهي: محطات الوقود والمطاعم والأماكن السياحية. وافترضنا أن مدخلات نظام إدارة السياق مستمدة افتراضياً من إنترنت الأشياء، ومخرج النظام هو قيمة لكل من الأنواع الثلاثة تحدد إذا كان السياق المناسب لتقديم لتوصية ويحدد نوع هذه التوصية من بين الأنواع الثلاثة المذكورة. في نظام إدارة السياق، استخدمنا الشبكة العصبية (Artificial Neural Network) لتحليل منطق السياق. تم اختبار نتائج 5000 سياقات عشوائية. ل 98٪ منها، أنتجت شبكتنا العصبية المدربة أنواع توصية صحيحة في الأوقات والسياقات الصحيحة.

أخيراً، تم تطوير نموذج أولي (Prototype) للنظام كتطبيق وتم إجراء مسح لقبول المستخدم لهذا النظام. تكونت العينة من 50 مستخدماً، وكانت نتائج المسح مرضية للغاية وأظهرت اهتمام كبير من قبل المستخدمين بهذا التطبيق.

# ACKNOWLEDGEMENT

*I would like to express my most sincere thanks to my supervisor, Dr. Abdellatif Abuissa, who gave me the opportunity to work on this interesting project. I am grateful for his advice, help, and support.*

*Special thanks go to my family, especially my husband and my parents for their love, support and encouragement.*

# TABLE OF CONTENTS

<b>INTRODUCTION .....</b>	<b>1</b>
1.1. Background and Motivation .....	1
1.2. Proposed System Vision .....	3
1.3. Pieces of The Puzzle .....	4
1.4. Summary of Contributions .....	4
1.5. Publications and Awards .....	5
1.6. Structure of this Report.....	5
<b>BACKGROUND and LITERATURE REVIEW.....</b>	<b>7</b>
2.1. Recommender Systems.....	7
2.1.1. Content- Based Approach.....	9
2.1.2. Collaborative-Based Approach.....	9
2.1.3. Hybrid Methods .....	10
2.1.4. Limitation of Content-Based and Collaborative Filtering Approaches .....	11
2.2. Context-Aware Systems .....	12
2.2.1. Context-Awareness Related Definitions.....	13
2.2.2. Context in Recommender Systems .....	14
2.2.3. Obtaining Contextual Information.....	15
2.2.4. Context Management Systems.....	15
2.3. Context Aware Recommender Systems (CARS) .....	21
2.3.1. Prior Work in Context Aware Recommender Systems .....	21
2.4. Internet of Things (IoT).....	23
2.4.1. Evolution of Internet.....	23
2.4.2. Internet of Things Appearance .....	24
2.4.3. Internet of Things Future Vision .....	26
2.5. Proactivity in Recommender Systems .....	27
2.6. Proactive Multi-Type Context-Aware Recommender System .....	30
2.7. Artificial Neural Networks (ANN) .....	31
2.7.1. Background.....	31
2.7.2. The Basic Artificial Model .....	32
2.7.3. ANN Classification Based on Structure .....	33

2.7.4.	ANN Learning Techniques .....	34
2.7.5.	Back-Propagation Algorithm .....	35
2.7.6.	Conclusion .....	36
2.8.	Chapter 2 Sections Interconnection .....	36
<b>SYSTEM DESIGN and IMPLEMENTATION .....</b>		<b>37</b>
3.1.	General System Design.....	37
3.2.	Context Aware Management System Design (CAMS) .....	38
3.2.1.	IoT Sensors Data.....	39
3.2.2.	Context Acquisition .....	40
3.2.3.	Context Modelling .....	40
3.2.4.	Context Reasoning.....	41
3.2.5.	Scoring Algorithm .....	48
3.3.	Recommender Type Triggering GUI.....	49
3.4.	Summary.....	52
<b>PROTOTYPING and USERS' EVALUATION.....</b>		<b>54</b>
4.1	System Prototype -As an Application.....	54
4.2.	User's Acceptance Evaluation .....	62
4.3	Survey Key Results & Findings .....	65
<b>CONCLUSIONS and FUTURE WORK .....</b>		<b>67</b>
5.1.	Conclusions.....	67
5.2.	Future Work.....	68
<b>REFERENCES .....</b>		<b>69</b>
<b>APPENDICES.....</b>		<b>76</b>
Appendix A: Matlab Codes .....		76
Appendix B: Survey Form.....		82
Appendix C: Published Paper .....		85

# LIST OF FIGURES

<b>Fig. 1.1:</b> Pieces of the Puzzle.....	4
<b>Fig. 2.1:</b> Context Life Cycle.....	16
<b>Fig. 2.2:</b> The five phases of the Internet evolution.....	23
<b>Fig. 2.3:</b> IoT: anytime, anyplace, and anything.....	25
<b>Fig. 2.4:</b> IoT Expansion Through 2019 .....	26
<b>Fig 2.5:</b> Proposed Two-phase Proactivity Model .....	29
<b>Fig. 2.6:</b> Thesis Idea Info-graphic .....	30
<b>Fig. 2.7:</b> Schematic Diagram of ANN .....	32
<b>Fig. 2.8:</b> Neuron Inputs/Outputs .....	33
<b>Fig. 2.9:</b> Chapter Two Sections Interconnection.....	36
<b>Fig. 3.1:</b> Proposed System Design.....	37
<b>Fig. 3.2:</b> Context Aware Management System (CAMS) .....	39
<b>Fig. 3.3:</b> IoT Virtual Data.....	40
<b>Fig. 3.4:</b> Neural Network Design.....	42
<b>Fig. 3.5:</b> ANN Input Vectors.....	42
<b>Fig. 3.6:</b> ANN Output Vectors.....	43
<b>Fig. 3.7:</b> ANN Training Window.....	44
<b>Fig. 3.8:</b> ANN Performance .....	45
<b>Fig. 3.9:</b> Matlab Code Output (ANN Contexts-Types Scores) .....	46
<b>Fig. 3.10:</b> ANN Test Results .....	47
<b>Fig. 3.3.a:</b> Recommender Type Triggering GUI- Gas stations type is triggered .....	49
<b>Fig. 3.3.b:</b> Recommender Type Triggering GUI- Restaurants type is triggered .....	50
<b>Fig. 3.3.c:</b> Recommender Type Triggering GUI- Attractions is triggered .....	51
<b>Fig. 3.3.d:</b> Recommender Type Triggering GUI- No type is triggered .....	52
<b>Fig. 4.1 a-o:</b> ProRec Screenshots .....	55-61
<b>Fig. 4.2 a-e:</b> User's Survey Results .....	62-65

# LIST OF TABLES

**Table 2.1:** User-Item Matrix..... 8

**Table 3.1:** ANN Results Count..... .. 47

**Table 3.2:** ANN Results Analysis.....47



# LIST OF ABBREVIATIONS

**ANN:** Artificial Neural Network.

**CAMS:** Context Aware Management System.

**CARS:** Context Aware Recommender Systems

**GPS :** Global Positioning System.

**GUI :** Graphical User Interface.

**HTML:** Hyper Text Markup Language.

**IBSG :** Internet Business Solutions Group.

**IoE:** Internet of Everything.

**IoT:** Internet of Things.

**ITU:** International Telecommunication Union.

**JITIR:** Just In Time Information Retrieval.

**KSOM:** Kohonen Self-Organizing Map.

**LBS:** Location Based Services.

**MD:** Multidimensional Data.

**OLAP:** On-Line Analytical Processing.

**OWL:** Ontology Web Language.

**RDF:** Resource Description Framework.

**UML:** Unified Modelling Language.

# CHAPTER ONE

## INTRODUCTION

The vast growth of information on the Internet as well as the increasing number of visitors to websites created a great need to new technologies that can assist us to find resources of interest among the overwhelming items available. One of the most successful such technologies is the Recommender System. In this chapter, an introduction to the recommender system that will be proposed in this thesis will be provided. Starting with a background and motivation of our system, then exploring the system future vision, then highlighting the contributions of this thesis, and finally over-viewing the structure of this report.

### 1.1. Background and Motivation

A Recommender System is an intelligent system that makes suggestion about items to users that might interest them. Recommender systems apply data mining techniques and prediction algorithms to predict users' interest in information, products and services among the tremendous amount of available items. A recommender system tries to make its recommendation depending on the user's background and personalization in order to provide affordable, personal, and high-quality recommendations [1]. A common example is the RS used by Amazon website which used to recommend books for users (when it was an online bookstore before diversifying to sell almost everything) depending on what they bought in the past, and what other similar users have bought.

Recommender systems have been developed in parallel with the web. Initially, recommender systems used information from content-based, demographic-based or collaborative filtering techniques or a hybrid of them. Currently, with the development of web 2.0, these systems integrate social information such as followed and followers,

friend's lists, posts, blogs, and tags [2]. They also started to integrate contexts in the recommendation process. Where context is any real time known information about a user that helps to give better recommendations, such as time, location, temperature ... etc. This resulted in a new approach for recommendations named as "context-aware recommendation". For example, if the recommender system knows the location of the user at a specific time "from his Smartphone GPS", it will not recommend to him a faraway restaurant, and that will enhance the quality of recommendations.

In the future, recommender systems may use local and personal information from various sensors and devices on the Internet of things (IoT) [2]. In the IoT, not only people are connected to the Internet through their PCs, laptops, and smartphone's, but also objects like cars, houses, roads, medical instruments, and many other objects will be connected to the internet. This has already been applied on some countries, partially. But the aim of the IoT is to connect anything, at any time, from any place, and for anyone [3]. The predictions talks about more than 50 billion things connected to the IoT by 2020 [4].

Most of the existing recommender systems deliver their recommendations upon the user's request. But recommendations can be delivered (pushed) to the user without his explicit request. That, when the context of the user is known in the real time, the system can know his situation and decide whether this situation needs a recommendation or not, then recommending a suitable item if that situation is favourable. Later, a new term that encapsulates the previous concept has appeared "Proactivity". It means that the system pushes recommendations to the user when the current situation seems suitable. The challenge in proactive systems is not only to know which item to recommend, but also when to recommend this item. Besides taking into account to be not disturbing to the user.

Also most of the existing recommender systems are specialized recommenders. They can recommend only one type of items. i.e. recommenders for books, movies, restaurants, hotels, attractions ...etc. But with the fact that users are becoming online all the time (through different devices mainly their smartphones), and with the big amount of information that is known about the user's context, which will become larger and larger

when deploying the IoT, the need for a multi-type recommender system will appear. This multi-type system will push different types of recommended items to the user according to his situation. The same system may recommend a restaurant or a hotel or a gas station ...etc.

In this thesis, a design of a proactive multi-type context aware recommender system will be proposed. This system will be adaptive enough to gain user's satisfaction and it will include a context management system that will best fit the IoT environment.

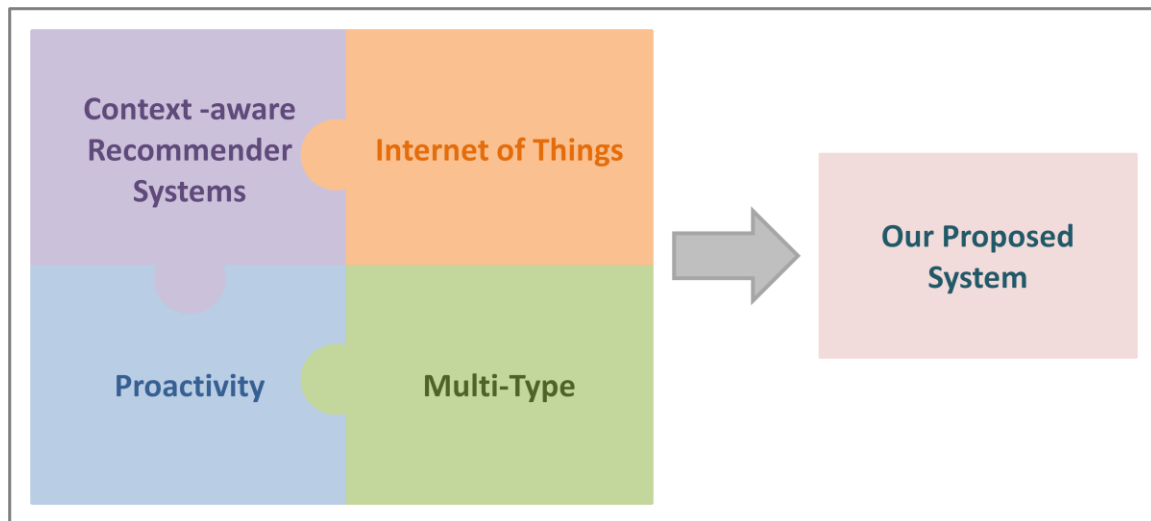
## **1.2. Proposed System Vision**

The proposed system will be smart enough to know the context and status of the user in real time and to recommend to him (in a proper time) what he needs and likes without even asking him. Below are some scenarios of how this system will work:

- A specific user is driving his car, and a small amount of fuel is remaining in the fuel tank of the car, he must be given recommendations about a close and highly rated fuel stations. The system depends on the status of the car (fuel level is low), its location, and the ratings of the fuel stations.
- The same system may recommend to the same user a restaurant depending on his location gotten from his smartphone's GPS, his history of favourite food, his explicit ratings, and may be depending on the access time. If it's morning the type of recommended restaurants may differ from afternoon. For example, in the morning time, the system will recommend snack meals restaurants and afternoon will recommend lunch meals restaurants.
- In the IoT, a person with a disease like high pressure or diabetes will have embedded sensors that measure his pressure frequently, and when sensing high pressure it will send an alarm to his mobile. In this case our proposed system must recommend to him highly rated nearby pharmacies and medical centers depending on his location that was gotten from his mobile GPS.

### 1.3. Pieces of The Puzzle

This research has incorporated the context-aware recommender systems field with the proactivity concept which has recently appeared, assuming the IoT environment. What mainly distinguishes our system from the existing recommender systems is the multi-type approach; in which more than one type of recommendations are recommended to the user rather than one type. For example, existing recommender systems may recommend hotels or restaurants or books...etc. In our proposed system many types may be recommended to the user, so that the same system will recommend restaurants and attractions and shops ...etc. Figure 1.1 shows how the pieces of the puzzle are integrated to produce the proposed system.



**Fig 1.1:** Pieces of the Puzzle

### 1.4. Summary of Contributions

In this work, recommender systems have been studied carefully. Starting from the general recommenders approaches, then moving toward the context-ware recommender systems, then concentrating on proactive recommender systems. Different from traditional recommender systems, proactive recommender systems pushes recommendations to the user when time and context seems appropriate, without explicit user's request. Considering the fact that the future is for the IoT, we propose a new proactive recommender system design that utilizes the IoT environment. Moreover, this design

attains multi-type recommender system rather than one type recommender system as all existing previous recommender systems.

**The contributions of this thesis can be summarized in the following points:**

- A comprehensive study about recommender systems and their future.
- A multi-type recommender system model idea is proposed for the first time, designed, implemented, and tested.
- The proposed system model was designed to work in the future IoT environment.
- The Artificial Neural Network (ANN) has been studied and used to make the decision when to push a recommendation and what type of recommendation to push. Using the ANN for deciding whether to push a recommendation or not and what type to push has been tested and it has achieved a high accuracy.
- The proposed recommender system model has been translated into application prototype that is supposed to work in the IoT environment, this prototype was subjectively evaluated by a group of users who showed an interest and provided a satisfactory evaluation of the system.

## **1.5. Publications and Awards**

- A paper based on this thesis has been published in the 15<sup>th</sup> IEEE International Conference on Computer and Information Technology (CIT 2015) in Liverpool-UK. This paper is attached in Appendix C.
- The system idea and the application prototype that are introduced in this thesis has won the Expotech 2015 digital contest competition for the IoT category.

## **1.6. Structure of this Report**

In the previous sections, we have introduced this thesis research area and its objectives and motivations. The remaining chapters of this thesis are organized as follows:

In chapter two, we provide a background and literature review.

In chapter three, we present the general design of the proposed system, and the design of the main block in the system which is the context aware management system (CAMS). It

also presents the methods of implementation and testing, and the results that have been achieved.

In chapter four, we provide a prototype of the system, if it will be developed as an application. We also provide the results of a survey that was conducted to users to evaluate their acceptance of the system idea and prototype.

In chapter five, we briefly conclude the work results and highlight some of the future works on this thesis.

# CHAPTER TWO

## BACKGROUND and LITERATURE REVIEW

In this chapter, a background and literature review for this thesis is explained. It covers recommender systems and their types and limitations in section 2.1, context aware systems and their design concerns in section 2.2, context aware recommender systems and their prior work in section 2.3, proactive recommender systems literature in section 2.4, Internet of things appearance and future in section 2.5. And in section 2.6 a theory that combines all the previous topics that will be the major concern of this thesis is introduced. In the last section, section 2.7, a brief theory about artificial neural networks which were used for the implementation in this thesis is introduced.

### 2.1. Recommender Systems

Recommender Systems have become an important research area since the mid-1990s. The roots of the work on recommender systems can be traced back to older work in cognitive science, approximation theory, information retrieval, forecasting theories, decision systems, and have relations to management science. But, after the 1990's, the focus on recommendation problems that explicitly rely on the ratings structure has appeared [1].

The basic idea behind recommendation is the problem of estimating ratings for the items that have not been seen by a user. Spontaneously, this estimation is usually based on the ratings given by this user to other items and on some other information. Once ratings could be estimated for the yet unrated items, then users can be recommended with items of the highest estimated ratings [1]. In order to execute this task, recommender systems



need an input usually demonstrated as a user-item rating matrix which shows users' preferences. Table 2.1 shows a sample of this matrix.

	Item 1	Item2	Item3
User1	3	5	4
User2	-	3	2
User3	4	-	3

**Table 2.1:** User-Item Matrix

In the user-item matrix, the cell corresponding to user  $x$  and item  $y$  must contain the rating of user  $x$  to item  $y$ . But, if the user has not rated the item, then the recommender system must estimate the rating of the not-yet-rated item from other users' ratings to this item or other items. For example, in the above table the – s are the ratings that must be estimated.

More formally, the recommendation problem can be formulated as follows [1]: Let  $C$  be the set of all users and let  $S$  be the set of all possible items that can be recommended, such as books, movies, or restaurants. The space  $S$  of possible items can be very large, ranging in hundreds of thousands or even millions of items in some applications, such as recommending books or CDs.

Similarly, the user space can also be very large. Let  $u$  be a utility function that measures usefulness of item  $s$  to user  $c$ , i.e.,  $u: C \times S \rightarrow R$ , where  $R$  is a totally ordered set (e.g., non-negative integers or real numbers within a certain range). Then for each user  $c \in C$ , we want to choose such item  $s' \in S$  that maximizes the user's utility. More formally:

$$\forall c \in C, s'_c = \arg_{s \in S} \max u(c, s) \dots\dots\dots (1)$$

The new ratings of the not-yet-rated items can be estimated in different ways using methods from machine learning, approximation theory, diverse heuristics, and multi-variable optimization [1]. Once this estimation occurred, the recommender system can recommend the Top- $N$  items that received the highest estimated ratings. Generally, recommender systems are classified according to their approach of rating estimation [1].

And, usually; they are classified to content-based, collaborative filtering, and hybrid approach [5]. These recommendation approaches are described in the following subsections.

### **2.1.1. Content- Based Approach**

The main idea behind the content based recommendations is that items similar to those that were highly rated by the user in the past will be recommended to that user.

More formally, to recommend item  $s$  to user  $c$ , the similarity between the content of  $s$  and other items previously rated by  $c$  is measured. In other words, the utility  $u(c, s)$  of item  $s$  for user  $c$  is estimated based on the utilities  $(c, s_i)$  assigned by user  $c$  to items  $s_i \in S$  that are “similar” to item  $s$  [1].

For example, in books recommender system, in order to recommend books to user  $c$ , the content based recommender system tries to understand the commonalities among the books that user  $c$  has rated highly in the past (specific author, subject matter, language of writing, ...etc.). Then books that have a high degree of similarity to user’s preferences are to be recommended.

### **2.1.2. Collaborative-Based Approach**

In this approach, the recommender system tries to predict the utility of items for a particular user based on the items previously rated by other users having the same taste with the target user. That is, the utility  $u(c, s)$  of item  $s$  for user  $c$  is estimated based on the utilities  $u(c_j, s)$  assigned to item  $s$  by those users  $c_j \in C$  who are “similar” to user  $c$  [1].

Returning to the books recommender system, if a user who is very similar to the target user has highly rated a specific book, then the recommender system will recommend the same book to the target user, based on the fact that these two users have common taste or preferences.

The earliest works in this field were introduced by Grundy [6], which is considered to be the earliest one, and Tapestry systems [7]. Other recommenders [8-10] are considered to be the first works in “automation” of prediction in collaborative filtering. Collaborative filtering approaches can be classified into two general categories: memory-based (or

heuristic-based) and model-based techniques [11]. These two categories are briefly described consequently:

- **Memory-Based Methods**

These methods use the user-item rating data to measure similarity between users or items. Most classic examples for these methods are item-based and user-based collaborative filtering. In the user-based approach the value of an unknown rating  $r_{c,s}$  by user  $c$  for item  $s$  can be calculated as an aggregate of the ratings by other users (most similar users to user  $c$ ) for item  $s$  [1]:

$$r_{c,s} = \underset{c' \in \hat{C}}{aggr} r_{c',s} \dots\dots\dots (2)$$

Where  $\hat{C}$  is the set of  $N$  users that are most similar to user  $c$  and who have rated item  $s$ . The simplest aggregation function that can be used is a simple average, whereas the most common aggregation approach used is the weighted sum. But this approach has a significant limitation that it does not consider that different users may use various rating scales. To solve this limitation, deviations are considered from the average rating of users by using adjusted weighted sum [9].

- **Model-Based Methods**

In the memory-based methods, the entire user-item rating matrix is considered in order to predict ratings of not-yet-rated items. But in model-based methods, the collection of ratings are used to learn a model using statistical and machine learning techniques, by which the unknown ratings are predicted [7].

In [11], a comparison between model-based approaches and standard memory-based approaches was performed. It was claimed that, model-based methods outperform memory-based approaches in terms of accuracy of recommendations.

### 2.1.3. Hybrid Methods

Hybrid methods combine collaborative and content-based methods to overcome certain limitations of each of these systems [12, 13]. These limitations are described in section

2.1.4. There are different ways to combine collaborative and content-based approaches into a hybrid recommender system, they are [1]:

1. Combining the prediction results of each of these methods that have been implemented separately.
2. Incorporating collaborative characteristics into a content-based approach or vice versa.
3. Using a general unifying model that incorporates both content-based and collaborative characteristics.

#### **2.1.4. Limitation of Content-Based and Collaborative Filtering Approaches**

For content-based approach, one major limitation is that they need to deal with items which contents are machine readable, such as text-based items. This is to be able to parse them and extract their features. And this limitation can make content-based approach for audio or video impractical, because their features cannot be automatically extracted and must be assigned manually, which is impossible due to the huge number of items [1].

Another limitation associated with these methods is over-specialization. It means that users are limited to recommendations related or similar to the items they liked in the past and they do not get a chance to receive recommendations of items in other tastes that might be interesting for them. For example, an engineer whose history is with engineering books ratings will always receive engineering books recommendations, although he may have other faraway interests such as fishing [1].

Also, the new user problem can be considered as another limitation to this approach. Because, to make a system able to understand the user's taste in order to recommend relative items to him, a user has to rate an enough number of items in anticipation [1].

Collaborative filtering methods do not have some of the described limitations associated with content-based methods. However, they still have the new user problem as in content-based methods. Because they need an enough number of ratings by the user to understand user taste and look for most similar users to that user. In collaborative

filtering methods, there is also a problem called the new item problem. Which occurs when a new item is added to the system, which frequently occurs. The system needs an enough number of ratings for an item by users to be able to recommend it. Both of these problems can be mitigated using hybrid methods which were described in the previous section [1], where the system that combines content-based filtering and collaborative filtering could take advantage from both the representation of the content from the content based approach as well as the similarities among users from the collaborative filtering approach. For example a new user to the hybrid system with no previous content can be recommended items that were highly rated by similar users (demographics similarities like age or region) until knowing his preferences and building his profile.

Another problem associated with collaborative filtering methods is sparsity. In today's systems, there are millions of users and items which can be illustrated with a huge matrix, but the number of available ratings in the matrix compared to the number of unknown ratings is very small. This fact makes the accurate and effective ratings prediction very difficult. Items with only a few ratings do not get a chance to be recommended even if those few ratings for these items are very high. Also, users with unusual tastes cannot be matched with other users according to preferences [12].

An extension of traditional collaborative filtering techniques to overcome the sparsity problem is demographic filtering [14]. In this method the similarity measurement between users is based not only on their common ratings, but also their profiles. For example, features such as location, occupation, education, age, and gender can be extracted from user profiles and, in addition to similar ratings for co-rated items, belonging to the same demographic segment is also considered for similarity measurements.

## **2.2. Context-Aware Systems**

Due to the development in the ubiquitous or pervasive computing technologies nowadays, many systems offer services to users anytime and anywhere. In ubiquitous computing, a user interacts with the computer which can be a laptop, a smartphone, a tablet, or any terminals in everyday objects such as a fridge or a pair of glasses [15].

To offer services that are appropriate to the user, applications and systems should be aware of user's context, and its surrounding environment. This is known as context-awareness.

Context includes information about person's location, social relationships, time, preferences, and any other information that can create a clear image about the situation of person/entity in real time.

A system is considered a context-aware system, if it uses context information of related entities, and adapts itself to the changing situations, in order to offer the best service to the user who will be more satisfied if the service provided to him is on the basis of his situation and needs. The context aware system should be able to extract, deduce, and use contextual information then adapt itself to the current situation, and then offer the services [16].

### **2.2.1. Context-Awareness Related Definitions**

The term context has been defined by many researchers. Among them were Abowd and Mynatt [17] who identified the five W's (Who, What, Where, When, Why) as the minimum information that is essential to understand the context. Rodden et al. [18], Hull et al. [19], and Ward et al. [20] used synonyms to refer to context, such as environment and situation. Schilit et al. [21] and Pascoe [22] have also defined the term context. But Dey [23] claimed that all the previous definitions and other - not mentioned - definitions were too specific and cannot be used to identify context in a broader sense. He provided a definition for context as follows: "*Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves* [24]."

Sanchez et al. [25] explained the difference between raw data and context information. He defined the raw (sensor) data as the unprocessed data which is retrieved directly from the data sources like sensors. And defined the context information as the data that is generated by processing the raw sensor data. For example when a location is captured as

latitude and longitude numbers, this is considered as raw data. But when these numbers are converted into a named location, this is the contextual information.

The term context-awareness was first brought up by Schilit and Theimer in 1994 [26]. Again, Dey and Abowed [24] criticized the previous definitions and stated that those definitions were too specific. That is why Dey has provided another definition for context-awareness as: *“A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user’s task. [24]”*.

### **2.2.2. Context in Recommender Systems**

In the previous section, context was defined in general. But in the field of recommender systems, context can be defined in association with data mining, e-commerce personalization, databases, information retrieval, ubiquitous and mobile context-aware systems, marketing, and management [27].

In the data mining field, context is defined as the events that surrounds the customer and can change his preferences or status. Such as graduation, marriage, a new job ...etc. because these milestone events may incredibly change his style of life. Context-awareness in data mining can help to discover patterns related to a specific context [27].

In e-commerce personalization, context is defined as “intent of a purchase by customers”. For example, the customer who intends to buy a movie to watch with his kids will not choose a similar movie to watch with a friend. So, the intent of purchase varies with the same customer when the context varies. That is why building different purchase profiles for the same user for different contexts will incredibly enhance the predictive models in different e-commerce applications.

In ubiquitous and mobile context-aware systems, location is one of the most important contexts that make the system location-aware or context-aware. Location based services (LBS) has recently been developed. They get context information such as location, time, and type of device in order to send more relevant information to users.

In information retrieval systems, retrieval decisions are not based only on queries and document collections, but also on information about search context. And the same concepts apply for marketing, and management.

### **2.2.3. Obtaining Contextual Information**

In general, there are two main ways to obtain contextual information: explicitly and implicitly. In the explicit way, the users can be asked directly about their preferences. Like the websites that required the user to sign up and fill a web form. In that step, all the required contextual information can be gathered. Another way for obtaining contexts explicitly is adding rating or like/dislike, which gets the user's feedback directly.

In the implicit way, information is extracted implicitly without interacting with the user or asking him. Location, time, device type, and network properties are examples of contextual information that can be obtained implicitly. For example, the location can be determined through the user's mobile GPS if he is accessing the system through his mobile.

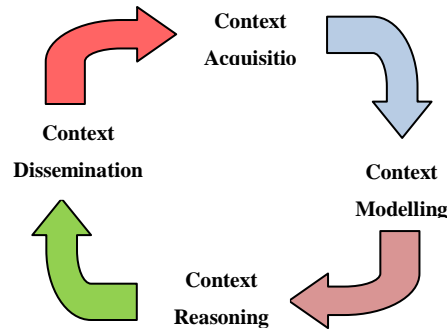
Another way for implicit information gathering is inferring the information using data mining techniques. For example, by mining the movies watched by a family, and using well trained classifier techniques, the number of family members, their ages and genders can be concluded [27].

### **2.2.4. Context Management Systems**

As concluded from above, context- aware recommender systems are adaptive and smart enough to act on behalf of users autonomously. The most important part of context aware systems is the context management phase in which a context modeling, representation, and reasoning is taking place. For each context there is a life cycle which starts when the raw data is acquired and ends when the context is disseminated.

In [28], the authors reviewed many data life cycles that shows how data moves from phase to phase in software systems (e.g. application, middleware). Then they derived proper context life cycle with minimum number of phases but includes all essential functions. This life cycle is shown in figure 2.1:





**Fig 2.1:** Context Life Cycle [28]

The above context life cycle consists of four phases. First, context needs to be acquired from various sources. These sources could be physical sensors or virtual sensors. Second, the collected data needs to be modeled and represented according to a meaningful manner. Third; modeled data needs to be processed to derive high-level context information from low-level raw sensor data, this is referred to as context reasoning. Finally, both high-level and low-level contexts need to be distributed to the consumers who are interested in context [28]. In each of the above phases, there are many factors that must be taken into consideration when designing the context management system that fits the IoT paradigm. The subsections below explore a brief description about the design of each phase.

- **Context Acquisition**

The techniques used to obtain context can be varied based on responsibility, frequency, sensor type, and acquisition process [28].

**Responsibility:** Involves two methods, push and pull [29]. In the pull method the software component which is responsible for acquiring sensor data from sensors sends a request to the sensor hardware periodically or instantly to acquire data. But in the push method, the data is pushed by the physical or virtual sensor to the software component.

**Frequency:** Describes how often context will be generated, instantly or periodically. Instant events do not span across certain amount of time like opening a door or switching on a light. Periodic events are those which span a certain period of time. Raining, animal eating a plant or winter are some examples of interval events. In order to detect this type

of event, sensor data needs to be acquired periodically (e.g. sense and send data to the software every 30 seconds).

**Sensor Types:** There are different types of sensors that can be employed to acquire context. Among the technical community, a sensor refers to as any data source that provides relevant context. Therefore, sensors can be divided into three categories physical, virtual, and logical [30].

**Acquisition Process:** There are three ways to acquire context: sense, derive, or manually provided. In the Sense way, the data is sensed through sensors directly. But in the derive way, the information is generated by performing computational operations on sensor data. Whereas in the manually provided way, users provide context information manually via predefined settings options such as preferences.

- **Context Modeling**

It is also widely referred to as context representation. Context models can be static or dynamic. Static models have a predefined set of context information that will be collected and stored [31]. The requirements that need to be taken into consideration when modeling context information are identified and explained in [32] as heterogeneity, mobility, relationships and dependencies, timeliness, imperfection, reasoning, usability of modeling formalisms, and efficient context provisioning.

The most popular context modeling techniques are surveyed in [33] and [34]. These surveys discuss a number of systems that have been developed based on the most popular context modeling techniques which are: key-value, markup schemes, graphical, object based, logic based, and ontology based modeling.

**Key-Value Modeling:** This is the simplest form of context representation among all the other techniques. It models context information as key-value pairs in different formats such as text files and binary files. They are easy to manage when they have smaller amounts of data. But this modeling is not scalable to store complex data structures.

**Markup Scheme:** This technique is an improvement over the key-value modeling technique. It models data using tags. Therefore, context is stored within tags. The advantage of using markup tags is that it allows efficient data retrieval. One main disadvantage of using this technique is that markup languages do not provide advanced expressive capabilities that allow reasoning.

**Graphical Modeling:** This technique models context with relationships. An example of this modeling technique is Unified Modeling Language (UML) [35]. In terms of expressive richness, graphical modeling is better than markup and key-value modeling because it allows relationships to be incorporated into the context model.

**Object Based Modeling:** Object based (or object oriented) concepts are used to model data using class hierarchies and relationships. Object oriented paradigm supports encapsulation and reusability. Since most of the high-level programming languages support object oriented concepts, modeling can be integrated into context-aware systems easily. Therefore, object based modeling is suitable to be used as an internal, code based, run-time context modeling, manipulation, and storage mechanism. However, it does not provide built-in reasoning capabilities. Also validation of object oriented designs is difficult due to the lack of standards and specifications.

**Logic Based Modeling:** In this technique, facts, expressions, and rules are used to represent information about the context. Rules are mainly used to convey policies, constraints, and preferences. It provides much more expressive richness compared to the other techniques discussed previously. Logic based modeling allows new high-level context information to be extracted using low level context. Therefore, it has the capability to enhance other context modeling techniques by acting as a supplement.

**Ontology Based Modeling:** In this technique, the context is organized into ontology's using semantic technologies. A number of different standards (RDF, RDFS, and OWL) and reasoning capabilities are available to be used depending on the requirement. For this type, many development tools and reasoning engines are on hand. However, context retrieval can be computationally exhaustive and time consuming when the amount of data is increased.

- **Context Reasoning Decision Models**

Context reasoning can be defined as a method of deducing new knowledge, and understanding better, based on the available context [36]. Another definition explained it as a process of giving high-level context deductions from a set of contexts [37]. Context reasoning techniques were classified broadly into six categories, they are: supervised learning, unsupervised learning, rules, fuzzy logic, ontological reasoning and probabilistic reasoning [28]. Below is a brief description for each of them.

**Supervised Learning:** In this category of techniques, first training examples are collected. Then they are labeled according to the expected results. Then a function that can generate the expected results using the training data is derived. Decision tree is a supervised learning technique where it builds a tree from a dataset that can be used to classify data. Bayesian Networks is a technique based on probabilistic reasoning concepts. It is commonly used in combining uncertain information from a large number of sources and deducing higher-level contexts. Artificial neural networks are a technique that attempts to mimic the biological neuron system. They are usually used to model complex relationships between inputs and outputs or to find patterns in data. Body sensor networks domain has employed this technique for pervasive healthcare monitoring in [38].

**Unsupervised Learning:** These techniques can find hidden structures in unlabeled data. Due to the use of no training data, there is no error or reward signal to evaluate a potential solution. Clustering techniques such as K-Nearest Neighbor is commonly used in context-aware reasoning. Unsupervised neural network techniques such as Kohonen Self-Organizing Map (KSOM) are used to classify incoming sensor data in a real-time fashion [39].

**Rules:** This is the simplest and most straightforward method of reasoning out of all of them. Rules are usually structured in an IF-THEN-ELSE format. This is the most popular method of reasoning. It allows the generation of high level context information using low level context. Rules are expected to play a significant role in the IoT [28], where they are the easiest and simplest way to model human thinking and reasoning machines.

**Fuzzy Logic:** This technique allows approximate reasoning instead of fixed reasoning. It is similar to probabilistic reasoning but confidence values represent degrees of membership rather than probability [40]. In fuzzy logic partial truth values are acceptable; whereas, in traditional logic theory, acceptable truth values are 0 or 1. It allows real world scenarios to be represented more naturally. It further allows the use of natural language (e.g. temperature: slightly warm, fairly cold) definitions rather than exact numerical values (e.g. temperature: 10 degrees Celsius). In most cases, fuzzy reasoning cannot be used as a separate reasoning technique. It is usually used to complement other techniques such as rules based or ontological reasoning.

**Ontology Based:** This technique is based on description logic, which is a family of logic based knowledge representations of formalisms. Ontological reasoning is mostly supported by two common representations of semantic web languages, they are: RDF(S) and OWL (2). The advantage of ontological reasoning is that it integrates well with ontology modeling. But its disadvantage is that it is not capable of finding missing values or unclear information where statistical reasoning techniques are good at that [28].

**Probabilistic Logic:** This technique allows decisions to be made based on probabilities attached to the facts related to the problem. It can be used to combine sensor data from two different sources. It also can be used to identify resolutions to conflicts among context. Hidden Markov Models are an example of such a probabilistic technique that allows state to be represented using observable evidence without directly reading the state [28]. Markov Models are commonly used in activity recognition in context-aware domains. For example, it has been used to learn situation models in a smart home [41].

- **Context Dissemination**

It provides methods to distribute context to the consumers. It is the same as context acquisition which was described above. All the factors that were discussed under context acquisition need to be considered for context distribution as well.

## **2.3. Context Aware Recommender Systems (CARS)**

As described in the previous sections, there has been much work done in the field of recommender systems. However, many of the recommendation systems rely only on users, items, and ratings which are seem insufficient for suggesting satisfactory recommendations. This is why incorporating contextual information in recommender systems starts to be necessary. Especially that due to the advances in ubiquitous computing, more contexts about the users can be gathered.

In some applications it is important when, where, and with whom the user is using the application. Therefore, relevant recommendations should be based on this context information of the user. For example, the selection of a movie to watch may depend on many factors , like the day of the week, the place in which they will watch the movie, and the accompany people with whom the movie will be watched. So, if the recommender system does not know the previous context information, then it will provide poor and non-relevant recommendations to the user. For Example, this recommender system may recommend a horror movie to a user who will watch a movie with his children. Absolutely this is a poor recommendation.

### **2.3.1. Prior Work in Context Aware Recommender Systems**

The Multidimensional Data (MD) model which was introduced by Adomavicius in [1] is the most referenced work in the area of context-aware recommendation systems. In this model, contextual information is defined using OLAP (On-Line Analytical Processing) which are used in the databases community under the data warehousing applications [2].

The main concept behind the MD model is to extend the classical two dimensional user-item approach. This is by adding multiple dimensions to present a multidimensional recommendation model, where the extra dimensions are the contexts. In the classical approach, once an initial set of ratings is provided to the recommendation process explicitly or implicitly, the recommender system tries to estimate the rating function  $R$  for the not-yet-rated- items by specific users. (This is explained in details in section 2.1). As noticed in this approach, the only considered dimensions are two (users and items).

Whereas, in the multidimensional approach, contextual information is incorporated into the recommendation process as an additional dimension. This requires the rating estimation function to be performed over a multidimensional space,  $R: \text{User} \times \text{Item} \times \text{Context} \rightarrow \text{Rating}$ .

Context-aware recommender systems are categorized into three categories: contextual pre-filtering, contextual post-filtering and contextual modeling. In the contextual pre-filtering approach, the contextual information is used as a label for filtering out those ratings that do not correspond to the specified contextual information. So, before launching the main recommendation method, the unrelated ratings in the sense of context are filtered out and the recommendation method is launched on the remaining reduced dataset. Whereas, in the contextual post-filtering approach, after launching the traditional recommendation method on the dataset and producing a list of recommendations, contextual information is used to contextualize the obtained recommendations. In contextual modeling, context is integrated directly into the model [28].

Baltrunas et al. proposed an item splitting method which extends the traditional collaborative filtering data model by assuming that each rating in a users-items matrix is stored together with a nominal piece of context information [42].

Panniello et al. applied contextual pre-filtering and post-filtering methods on two datasets and compared the results. They used two different post-filtering methods, weight and filter, as well as the exact pre-filtering method and applied them on the two datasets. Their results showed that the filter post-filtering method outperforms the exact pre-filtering, and that exact pre-filtering outperforms the weight post-filtering method on two datasets [43].

Another contribution for Baltrunas et al. who provided a general architecture of context-aware recommender systems and analyzed separate components of this model [44]. In his proposed architecture, there is a component named model adapter which is responsible for integrating contextual data into the prediction algorithm. Baltrunas et al. also introduced a context-aware (time-aware) recommendation approach called user micro-

profiling, in which they split each single user profile into several possibly overlapping sub-profiles, each representing users in particular contexts (times) [45].

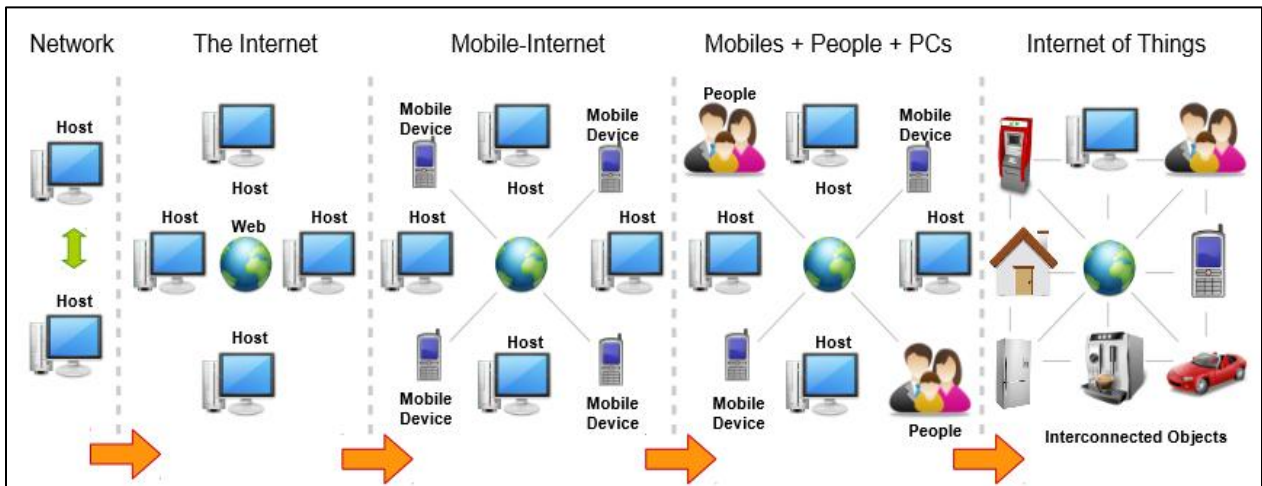
## 2.4. Internet of Things (IoT)

In this section, an introduction to the evolution of the internet starting from the simple network connecting two PCs to the IoT will be provided.

### 2.4.1. Evolution of Internet

Before going deep to the IoT, it is worthy to go back to the internet evolution history. In 1960s, a computer network between two computers had been established, making it possible for the two computers to communicate. Then in the early 1980s the TCP/IP was first presented.

Then in the late 1980s the commercial use of the internet started. In 1991, the World Wide Web became available and made the internet more popular. Later the mobile devices become smart enough to be connected to the internet and they formed the mobile-internet. Then social networking appeared and users started being connected socially together through the internet. The next step will be the IoT where objects around us will be connected to each other and communicate through the internet. Figure 2.2 shows these phases of the internet evolution [28].



**Fig.2.2:** The five phases of the Internet evolution [28]



The term Internet of Things or IoT refers to a wireless network between objects, usually the network will be wireless and self-configuring. Its principle is a comprehensive connectivity where anything (object) is connected to anyone, at anytime and anywhere. Applications to this will be all over our life, such as smart cities, agricultural applications, retail services, animal tracking, home automation, and many other applications. All objects in these applications can be connected to the internet to tell its status or any information it is exploited for. IoT is started to be applied gradually and partially in the world. Cisco Internet Business Solutions Group (IBSG) estimates that the IoT was “born” sometime between 2008 and 2009 [28].

Looking to the future, Cisco IBSG predicts that there will be 25 billion devices connected to the Internet by 2015 and 50 billion by 2020. These predictions rely on what is known nowadays, but the development of future devices may increase these predictions [9]. Many corporations such as Cisco’s Planetary Skin, HP’s central nervous system for the earth (CeNSE), and smart dust, have the potential to add millions -even billions- of sensors to the Internet [27]. According to BCC Research [28], in 2010 the global market of sensors was around \$62.8 billion and expected to increase to \$91.5 billion by 2016.

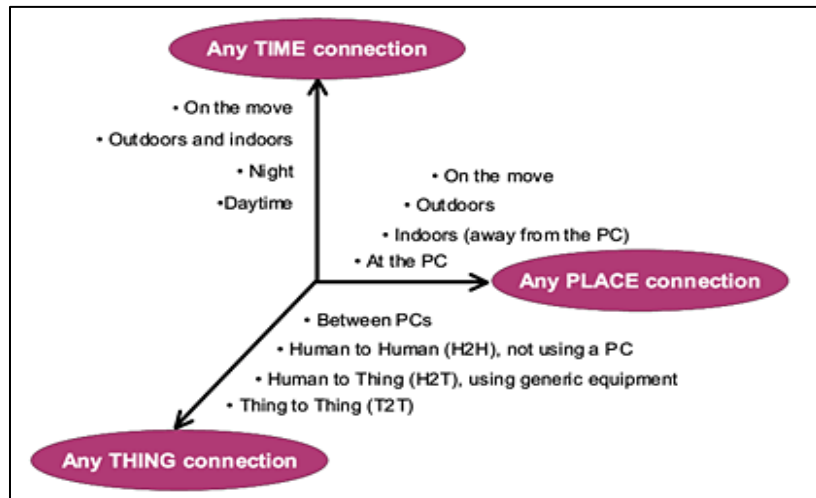
As cows, water pipes, people, and even shoes, trees, and animals become connected to the IoT, the world will have the potential to become a better place. A clear vision from Peter Hartwell, as Senior Researcher at HP labs is: “*With a trillion sensors embedded in the environment -all connected by computing systems, software, and services- it will be possible to hear the heartbeat of the Earth, Impacting human interaction with the globe as profoundly as the Internet has revolutionized communication*”[46].

#### **2.4.2. Internet of Things Appearance**

The IoT was coined as a term by Kevin Ashton in 1999 in the context of supply chain management. He has mentioned “*The Internet of Things has the potential to change the world, just as the Internet did. Maybe even more so*” [47]. Then in 2001 the MIT auto-ID center introduced their vision of the IoT. Later in 2005, the IoT was formally introduced by the International Telecommunication Union (ITU) in ITU Internet report.

In the past decade, the definition has been more broad covering variety of applications like healthcare, transport, utilities ... etc. [49]. Mark Weiser, the forefather of Ubiquitous Computing (ubicmp) defined a smart environment as *“the physical world that is richly and invisibly interwoven with sensors, actuators, displays, and computational elements, embedded seamlessly in the everyday objects of our lives, and connected through continuous network”*[48].

However, there is no any standard definition for IoT because the research in it still in its early stages. But some definitions were introduced by researchers. In [50] the author has defined the IoT as: *“Things have identities and virtual personalities operating in smart spaces using intelligent interfaces to connect and communicate within social, environment, and user contexts.”*. In [51], the IoT was introduced as: *“The Internet of Things allows people and things to be connected anytime, anyplace, with any-thing and anyone, ideally using any path/network and any-service.”*. The definition in [51] seems to be a comprehensive and expressive definition. Figure 2.3 illustrates this definition:



**Fig. 2.3:** IoT: anytime, anyplace, and anything [3]

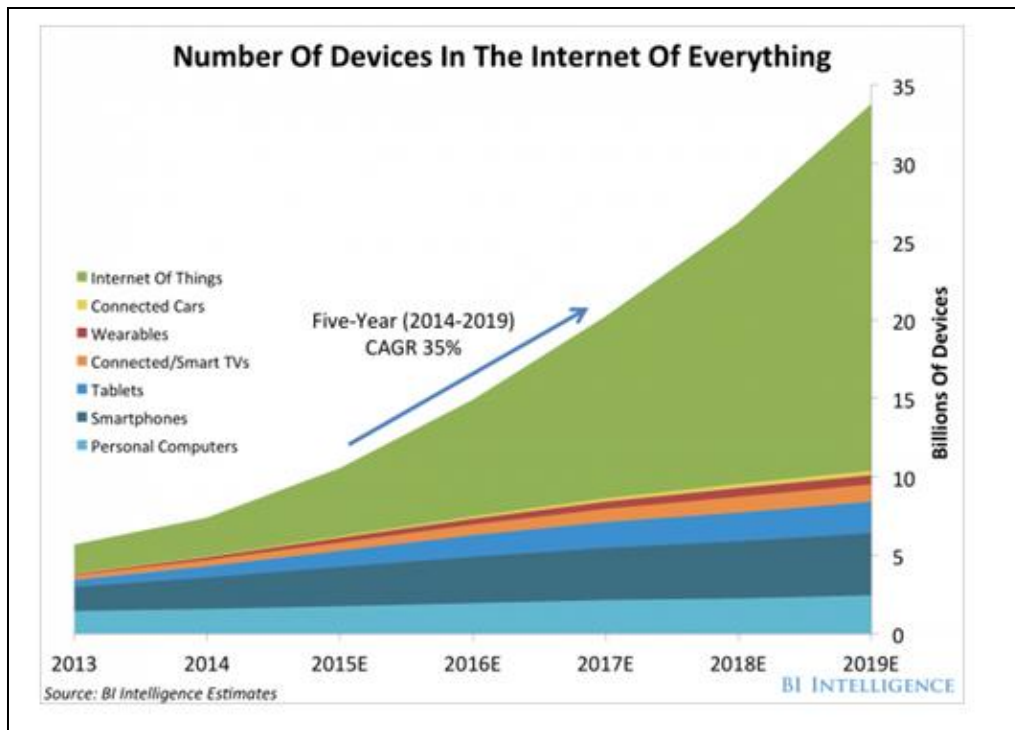
In this context, its worthy to mention the Internet of Everything (IoE) that encompasses the wider concept of connectivity from the perspective of modern connectivity technology use-cases. This concept had been defined by Cisco where a set up of an end-to-end ecosystem of connectivity including technologies, processes and concepts employed across all connectivity use-cases. IoE comprises of four key elements including all sorts of connections imaginable which are people, things, data and processes [4]. That

is why any other terms like the Internet of things, Internet of humans, Internet of digital will eventually constitute a subset of IoE.

### 2.4.3. Internet of Things Future Vision

All the indicators around us assure that the future is for the internet of things. The number of connected devices is growing rapidly, and IoT will become soon an integral part of businesses and society as a whole. The vision of the Internet of Things (IoT) is becoming reality and part of mainstream technology for the realization of digital services, applications, and solutions.

The graph shown on figure 2.4 revealing a prediction for the IoT market expansion through 2019. The prediction here is talking about 35 billion devices connected to the internet by 2019.



**Fig 2.4:** IoT Expansion Through 2019 [66]

According to the World Economic Forum, “by 2020, more than 5 billion people will be connected, not to mention 50 billion things” [4]. And according to a report by the UK government, “Today, there are about 14 billion objects connected to the Internet. Industry analysts estimate the number of connected devices could be anywhere from 20

*billion to 100 billion by 2020” [52]. And as said by IDC “ Within the next five years, more than 90% of all IoT data will be hosted on service provider platforms as cloud computing reduces the complexity of supporting IoT “Data Blending” [53].*

## **2.5. Proactivity in Recommender Systems**

The term “proactivity” has recently appeared in recommender systems. Originally, in organizational behavior, proactivity or proactive behavior by individuals refers to change-oriented and self-behavior in situations. It involves acting in advance of a future situation, rather than just reacting [54].

With the fast development of mobile computers that are available in the daily lives of the users, and the problem with interacting with all of these mobile devices and the systems that interacts with them, proactive computing appeared to handle this overflow. Applying this kind of computing to recommender systems produces the “proactive recommender systems” [55].

Traditional recommender systems need users to focus their full attention to the system. But Proactive recommender systems has changed this orientation. In [56], Salovaara and Oulasvirta consider a system proactive if it is working for or on behalf of the user. These recommenders decide to provide recommendations even if not explicitly requested. The largest majority of the recommender systems developed so far follow a “pull” model [57]; in which the user originate the request for a recommendation (request - response pattern). But these days, a new scenario is rising. Computers are ubiquitous and users are always connected through heterogonous devices (laptops, mobiles, iPads ...etc.). And that required to change the look to the recommendation origination. It seems natural that a recommender system must start to detect implicit requests without being explicitly triggered. This is called the push model, which seems to be very effective in the applications where the availability of items changes quickly, and where users and items can be monitored in real time.

Proactive recommender systems have the challenge not only in knowing what items to recommend, but also when and how to push these recommendations to users. Since if the

system pushes uninterested information to the user, or pushes interested information but at unfitting context, then the user's acceptance of proactively delivered recommendations will extremely decrease [58]. By the rest of this section, we will briefly focus light on research's that has been done in proactive systems generally, and proactive recommender systems particularly.

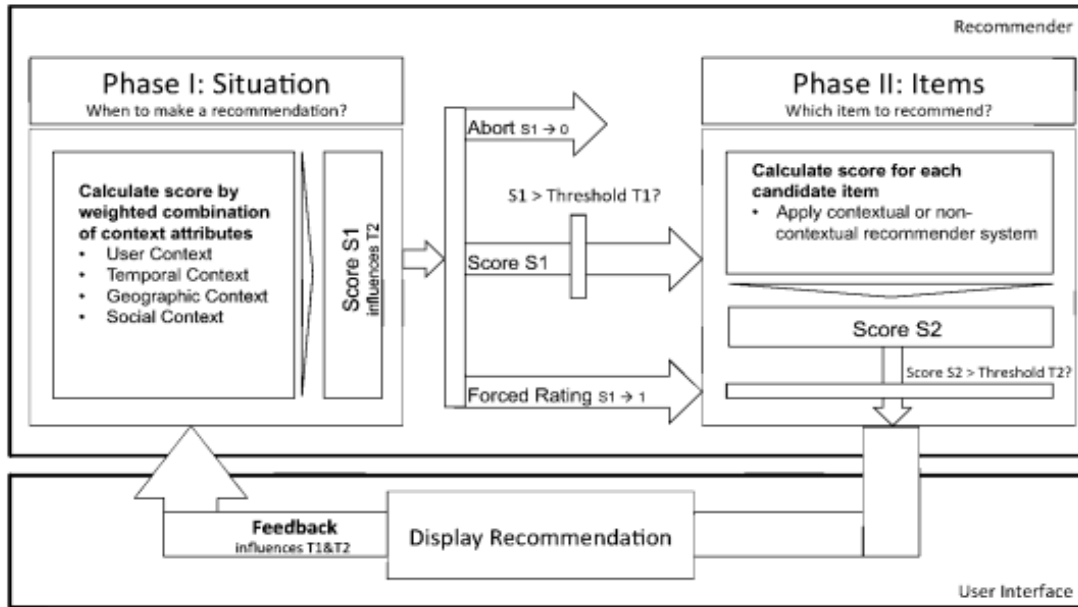
The delivery of information at the right time (Just-In-Time Information Retrieval (JITIR)) was first proposed by Rhodes in [59] who emphasized the advantage of this system over the interactive one. In [60], the author emphasized that the interface of any proactive information system must not be too intrusive in order to keep the user acceptance. And the information should be easy to access, noticeable to the user, and have the property of being ignorable in delivery to be retrieved later. In order to start proactive behavior, a JITIR assistant should be aware of local context of the user. Location is one of the most used context information for delivering recommendations. The idea is to push information related to the location of the user. For example, Rhodes in [61] have created a mobile version of JITIR called Jimminy, which is a wearable computer that automatically displays old notes relevant to current location. Notes are saved previously with their context. For example, by entering a room, all notes taken in that room are displayed. Many other researchers shared the same concept in which information is automatically saved and retrieved if the current context matches the saved one.

The Hippie/HIPS system was an early mobile guide system that parades proactive behavior [62]. This system approach offered guidance by using maps and indoor positioning technology at room and object level. It notified users of upcoming sights proactively. Podnar described a technical architecture to push information such as weather, traffic or news to a user. His architecture involved a user and a content provider. The content provider pushes information to the interested user from time to time [63].

For recommender systems, context can be used for two things: to trigger recommendations and to filter items. In [64] Vijayalakshmi and Kannan have classified the context into context that triggers the delivery and context that is used for assessment.

Ricci discusses proactivity in mobile recommender systems in his recent survey [65]. Personalization on mobile devices and in ubiquitous computing in general can be improved by considering user's context and situation. He concludes that proactive recommender systems still premature, because none of the reviewed systems is able to interrupt the user activity with unsolicited but relevant recommendations. And mentioned that proactive recommenders can revolutionize the role of recommender systems.

Wörndl has proposed a generic model for proactivity that relies on score calculation [66]. His model separates proactivity and context-aware assessment. A score  $S_1$  is calculated and triggers the calculation of a second score  $S_2$  for each item.  $S_1$  indicates whether a specific situation needs a recommendation. If yes, then  $S_1$  triggers the calculation of  $S_2$  that indicates the quality of an item. Situational parameters are a weighted combination of a user, temporal, geographic and social context. If  $S_1$  exceeds a threshold, the calculation of  $S_2$  is initiated. And if  $S_2$  for an item exceeds a threshold, then this item will be recommended. The figure Fig 4 shows the proposed two-phase proactivity model of Wörndl.



**Fig 2.5:** Proposed Two-phase Proactivity Model [66]

Moreover, in [67] Wörndl has tested a scenario of a context-aware restaurant recommender for Android smartphones relying on the model of Proactivity that he has

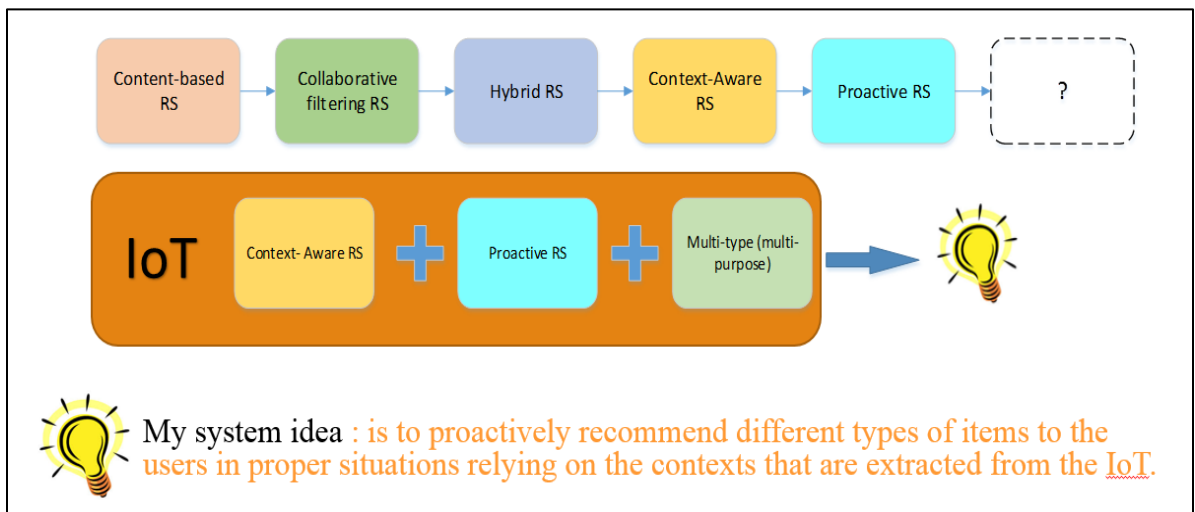
introduced in [66], then he has designed two options for the user interaction with a proactive recommender: a widget- and a notification-based solution.

Bedi in [58] followed the steps of Wörndl in [66]. She proposed a Situation-Aware and Proactive Recommender System (SAPRS) that pushes relevant items to the target user at the right context only. The recommendation process in this system is divided into two phases: the situation assessment phase and the item assessment phase. SAPRS uses fuzzy logic as an inference technique to handle uncertainty within situation assessment phase. SAPRS prototype has been designed and developed for restaurant recommendation, and was evaluated using user's subjective feedback.

Previous proactive recommender systems were proposed for one type of recommendations, such as restaurants as in [58], gas stations like in [66], but none of them was proposed for multi-types recommendations. Moreover, none of the previous systems was designed to cope with the IoT.

## 2.6. Proactive Multi-Type Context-Aware Recommender System

Throughout the previous sections, concepts and topics that are related to this thesis are studied and introduced. The main idea of this thesis is to design a proactive multi-type context aware recommender system in the environment of the IoT.



**Fig 2.6:** Thesis Idea Info-graphic

The info-graphic that is shown on figure 2.5 can best explain a chronological development of recommender systems that can illustrate how do we reach to our system idea. The recommender systems started with the content based approach that delivers recommendations according to the users' history. Then a collaborative filtering approach- in which the user is recommended items that are liked by other users with the same preferences with him- has appeared. After that a hybrid recommender system that uses both of the previous techniques has been introduced to decrease the shortcomings of both of them. Then after the appearance the context aware computing, incorporating context into recommender systems yielded the context-aware recommender systems in which the quality of recommendations extremely enhanced. For all of the previous recommender systems we were talking about request-response approach in which the user must explicitly request the recommendation to have it. Recently a new concept called Proactivity has appeared over context-aware recommenders. In this concept recommendations are pushed to the user when his current context is appropriate. This yielded a proactive context-aware recommender systems. Until now, most of the previous mentioned types of recommenders are single type. They only recommend one type of items ( books or hotels or restaurants or attractions ....etc.)

## **2.7. Artificial Neural Networks (ANNs)**

In our work we have used the ANN for context reasoning to assess the situation according to the user's context, and generate the scores of the different types of recommendations. This is explained in details in chapter 3.

In this section we provide a brief background about ANN, including the basic artificial model, the ANN classifications, the ANN learning techniques, and finally a brief summary of the back-propagation algorithm.

### **2.7.1. Background**

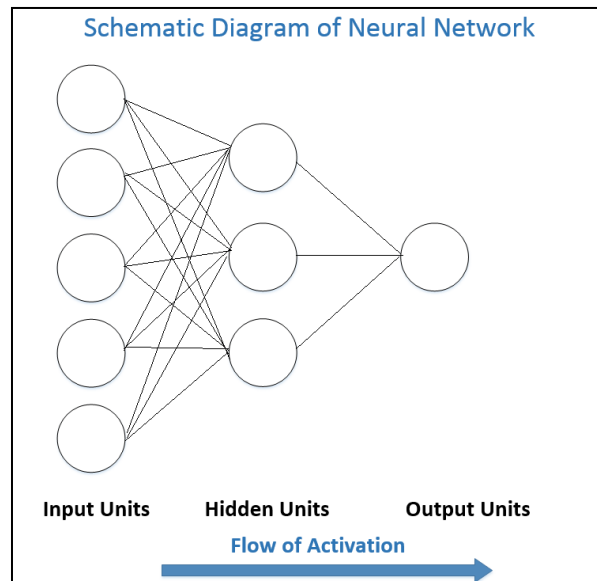
In machine learning, Artificial Neural Networks (ANNs) are statistical learning models that are inspired by networks of biological neurons. They are used to estimate functions that can depend on a large number of inputs and are generally unknown. ANNs gather



their knowledge by detecting the patterns and relationships in data and learn (or are trained) through experience, not from programming.

ANNs are generally presented as systems of interconnected "neurons" or "nodes" which exchange messages between each other. The connections have numeric weights that can be tuned based on experience (during a training process), making neural nets adaptive to inputs and capable of learning. Figure 2.6 below shows a schematic diagram of a neural network.

Successful training can result in artificial neural networks that perform tasks such as predicting an output value, classifying an object, approximating a function, recognizing a pattern in multi-factorial data, and completing a known pattern [68].



**Fig 2.7:** Schematic Diagram of ANN.

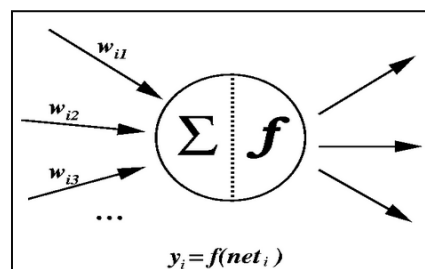
### 2.7.2. The Basic Artificial Model

Inspiring of the core of the biological neural systems, an artificial neuron or "node" is defined as follows [68]:

- It receives a number of inputs (either from original data, or from the output of other neurons in the neural network).
- Each input comes via a connection that has a weight; these weights correspond to synaptic efficacy in a biological neuron.

- Each neuron also has a single threshold value. The weighted sum of the inputs is formed, and the threshold subtracted, to compose the activation of the neuron.
- The activation signal is passed through an activation function (also known as a transfer function) to produce the output of the neuron.

So, our basic computational element (node) receives input from some other units, or perhaps from an external source. Each input has an associated **weight**  $w$ , which can be modified so as to model synaptic learning. Figure 2.7 shows a simple neuron inputs/outputs:



**Fig 2.8:** Neuron Inputs/Outputs

### 2.7.3. ANN Classification Based on Structure

There are different types of neural networks, but generally they are classified according to their structure into: feed-forward neural networks, and feed-back or recurrent neural networks.

- **A Feed-forward Neural Network:** is a network that has inputs, outputs, and hidden layers, where connections between the units do not form a directed cycle. The signals can only move in one direction. There are no limitations on number of layers, type of transfer function used in individual artificial neuron or number of connections between individual artificial neurons. Input data is passed into a layer where calculations are performed by processing elements in the hidden layers. Each processing element makes its computation based upon a weighted sum of its inputs. The new calculated values then become the new input values that are passed to the next layer. This process continues until it has gone through all the layers and determines the output.

- **A Recurrent Neural Network:** This type of networks is similar to feed-forward neural network except that its neurons send feedback signals to each other. In these cases information is no longer transmitted only in one direction but it is also transmitted backwards. This creates an internal state of the network. Recurrent artificial neural networks can use their internal memory to process any sequence of inputs. There are many types of recurrent networks, the most basic topology of recurrent artificial neural network is fully recurrent artificial network where every artificial neuron is directly connected to every artificial neuron in all directions [69].

#### 2.7.4. ANN Learning Techniques

There are three major learning paradigms; supervised learning, unsupervised learning and reinforcement learning. Each learning paradigm has many training algorithms. Below is a brief description of each of the three paradigms.

- **Supervised Learning** is the task of inferring a function from labeled training data. The training data consists of pairs of input and desired output values that are traditionally represented in data vectors (training examples). The concept in the supervised learning is to produce an inferred function after analyzing the training examples by the learning algorithm, then using this function to map new examples. A best scenario will allow for the algorithm to correctly resolve the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "logical" way. In supervised learning, solving a given problem passes with several steps. They are: first to determine the type of training examples, then to gather a training data set that describes the problem well after that we do the learning and after the learning, we can test the performance of learned artificial neural network with the validation (test) data set. Test data set consists of data that has not been introduced to artificial neural network while learning [70].
- **Unsupervised learning** is machine learning technique that studies how systems can learn to represent particular input patterns, by setting parameters of an artificial neural network based on given data and a cost function. Then, reflects the statistical

structure of the overall collection of input patterns. Cost function can be any function and it is determined by the task formulation. Unsupervised learning is mostly used in applications that fall within the domain of estimation problems such as statistical modeling, filtering, compression, and clustering. In unsupervised learning, we look for determining how the data is structured. It differs from supervised learning and reinforcement learning in that the artificial neural network is given only unlabeled examples. The Self-Organizing Maps (SOMs) are the ones that the most commonly use unsupervised learning algorithms [70].

- **Reinforcement learning** is a machine learning technique that sets parameters of an artificial neural network, where data is generated by interactions with the environment. The Reinforcement learning policy is learning what to do (how to map situations to actions) so as to maximize a numerical reward signal. The learner must discover which actions yield the most reward by trying them. Not as in other machine learning techniques, where the learner is told which actions to take. Reinforcement learning is frequently used as a part of artificial neural network's overall learning algorithm [70].

#### **2.7.5. Back-Propagation Algorithm**

The back propagation algorithm is used in layered feed-forward neural networks. As mentioned before, the layered ANN has neurons that are organized in layers, these neurons forward the signals to each other, then using the back propagation algorithm, the errors are propagated backward. The back propagation algorithm uses supervised learning, in which we provide the algorithm with examples of inputs and outputs that we want the network to compute, then the error is computed. Where the error here is the difference between the actual output and the computed output. The concept in the back propagation algorithm is trying to reduce this error until the ANN learns the training data. The training process starts with random weights on the connections between neurons, and keeps adjusting until minimizing the error to the maximum [71].

The activation function that is used to activate neurons in the ANNs that implement back-propagation algorithm is the “weighted sum”. For realistic reasons, ANNs that implement

the back propagation algorithm do not have too many layers, as the time for training the network grows exponentially [71].

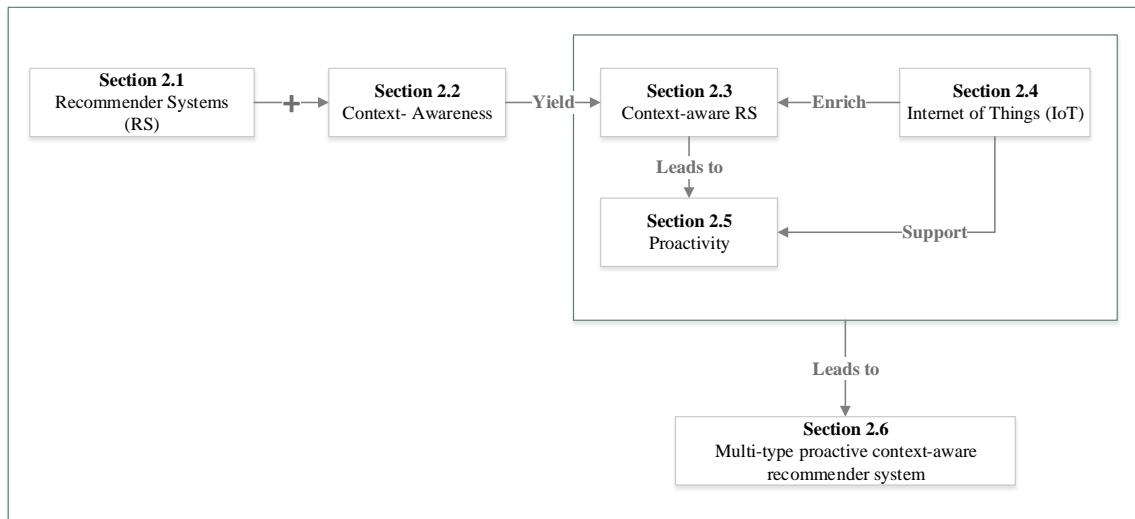
### 2.7.6. Conclusion

The computing world has a lot to get from neural networks. Their ability to learn by example makes them very flexible and powerful. Furthermore, there is no need set up an algorithm to achieve a task or even know how the task is internally achieved.

ANN also best fit real time applications where the fast response and low computational time is very important.

## 2.8. Chapter 2 Sections Interconnection

The flowchart in fig 2.9 demonstrates the relations between the different concepts and topics that were introduced in this chapter.



**Fig 2.9:** Chapter Two Sections Interconnection

# CHAPTER THREE

## SYSTEM DESIGN and IMPLEMENTATION

In this chapter, we present the general design of the proposed proactive multi-type context aware recommender system. Including the design of a main block in the system, which is the Context Aware Management System (CAMS). Inside this block, there will be a design for the context reasoning block using a neural network. The implementation and testing results of the CAMS will also be analyzed and presented, in addition to screenshots for the GUI that was developed for the CAMS showing different context-output scenarios.

### 3.1. General System Design

The proposed system model generally consists of two phases, the situation and recommender type phase, and the items recommender phase. The proposed system design is shown on figure 3.1.

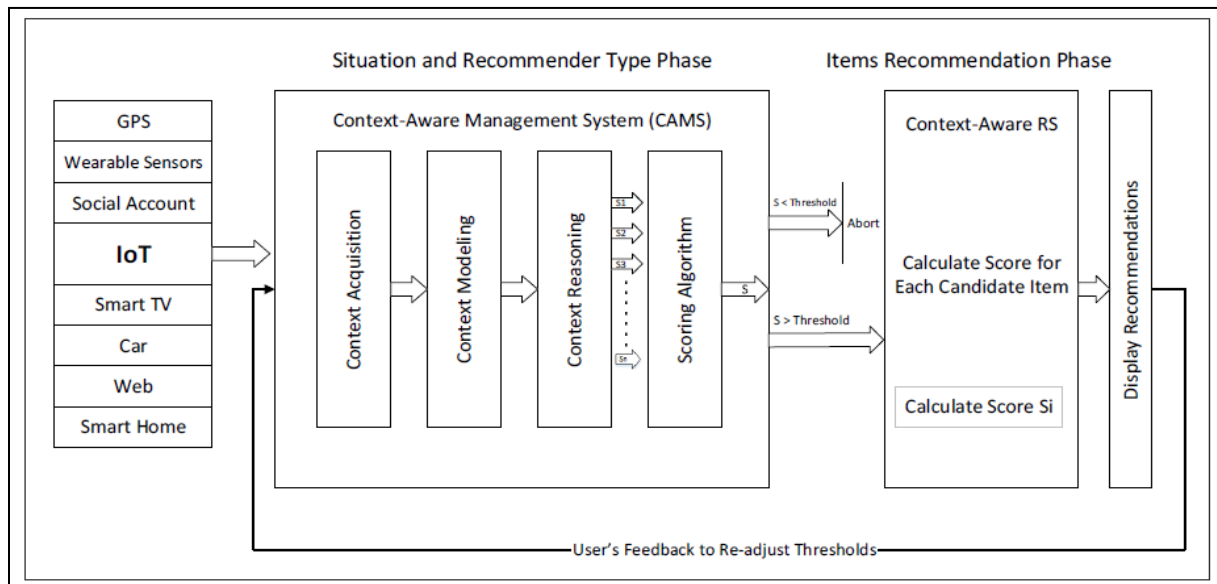


Fig 3.1: Proposed System Design

**Situation and Recommender Type Phase:** the main function of this phase is to determine whether a situation is proper to push a recommendation or not, and to determine what type of recommendations to push from a pre-defined set of recommendation types. This phase is encapsulating a design of a CAMS relying on the context lifecycle introduced in [28]. The details of the CAMS design is introduced in the next section.

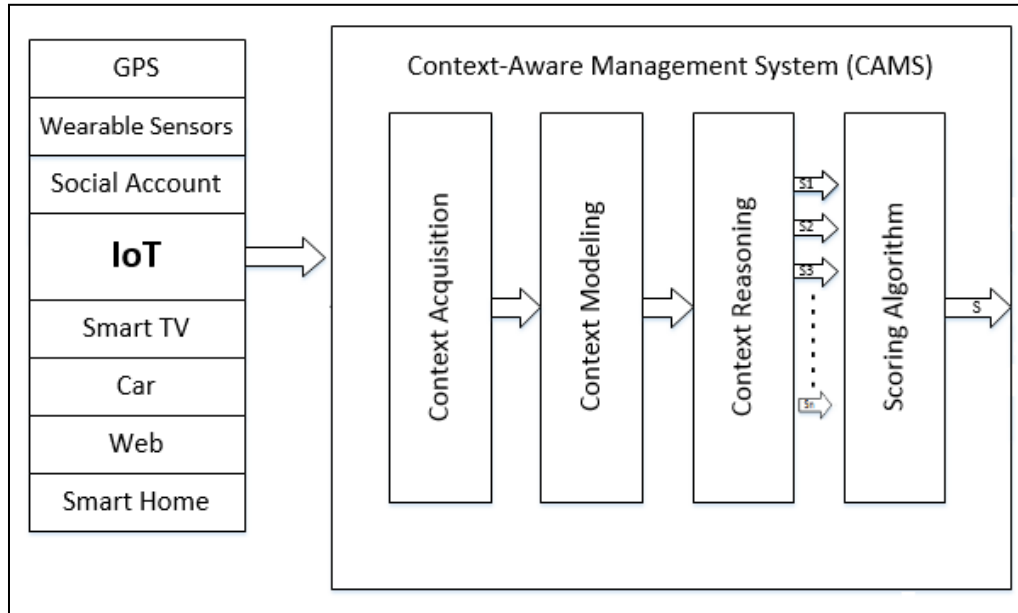
**Items Recommender Phase:** this phase is triggered by the previous phase. This is a traditional context aware recommender system based on the collaborative filtering technique. As in [66], new scores are calculated for the items, then the items whose scores are above a threshold will be displayed to the user ranked from max to min. These scores correspond to the predicted items ratings of a standard collaborative filtering. The threshold with which the items scores is compared is adaptive and changes according to the user feedback which is given by liking and disliking the recommended items.

### **3.2. Context Aware Management System Design (CAMS)**

The CAMS is a major block in the proposed system. The CAMS will enable the system to decide if the situation is proper for a recommendation, and it will select the type of item that may be recommended to the user.

The CAMS consists of four stages: the first stage is the data acquisition, which retrieves the raw data from the IoT. The second stage is the data modelling that models (represents) the raw data to be understood, like in key-value modelling, markup scheme, graphical modelling, and object-based modelling. The third stage is the context reasoning, which is the method of deducing new knowledge based on the available context. The fourth stage is the scoring algorithm, which generates an output score based on the output of the context reasoning block. In our design the reasoning phase is implemented using a neural network that generates different scores for the recommendation types based on the context, then the maximum of these scores is considered; i.e. the scoring algorithm takes the maximum of the input scores. If this

maximum score is less than a pre-determined threshold, then the next phase will not be triggered. But if this score is greater than the threshold, then the next phase in which a specific items belong to the type that got the maximum score will be triggered. A block diagram of the CAMS is shown in figure 3.2.



**Fig 3.2:** Context Aware Management System (CAMS)

In the following sections we will describe the inputs and explain the design of each block of the CAMS.

### 3.2.1. IoT Sensors Data

For the three types of recommendations: gas stations, restaurants, and attractions, a set of attributes were set to determine the user's context, thus to affect the types scores. These are: time, gas sensor (tank empty or not), car on/off, hungry, outside the country, and holiday. We have initialized 115 records of these attributes with the expected outputs for them. The gas stations were given the highest priority then the restaurants then the attractions. These records were important to train the neural network in the next step, since we have followed the supervised learning approach in our machine learning technique. Figure 3.3 shows a snapshot of these records.



Location	Time	Inputs						Outputs		
		Gas Sensor	Car On/Off?	Hungry?	Outside the country?	Holiday?	Temperature	S1/Res	S2/Gas	S3/Att
Ramallah	0:00	0	1	0	0	0	22	0.1	0.1	0.1
Ramallah	0:00	1	1	0	0	0	23	0.1	0.92	0.1
Rome	0:00	1	1	1	1	0	18	0.2	0.92	0.15
Amman	0:00	1	1	0	1	0	20	0.1	0.92	0.15
Jenin	1:00	0	1	1	0	0	30	0.2	0.1	0.1
Istanbul	2:00	0	1	0	1	0	24	0.1	0.1	0.15
Ramallah	3:00	1	1	1	0	0	22	0.2	0.92	0.1
Amman	4:00	0	1	1	1	0	18	0.2	0.1	0.15
Istanbul	5:00	1	1	1	1	0	22	0.2	0.92	0.15
Dubai	6:00	1	1	0	1	0	32	0.1	0.92	0.15
Jenin	7:00	1	1	0	0	0	30	0.1	0.9	0.1
Ramallah	8:00	0	1	1	0	0	21	0.75	0.1	0.1
Dubai	9:00	0	1	0	1	0	35	0.1	0.1	0.5
Nablus	10:00	1	1	1	0	0	24	0.75	0.9	0.1
Amman	11:00	0	1	1	1	0	15	0.75	0.1	0.5
Istanbul	12:00	1	1	1	1	0	18	0.8	0.9	0.5
Paris	13:00	1	1	0	1	0	10	0.1	0.9	0.5
Ramallah	14:00	1	1	0	0	0	20	0.1	0.9	0.1
Ramallah	15:00	0	1	1	0	0	22	0.7	0.1	0.1
Istanbul	16:00	0	1	0	1	0	26	0.1	0.1	0.5
Ramallah	17:00	1	1	1	0	0	22	0.7	0.9	0.1
Amman	18:00	0	1	1	1	0	19	0.75	0.1	0.5

**Fig 3.3:** IoT Virtual Data

### 3.2.2. Context Acquisition

The first step in any context-aware application is the context acquisition, in which context is acquired from various sources. The sources could be physical sensors or virtual sensors. As it was mentioned in chapter 2, the techniques used to acquire context from real world can be varied based on responsibility, frequency, context source, sensor type, and acquisition process. In our implementation, we assume a ready acquired data from virtual existing sensors as it is difficult to get real physical IoT sensors data.

### 3.2.3. Context Modelling

Context modelling is also referred as context representation. It is the phase in which the sensors data is given a meaning to be understood. There are many context modelling techniques, these were discussed in chapter 2. In our implementation we have used the simple key-value modelling. We have chosen this technique because the context then is easy to be managed, especially that we didn't apply it on a huge amount of data, and because it suits the purpose of temporary storage such as less complex application configurations and user preferences. Fig 3.3 also shows the key-value pairs. Both context acquisition and context modelling phases were implemented simply in our work; the

context reasoning phase is the complex phase that incorporates artificial intelligence, this phase will be explained in the following section.

#### **3.2.4. Context Reasoning**

For context reasoning, we used the ANN technique. Below the details of the used ANN design, training and results analysis are presented.

#### **The Neural Network Design**

A two-layer feed-forward neural network with sigmoid hidden neurons and linear output neurons is designed with 6 inputs, 3 outputs, and 16 hidden neurons in its hidden layer. Figure 3.4 shows our neural network architecture. Below are the specifications of each layer design.

- **The Input Layer**

Every ANN has only one input layer. The number of neurons comprising this layer, was completely and uniquely determined once we have knew the shape of our training data. Specifically, the number of neurons comprising this layer is equal to the number of features (columns) in our data, which were six columns.

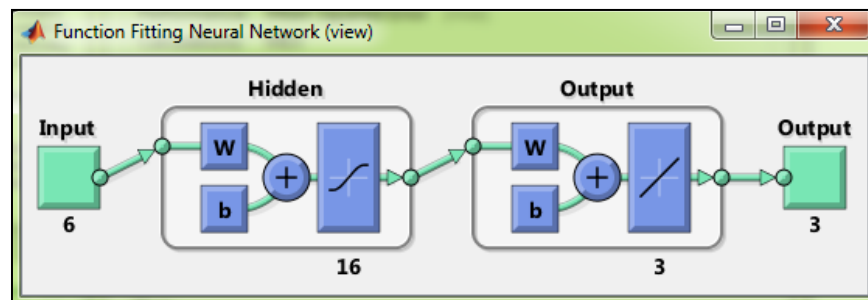
The 6 inputs represent the 6 sensor/status raw data of the user which are: time, gas sensor (tank empty or not), car on/off, hungry, outside the country, and holiday.

- **The Output Layer**

Like the input layer, every ANN has exactly one output layer. Determining its size (number of neurons) is simple; it is completely determined by the chosen model configuration. And as we have designed it to generate three scores, the output layer was designed with three outputs. The 3 outputs represent the scores of the three types of recommendations, where these scores are given to the types according to the user's context and need. The network has 3 output neurons, because there is 3 target values associated with each input vector.

- **The Hidden Layer**

We have used one hidden layer because the situations in which performance improves with a second (or third, etc.) hidden layer are very small. One hidden layer is sufficient for the large majority of problems. The number of the neurons used in the hidden layer were 16 neurons. We have specified this number of neurons based on experimental analysis as we found that we got a very good results when using this number of neurons.



**Fig 3.4:** Neural Network Design

### The Neural Network Training

The network was trained using scaled conjugate gradient back propagation ([trainscg](#)). This function was used for training because it uses less memory than the other training functions.

**The first step** before the training was to load the input vectors and target vectors into the workspace as shown in fig 3.5 and fig 3.6.

Variables - input										
input target										
6x115 double										
	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	1	2	3	4	5	6
2	0	1	1	1	0	0	1	0	1	1
3	1	1	1	1	1	1	1	1	1	1
4	0	0	1	0	1	0	1	1	1	0
5	0	0	1	1	0	1	0	1	1	1
6	0	0	0	0	0	0	0	0	0	0

**Fig 3.5:** ANN Input Vectors

	1	2	3	4	5	6	7	8	9	10
1	0.1000	0.1000	0.2000	0.1000	0.2000	0.1000	0.2000	0.2000	0.2000	0.1000
2	0.1000	0.9200	0.9200	0.9200	0.1000	0.1000	0.9200	0.1000	0.9200	0.9200
3	0.1000	0.1000	0.1500	0.1500	0.1000	0.1500	0.1000	0.1500	0.1500	0.1500

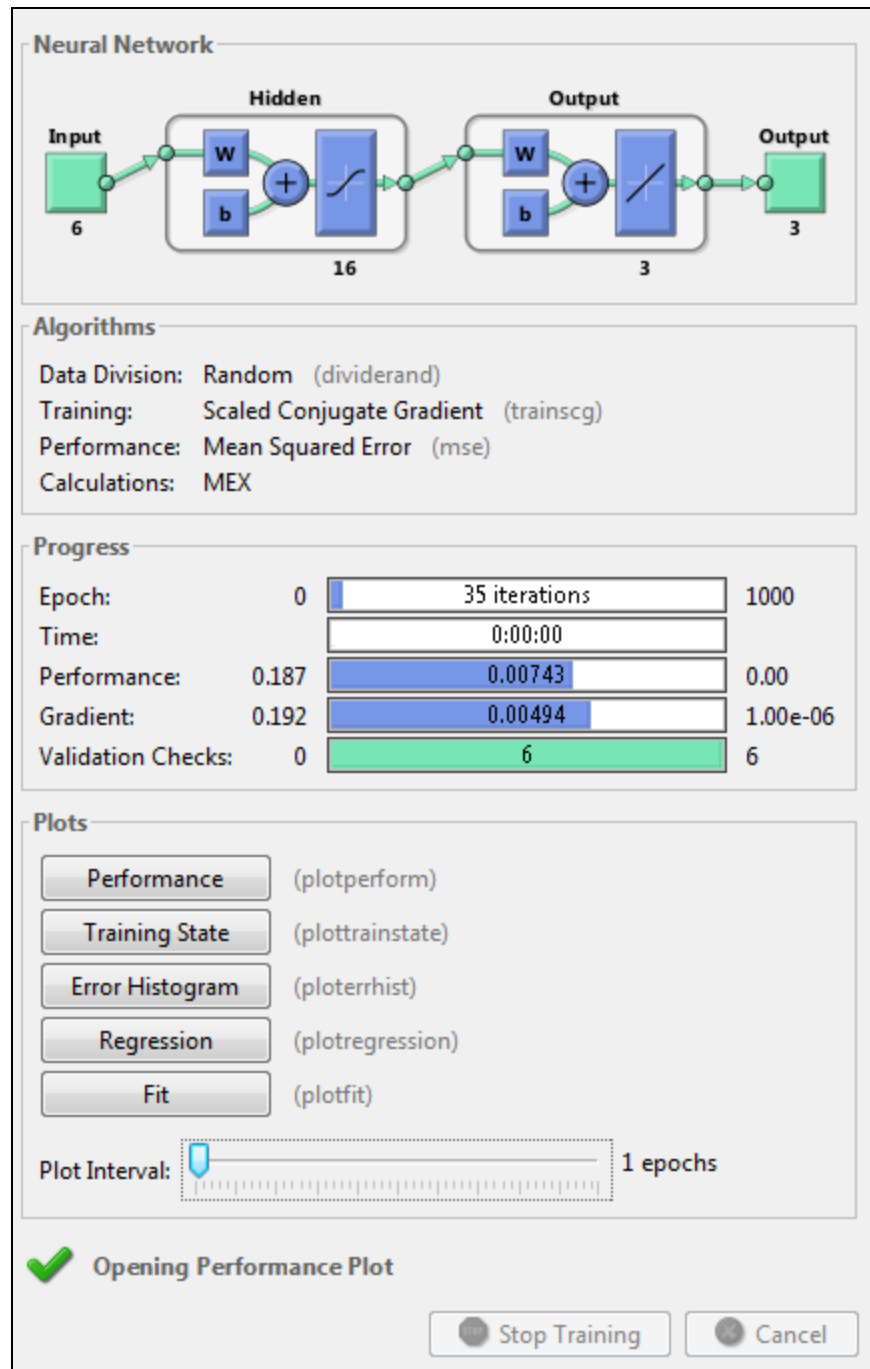
**Fig 3.6:** ANN Output Vectors

**The second step** was to create a network. The network which was used is a feed-forward network with the default tan-sigmoid transfer function in the hidden layer and linear transfer function in the output layer. we assigned 16 neurons (after testing) to the one hidden layer.

**The third step** was to set up the division of data. The input vectors and target vectors were randomly divided, with 70% used for training, 15% for validation and 15% for testing.

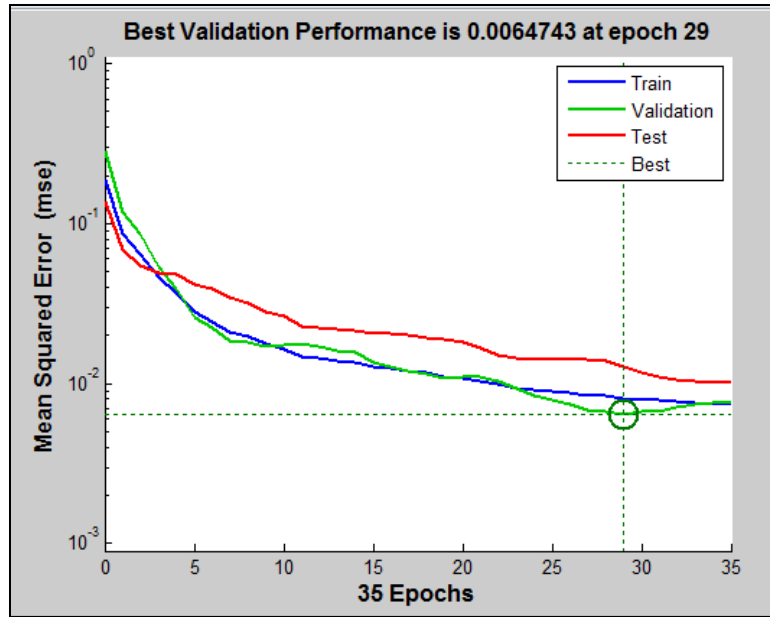
**The fourth step** was to train the network, as mentioned before, the scaled conjugate gradient back propagation ([trainscg](#)) function was used for training. During training, the training window shown in fig 3.7 opened.

This training stopped when the validation error increased for six iterations, which occurred at iteration 35.



**Fig 3.7: ANN Training Window**

The performance of our trained network can be found as a plot of the training errors, validation errors, and test errors appears, as shown in fig 3.8.



**Fig 3.8: ANN Performance**

In the above figure, we have considered the result reasonable because of the following considerations:

- The final mean-square error is small.
- The test set error and the validation set error have similar characteristics.
- No significant over-fitting has occurred by iteration 29 (where the best validation performance occurs).

**The fifth step** was to test the network. After the network has been trained, it was used to compute the network outputs.

Many trials were made to get as accurate results as possible. The following approaches were done to increase the accuracy of the results:

- The number of hidden neurons were increased from 10 to 16.
- The number of training vectors were increased from 95 to 115.
- Different training algorithms were tried.
- The initial network weights and biases were reset to new values and trained again.

## ANN Results and Analysis

After applying the previous steps, Our network became ready to predict the outputs of any input vectors. To be able to analyze the results thoroughly, a Matlab code that generates random input records (random user's context) was written. The generated random records were entered to our trained neural network; then it was run 5000 times to generate 5000 different recommendations types triggering.

The highest priority was given to the gas stations, so that a recommendation will be pushed if the score gained for the gas station is more than 0.7. then for the restaurants so that a recommendation will be pushed for the restaurants if the score gained is more than 0.5, and the least priority is given for attractions with threshold 0.35.

Running the previous code has resulted in an excel file of 5000 random contexts samples with their outputs that are predicted by the neural network. Fig 3.9 shows a screenshot from this excel file.

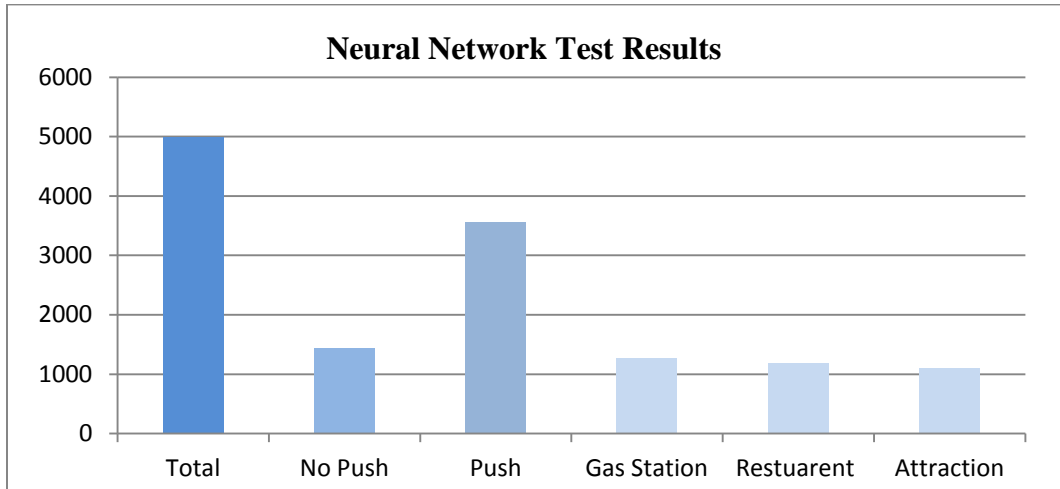
	A	B	C	D	E	F	G	H	I	J	K
1	time	gas tank	car on	hung	outside	holi	res	gas	att	push?	type
2	8	1	1	1	0	0	0.603868	0.86334	0.143375	1	2
3	22	0	0	0	1	0	0.037912	0.153799	0.468496	1	3
4	4	0	0	0	0	1	0.357034	0.046915	0.520604	1	3
5	19	1	0	1	1	0	0.768738	0.2143	0.547518	1	1
6	19	0	1	1	1	1	0.742021	0.091387	0.494938	1	1
7	4	0	0	0	0	0	0.31814	0.127481	0.222923	0	1
8	19	1	1	0	1	0	0.087824	0.913939	0.474901	1	2
9	18	0	0	1	0	1	0.830016	0.115403	0.276692	1	1
10	7	1	0	1	1	1	0.350899	0.241031	0.270611	0	1
11	13	1	0	1	1	1	0.791722	0.239136	0.539267	1	1
12	4	1	0	0	1	1	0.155213	0.249074	0.40574	1	3
13	22	0	0	0	0	1	0.127275	0.075043	0.47127	1	3
14	21	0	0	1	1	0	0.887439	0.140018	0.581808	1	1
15	1	0	0	0	0	1	0.261638	0.034078	0.439682	1	3
16	6	0	1	0	0	0	0.226413	0.124332	0.263376	0	3
17	0	1	1	0	1	0	-0.02641	0.90136	0.099777	1	2
18	17	0	0	0	0	0	0.106862	0.134501	0.23971	0	3
19	16	1	1	1	0	1	0.781453	0.906099	0.407508	1	2
20	15	1	1	1	1	1	0.841922	0.881927	0.562548	1	2
21	19	1	1	0	0	1	0.138177	0.942818	0.590161	1	2
22	21	1	1	0	0	0	0.084517	0.926578	0.153012	1	2

**Fig 3.9:** Matlab Code Output (ANN Contexts-Types Scores)

The results of running this code were 3559 of 5000 to trigger a type of the recommenders (push recommendation), and 1441 not to push any recommendation. And 1271 out of the 3272 pushes were gas stations, 1190 were restaurants and 1098 were attractions. These results are shown in table 3.1 and figure 3.10.

Total	5000
No Push	1441
Push	3559
Gas Station	1271
Restaurant	1190
Attraction	1098

**Table 3.1:** ANN Results Count



**Fig 3.10:** ANN Test Results

The above results were analyzed, and the records of each push type were filtered into one of the four classes: true positive, true negative, false positive, and false negative. Where:

- True positive (TP): the neural network triggers the true type when it must trigger it.
- True negative (TN): the neural network does not trigger the type when it must not trigger it.
- False positive (FP): the neural network does trigger the type when it must not trigger it.
- False negative (FN): the neural network does not trigger the type when it must trigger it.

Relying on the above definitions, the results of records analysis are shown in table 3.2.

Type/Result	TP%	FN%	TN%	FP%
Gas	100	0	100	0
Restaurant	97.8	2.2	99.6	0.4
Attractions	93.1	6.9	96.2	3.8

**Table 3.2:** ANN Results Analysis



The accuracy then is calculated from the above table and the following results were found:

- Accuracy (Gas)= 100%
- Accuracy (Restaurant) =99.16%
- Accuracy (Attraction) =95.62%

**The accuracy average for the above types is 98.26%** which is considered a very satisfactory result for deciding whether to push or not to push a recommendation, and what type of recommendation to trigger.

### **3.2.5. Scoring Algorithm**

This is the final stage in the CAMS. This phase takes the outputs of the context reasoning block (which are scores for the types of recommendations) as input and yields one aggregated score as an output. Depending on the output of this phase the CAMS decide to push a recommendation or not and decides what type of recommendations to push after comparing it with a pre-defined threshold. In our implementation, the scoring algorithm takes the maximum of the type's scores as the algorithm output. This is because we have trained the ANN by giving priorities to the types by giving the highest priority type a higher score according to the context. This algorithm may be different if the ANN is trained in a different way. The Matlab codes that implements the CAMS can be found in Appendix A

### 3.3. Recommender Type Triggering GUI

To easily track the output of the CAMS, a graphical user interface of this phase of the system was created using Matlab, where the input data can be entered manually, then run the ANN by pressing the “Calculate Score” button to get the scores, then get the triggered type of recommender as output in the same GUI. Snapshots of this GUI is shown in figures 3.3.a-3.3.d by giving different scenarios.

- In the first scenario (3.3.a), the user context is considered to be: Time is 2:00 PM, his car is on, the gas tank is below threshold, he isn't hungry and it is holiday, the ANN gave the maximum score to the gas stations as shown on the figure, thus by the scoring algorithm which takes the maximum score and compare it with the predefined threshold (0.7), the decision will be to trigger the gas stations type to be recommended to the user, this was the best recommendation that the user should receive in his context.

**Context Reasoning Using ANN - Recommender Type Triggering**

**Input**

Time: 14

☒ Car On

☒ Gas Tank Below Threshold

☐ Hungry

☐ Outside Country

☒ Holiday


Temp: 22

**Calculate Score** **Reset**

**Output**

Restuarent's Score	0.053116
Gas Station's Score	0.862449
Attraction's Score	0.3284

**Push a gas station Recommendation**



**Please Give your Feedback:**

**Yes, I need this** **Not Now**

**View NNT Arch** **View NNT Performance** **Run with Random Vector**

**Fig 3.3.a:** Recommender Type Triggering GUI- Gas stations type is triggered.

- In the second scenario (fig 3.3.b), the user context is considered to be: Time is 3:00 PM, the user car isn't on and the gas tank isn't below the threshold, he is hungry, outside his country and it is a holiday, the ANN gave the maximum score to the restaurants as shown on the figure, thus by the scoring algorithm which takes the maximum score and compare it with the predefined threshold (0.5), the decision will be to trigger the restaurants type to be recommended to the user, this was the best recommendation that the user should receive in his context.

**Context Reasoning Using ANN - Recommender Type Triggering**

**Input**

Time: 15

☐ Car On

☐ Gas Tank Below Threshold

☒ Hungry

☒ Outside Country

☒ Holiday

Temp: 22

**Calculate Score** **Reset**

**Output**

Restuarent's Score: 0.768421

Gas Station's Score: 0.0877443

Attraction's Score: 0.497041

**Push a Restuarent Recommendation**

**Please Give your Feedback:**

**Yes, I need this** **Not Now**

**View NNT Arch** **View NNT Performance** **Run with Random Vector**

**Fig 3.3.b:** Recommender Type Triggering GUI- Restaurants type is triggered.

- In the third scenario (fig 3.3.c), the user context is considered to be: Time is 11:00 AM, the user car isn't on and the gas tank isn't below the threshold, he is not hungry, he is outside his country and it is a holiday, the ANN gave the maximum score to the attractions as shown on the figure, thus by the scoring algorithm which takes the maximum score and compare it with the predefined threshold (0.35), the decision will

be to trigger the attractions type to be recommended to the user, this was the best recommendation that the user should receive in his context.

**Context Reasoning Using ANN - Recommender Type Triggering**

**Input**

Time: 11

☐ Car On

☐ Gas Tank Below Threshold

☐ Hungry

☒ Outside Country

☒ Holiday

Temp: 18

**Calculate Score** **Reset**


**Output**

Restuarent's Score: 0.174765

Gas Station's Score: 0.0854594

Attraction's Score: 0.45747

**Push an attraction Recommendation**



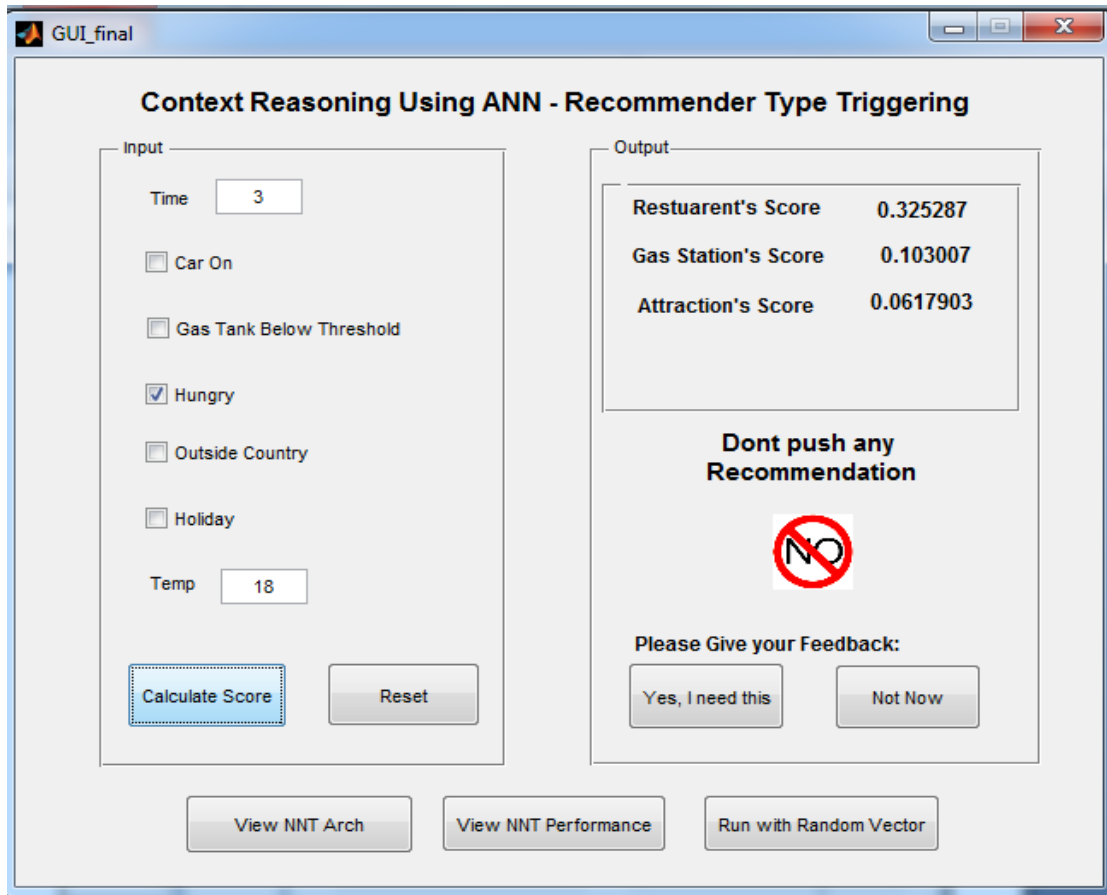
**Please Give your Feedback:**

**Yes, I need this** **Not Now**

**View NNT Arch** **View NNT Performance** **Run with Random Vector**

**Fig 3.3.c:** Recommender Type Triggering GUI- Attractions is triggered.

- In the fourth scenario (fig 3.3.d), the user context is considered to be: Time is 3:00 AM, the user car isn't on and the gas tank isn't below the threshold, he is hungry, he is not outside his country and it is not a holiday, the ANN gave the maximum score to the restaurants, but this score was not enough to trigger the next phase as it was less than the predefined threshold as shown on the figure. The decision will be not to trigger any type to be recommended to the user. This was the best decision, as it doesn't make sense to recommend a restaurant for a user after midnight even if he is hungry.



**Fig 3.3.d:** Recommender Type Triggering GUI- No type is triggered.

Also using this GUI, the neural network architecture and performance can be viewed using buttons.

### 3.4. Summary

This design is based on a model of Proactivity in which recommendations of different types are pushed to the user when his context seems appropriate without his explicit request. Most of the work is concentrated in designing and implementing the context aware management system (CAMS) which assesses the situation to decide whether to push a recommendation or not and what type of recommendations to push. This CAMS is designed to include four stages which are: context acquisition, context modelling, context reasoning, and a scoring algorithm. To test the system design, it was assumed to have three types of recommendations which are gas stations, restaurants, and attractions. Using Matlab, we were able to implement and test the CAMS and decide what type of

recommendations to push according to the user's context, which was virtually entered to Matlab representing the sensors values. The context modelling stage was simply implemented by the key-value mapping, and the context reasoning stage (which is the hardest) was implemented by using a feed-forward artificial neural network that was designed and trained to calculate the scores of the three previously mentioned types, depending on its inputs that represent the user's context. Then the role of the scoring algorithm appears, it was to choose which type to recommend according to the type's scores. In our implementation the scoring algorithm was as simple as taking the maximum score and recommend the type that corresponds to it if this maximum is more than a pre-defined threshold. The accuracy of the used neural network in recommending the correct types in the appropriate contexts was 98.26% which is an excellent percentage.

# CHAPTER FOUR

## PROTOTYPING and USERS' EVALUATION

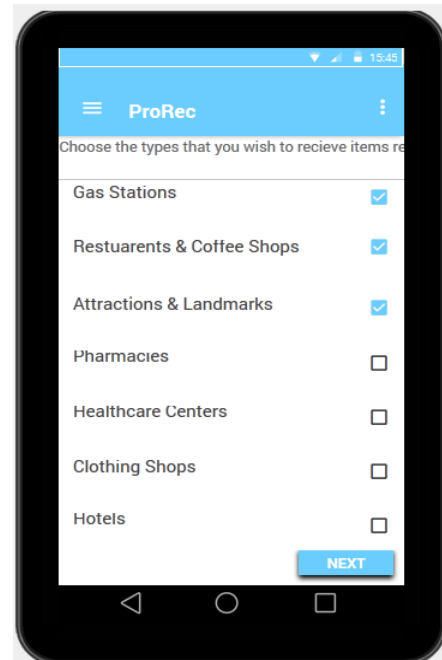
A prototype is a representation of a software program that includes only a few aspects of the concluding application and resembles it up to some degree, and whose purpose is to illustrate how that application is to look and behave to get an impression of its capabilities and shortcomings. Usually prototyping is built to understand the requirements and let the client get an “actual feel” of the system, since the interactions with prototype can enable the client to better understand the requirements of the desired system. Our prototyping is a bit different. The main reason behind prototyping our system was to get the user's acceptance evaluation of this system, especially that the multi-type concept in recommender systems is new, and we need to get the user's feedback about their acceptance of the reality of such a system. In this chapter, we introduce the prototype that we have developed, in which we demonstrated how the system responds in some different user's contexts, if it was developed as a service-oriented application. In addition to a survey results analysis for the users who tried the prototype or watched a video about the system prototype.

### 4.1 System Prototype -As an Application

We have used Justinmind Prototyper [72] to develop our system prototype. Justinmind Prototyper is a quick prototyping tool that allows us to create interactive and accurate simulations of the applications we need. With Justinmind Prototyper, we can easily incorporate any corporate image to our prototypes, export them in HTML format to demonstrate them online or automatically generate all of the documentation in an Open Office or MS Word file.

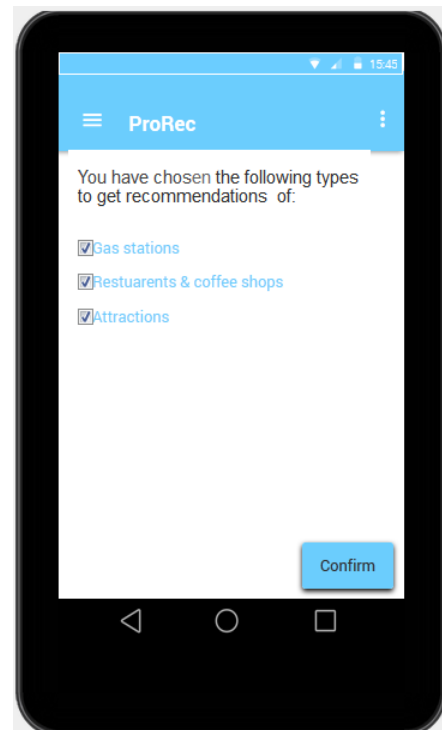
The following screenshots show the prototype of the proposed application that was called “ProRec”.

- Whenever you download the application on your smart device, you should for the first time check the types of items that you wish to receive recommendations about.



**Fig 4.1.a**

- By pressing the next button, a confirmation message will appear to you of the types you have selected.



**Fig 4.1.b**



- After confirming the selected types, ProRec process starts to work on the background.
- It pushes recommendations of one of the specified types when time and context seems appropriate.



**Fig 4.1.c**

### **Example Scenarios**

#### **Scenario # 1**

A user who has the ProRec on his Smartphone is walking in Al-Masyoune, Ramallah. It is lunch time.

ProRec will send a notification to the user of a recommended restaurants to take his lunch in.

- If the user is interested, he should press on the 'View button', if not, he should press on the 'Not Now' button or simply ignore the notification.
- He can view these restaurants with their ratings and the option to view them on map, in addition to the 'like' and 'dislike' options.
- These restaurants will be recommended according to their closeness, rate, delivered menu, etc... to best fit the user's preferences.

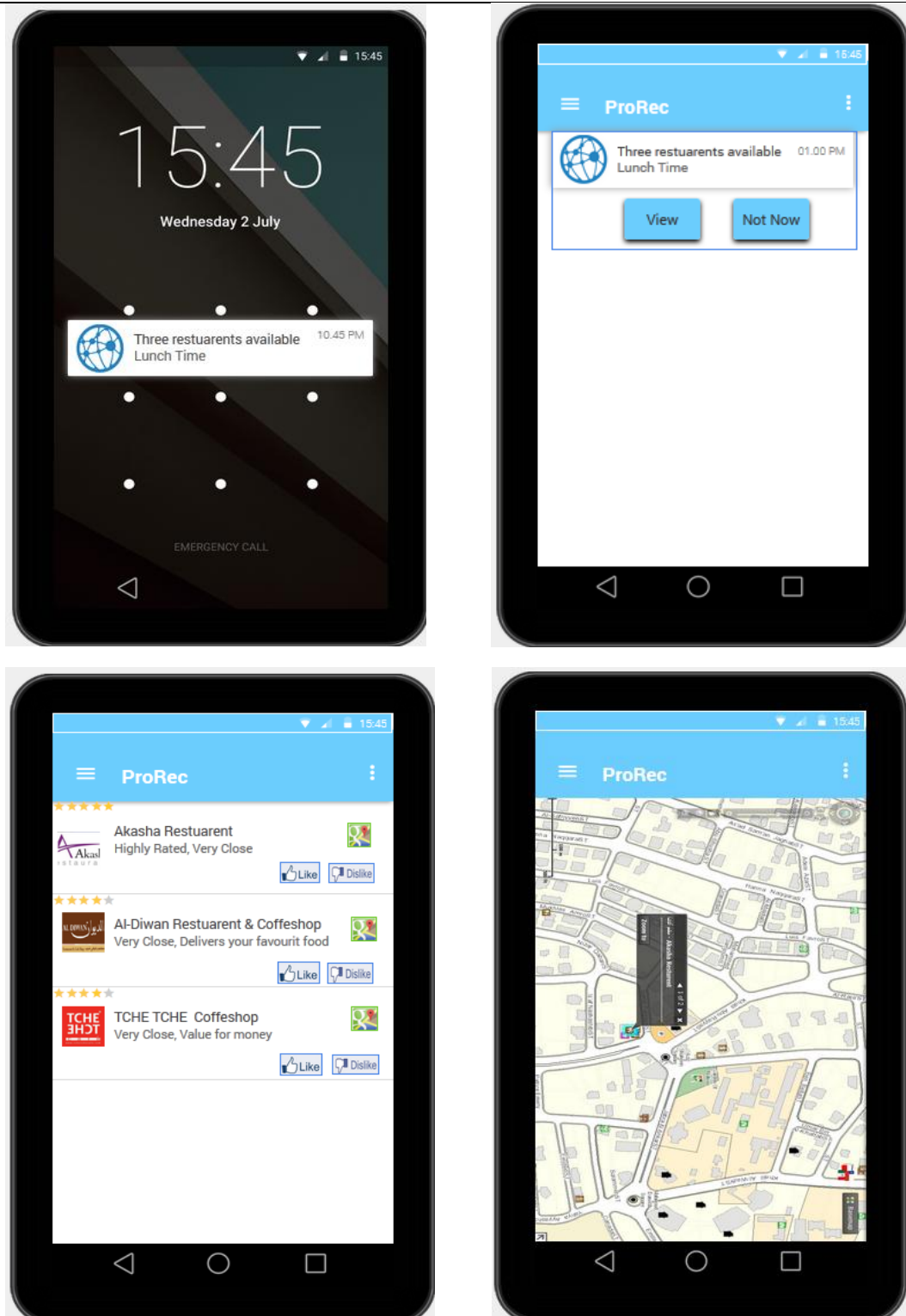


Fig 4.1 d-g

## Scenario # 2

- The same user is driving his car at al-Irsal street going from Ramallah to Birzeit.
  - His car fuel level became low, this is known from the fuel sensor on his car that is connected to the IoT.
  - ProRec will show on his car navigator a notification to view the available gas stations telling why this notification is being popped up.



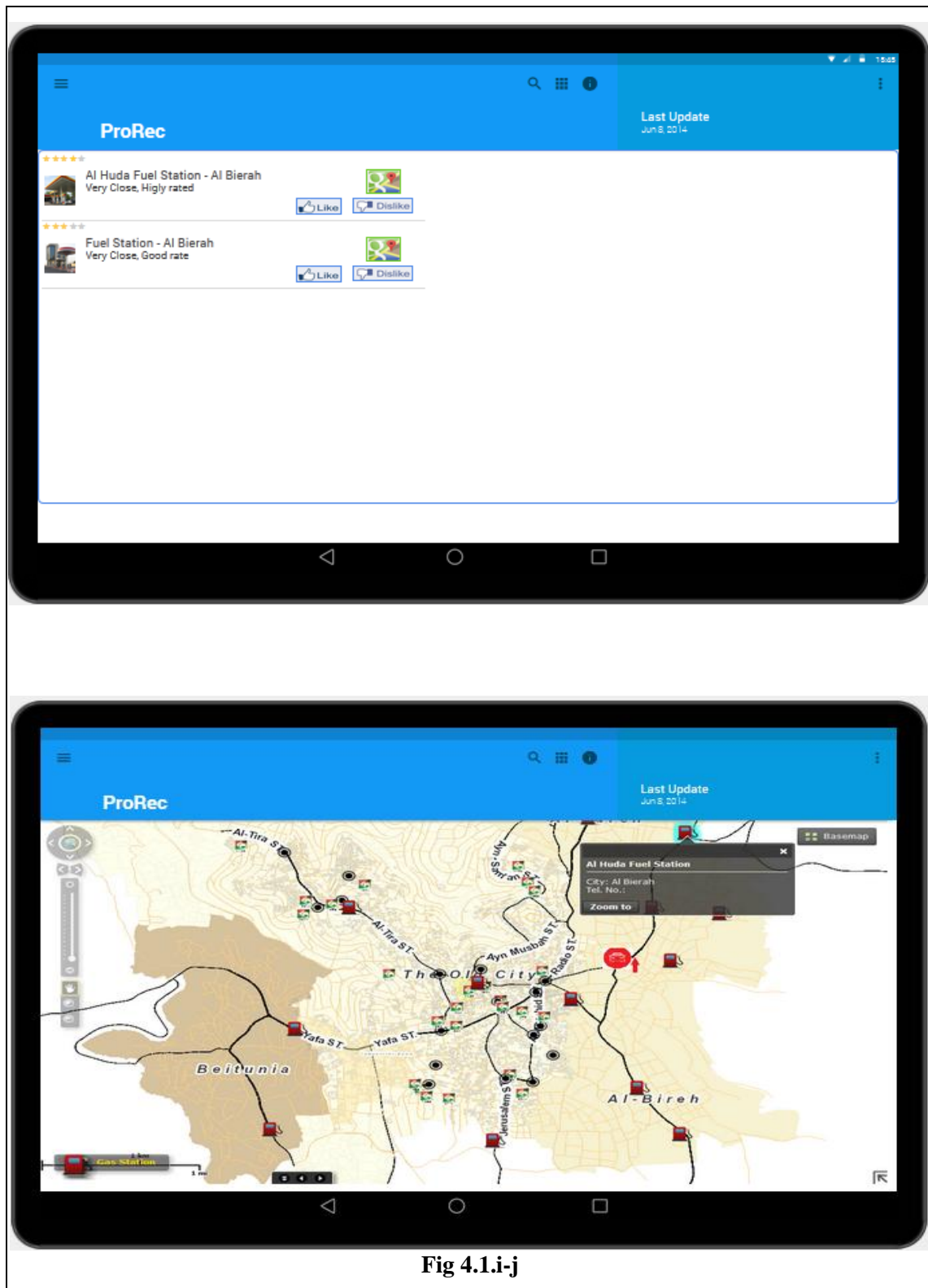
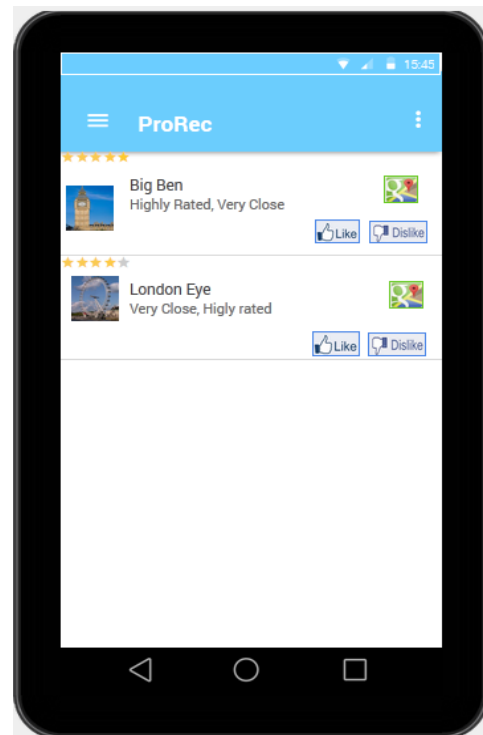
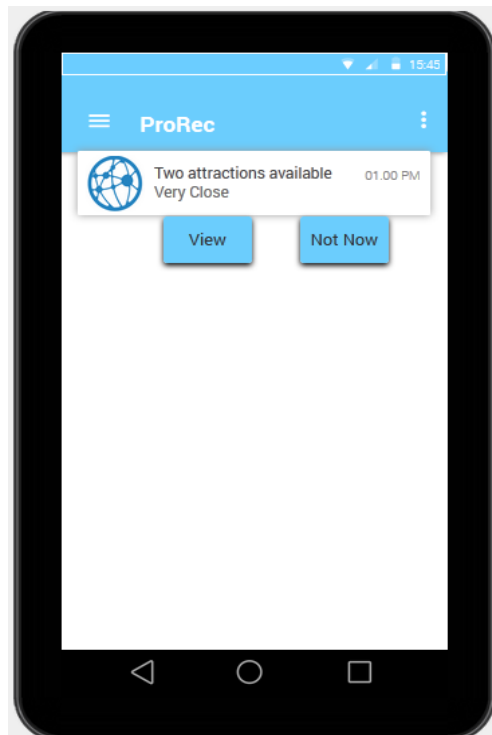


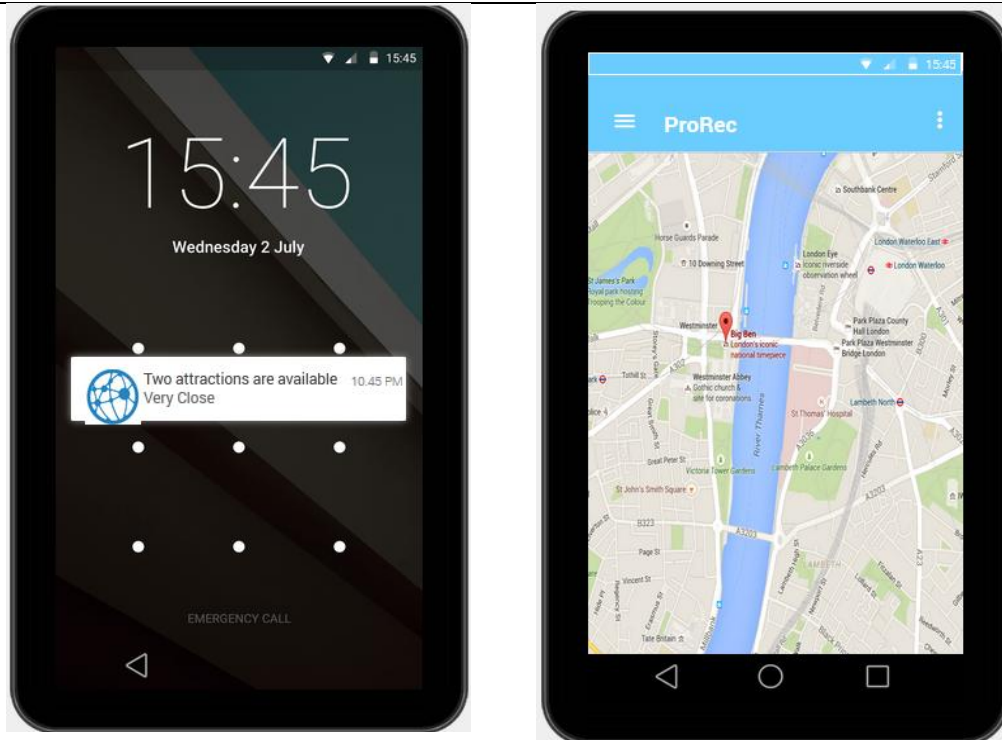
Fig 4.1.i-j

### Scenario # 3

The same user has travelled to London in a vacation.

- He isn't driving a car and it is not a food time.
- ProRec will send a notification of a recommended attractions that suits his context and interests.
- In our demo, these attractions were Big Ben, and London Eye.

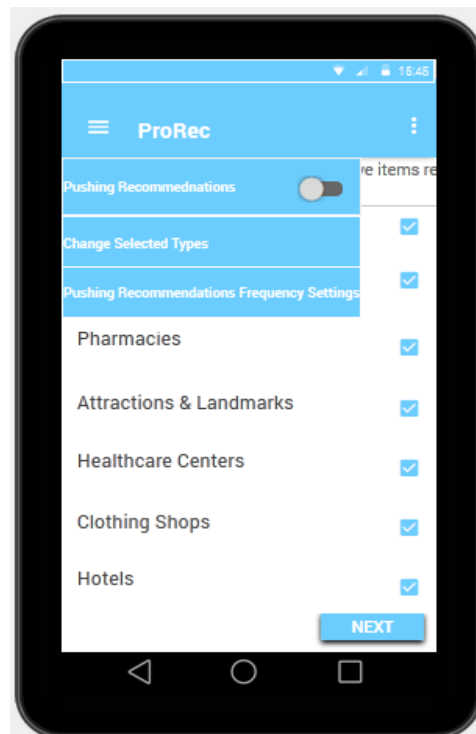




**Fig 4.1.k-n**

### ProRec Settings

- The user can change the types that he wants to receive recommendations of, at any time, by pressing on ProRec icon, then → settings.
- The user can stop receiving recommendations also from the settings.
- The user also can control the frequency of getting recommendations if he wants



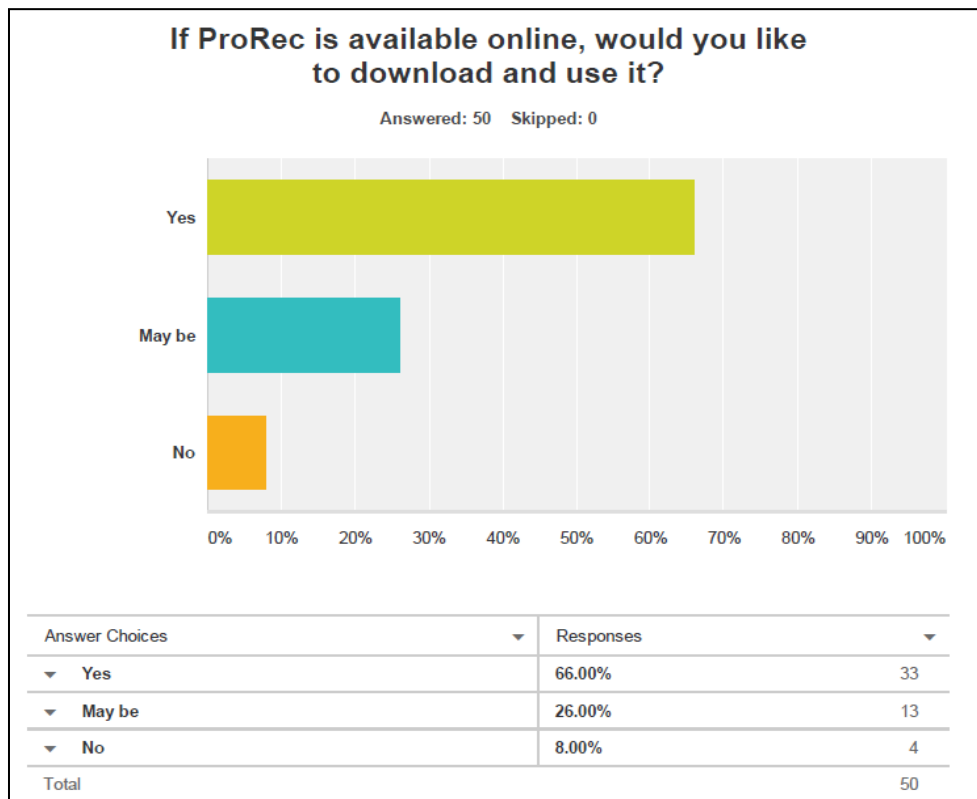
**Fig. 4.1.o**

## 4.2. Users' Acceptance Evaluation

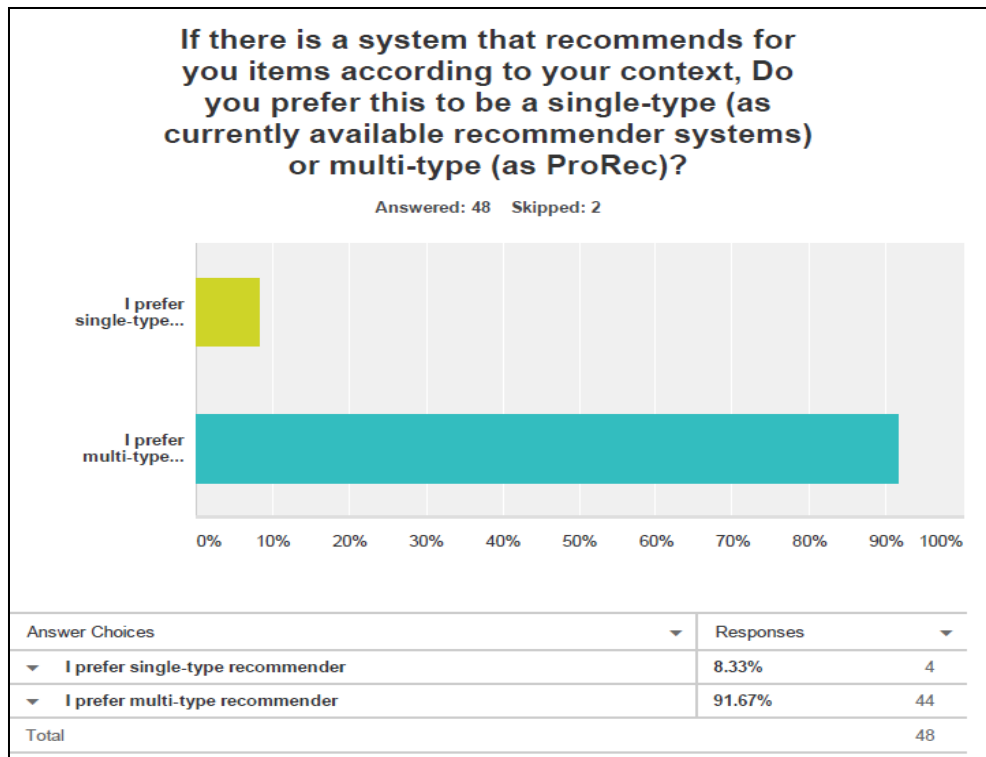
Evaluation of users' satisfaction and acceptance is often seen as a deputy for the success of an application or information system.

In order to evaluate the user's acceptance of our system idea and prototype, a survey had been conducted to 50 random users after showing them the system prototype, and the answers were analyzed. This survey can be found in appendix B.

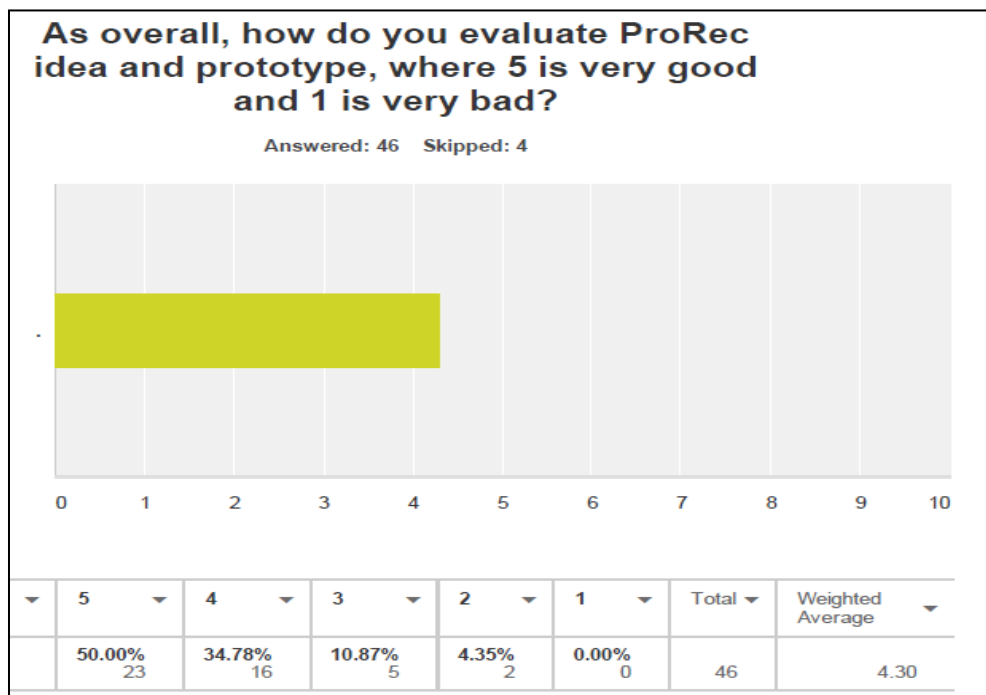
Figures 4.2.a -4.2.e below show the key survey results.



**Fig 4.2.a**

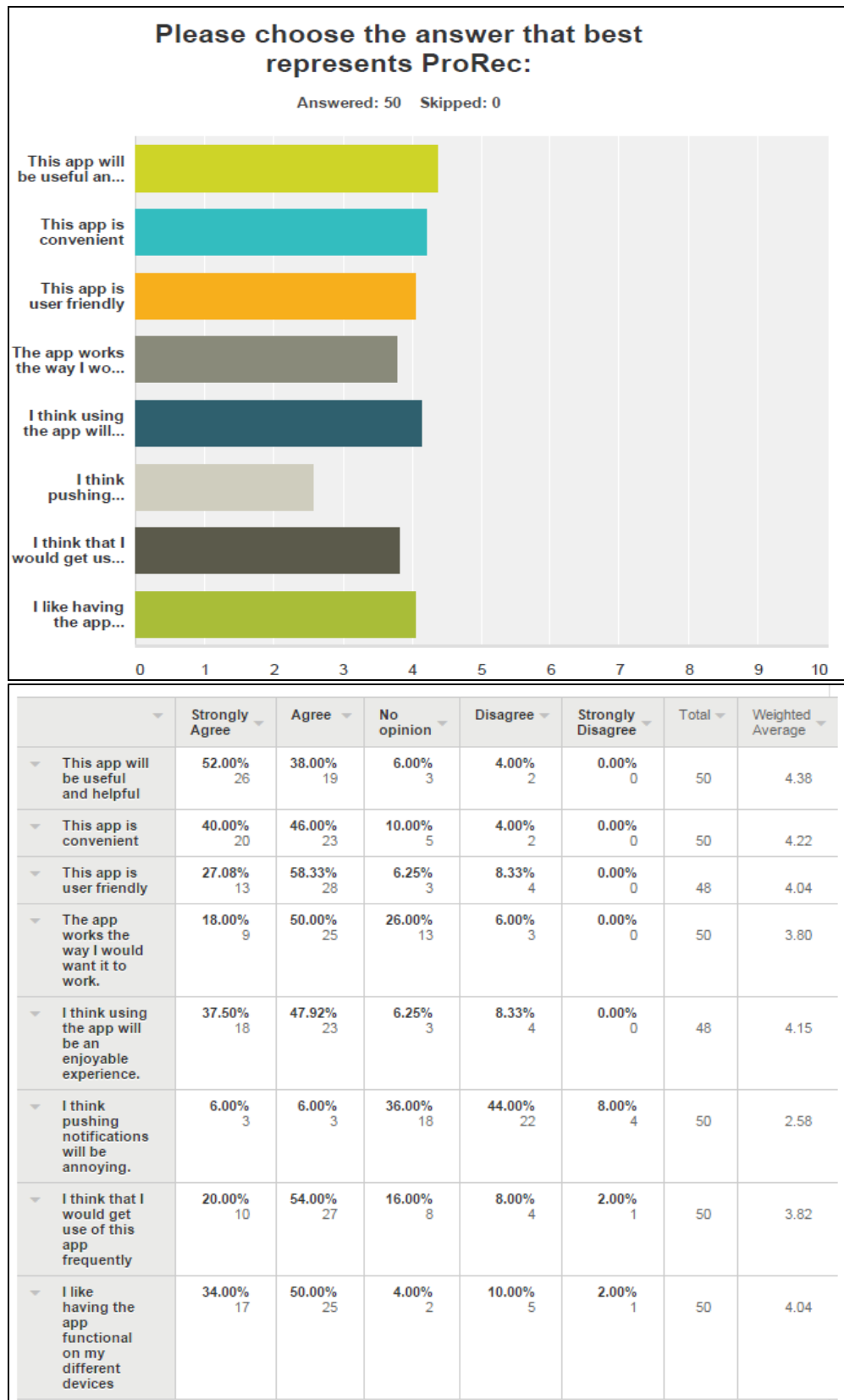


**Fig 4.2.b**

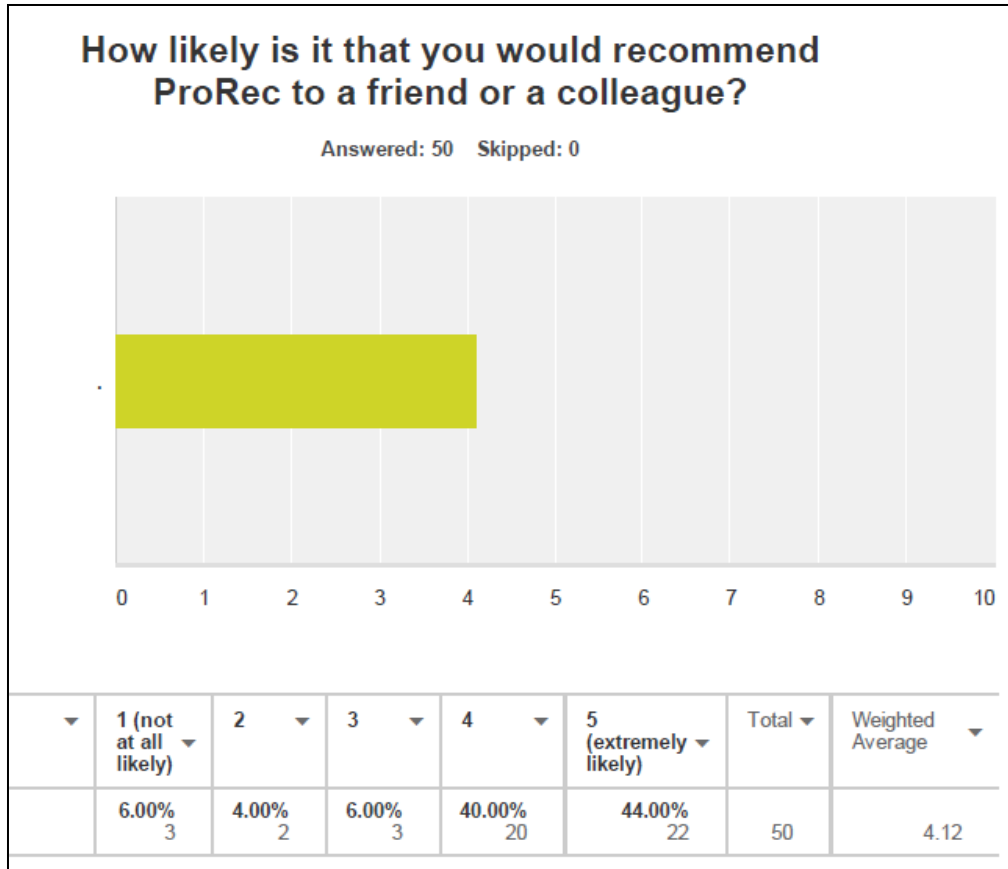


**Fig 4.2.c**





**Fig 4.2.d**



**Fig 4.2.e**

### 4.3 Survey Results & Key Findings

As mentioned earlier, the main aim of the survey was to evaluate ProRec idea and prototype subjectively. The survey results show a clear user's acceptance and encouragement for having such as application and using it. Knowing that the survey respondents were 50 persons, 52% of them were males, most of them where in the age interval (25-34), and their disciplines varies between computer, engineering and other disciplines. Below is a summary of the survey results:

- 96% of the respondents own smartphones, and 68% of them check their smartphones many times per hour. This is a very good encouraging beginning for our system.

- About 92% of the respondents prefer having multi-type recommender rather than single-type. This result indicates that the multi-type idea that was proposed in this thesis have a very strong potential in future when converted into a real application.
- ProRec idea and prototype was evaluated with 4.3/5, which is a very good score. And 66% of the respondents will download ProRec if it is available online, and 26% may be download it.
- The other statements that were asked about ProRec as an application also get high scores for all of them. 90% of the respondents strongly agree and agree this app will be useful and helpful, about 86% agree and strongly agree that this app is convenient, about 85% agree and strongly agree that this app is user friendly, about 68% think that the app works the way they would want it to work, about 86% think that using the app will be an enjoyable experience. Only 12% agree that pushing notifications will be annoying, about 80% think they would get use of this app frequently, and about 84% agree and strongly agree that they will like having the app functional on their different devices.
- About 84% of the respondents would likely and extremely likely recommend ProRec to a colleague or a friend.
- As overall, all the results and user's evaluations were encouraging and satisfactory.

# CHAPTER FIVE

## CONCLUSIONS and FUTURE WORK

This chapter summarizes and concludes our work all through this thesis. Some possible directions for potential future work are also suggested.

### 5.1. Conclusions

The increasing information overload has increased the importance of recommender systems and contributed to their evolution. The fact that the future is for the IoT creates an opportunity to improve the existing recommender systems.

In this thesis, a thorough study of recommender systems history and future has been conducted. Then a design of a proactive multi-type context aware recommender system- in which different types of recommendations are proactively provided to the user in the proper context- has been proposed. The major part that leads to the achievement of our system idea is the CAMS that has been designed and implemented. It is the part that assesses the situation to decide whether to push a recommendation or not and what type of recommendations to push. The CAMS design has been implemented and tested using Matlab; assuming having three types of recommendations, which are gas stations, restaurants, and attractions. An ANN has been designed and implemented to achieve the context reasoning part of the CAMS. Based on the tests results, an excellent accuracy of the system has been achieved.

An application prototype for the system has been developed and a survey has been conducted to a group of users to evaluate the idea and prototype. The users have shown interest in the proposed system, and we got excellent acceptance evaluation from them.

We conclude that this system is promising in the environment of the IoT where much information about the user context will be available. Using the neural network for context reasoning provided more than 98% accuracy of the triggered types in the right contexts. This makes using neural networks for type triggering a strong option to cope with the variety of contexts related data retrieved from the IoT.

## **5.2. Future Work**

Proposing a new system design usually drive to many future works that may be done. Further work may include:

- Improving the adaptivity and accuracy of the system by adjusting the score threshold based in the user's feedback. The more the times the user neglects the recommendations, the higher the threshold to be.
- Increasing the number of types during testing to be more than three, then calculating the accuracy. This will give us better thoughts about the scalability of the ANN in this perspective.
- Developing the system as a real service-oriented application. This will require a complete cycle of software engineering.

# REFERENCES

- [1] Adomavicius, G., Tuzhilin, A.: “Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions”. IEEE Transactions on Knowledge and Data Engineering 17(6), 734–749 (2005).
- [2] J. Bobadilla , F. Ortega, A. Hernando, A. Gutiérrez: “Recommender systems survey”, Knowledge-Based Systems 46 (2013) 109-132.
- [3] ITU, International Telecommunication Union: “Internet of Things: Executive Summary”. ITU Internet Reports (2005) : 2-7. Accessed 06 April 2015. [Online]. Available at:  
[http://www.itu.int/osg/spu/publications/internetofthings/InternetofThings\\_summary.pdf](http://www.itu.int/osg/spu/publications/internetofthings/InternetofThings_summary.pdf).
- [4] John Chambers, “Are-you-ready-for-the-internet-of-everything?”. World Economic Forum 2014. Accessed April 20, 2014.  
<https://agenda.weforum.org/2014/01/are-you-ready-for-the-internet-of-everything/>.
- [5] Balabanovic, M. ,Y. Shoham. Fab: “Content-based, collaborative recommendation”. Communications of the ACM, 40(3):66-72, 1997.
- [6] E. Rich, “User modelling via stereotypes”, COGNITIVE SCIENCE 3, 1979.
- [7] D. Goldberg, D. Nichols, B. Oki, and D.Terry, “Using collaborative filtering to weave an information tapestry”. Communications of the ACM, 35(12):61-70, December 1992.
- [8] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl,” Grouplens: an open architecture for collaborative filtering of net news”. In Proceedings of the 1994 ACM conference on Computer supported cooperative work, (CSCW '94), pages 175-186, New York, NY, USA, 1994. ACM.
- [9] U. Shardanand, P.Maes. “Social information filtering: algorithms for automating `word of mouth””. In Proceedings of the SIGCHI conference on Human factors in computing systems, (CHI '95), pages 210-217, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.
- [10] W. Hill, L. Stead, M.Rosenstein, and G. Furnas, “Recommending and evaluating choices in a virtual community of use”. In Proceedings of the SIGCHI conference on Human factors in computing systems, CHI '95, pages 194-201, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

- [11] J.Breese, D.Heckerman, and C. Kadi, “ Empirical analysis of predictive algorithm for collaborative filtering”. In Proceedings of the 14<sup>th</sup> Conference on Uncertainty in Artificial Intelligence, pages 43-52, 1998.
- [12] M.Balabanovif and Y.Shoham, “Fab: content-based, collaborative recommendation”.Communications of the ACM, 40(3):66-72, March 1997.
- [13] C.Basu, H.Hirsh, and W.Cohen,” Recommendation as classification: Using social and content-based information in recommendation”. In Proceedings of the Fifteenth National Conference on Artificial Intelligence, pages 714-720. AAAI Press, 1998.
- [14] M. Pazzani, “A framework for collaborative, content-based and demographic Filtering”. Artificial Intelligence Review, 13(5-6):393-408, December 1999.
- [15] Greenfield, Adam ,”Everyware: the dawning age of ubiquitous computing”. New Riders. pp. 11– 12. ,2006.
- [16] Jong-yi Hong, Eui-ho Suh, and Sung-Jin Kim. “Context-aware systems: A literature review and classification”. Expert Syst. Appl., 36(4):8509-8522, May 2009.
- [17] G. Abowd and E. Mynatt, “Charting past, present, and future research in ubiquitous computing,” ACM Trans. Comput.-Hum. Interact., vol. 7, pp. 29–58, March 2000. [Online]. Available: <http://doi.acm.org/10.1145/344949.344988>.
- [18] T. Rodden, K. Chervest, N. Davies, and A. Dix, “Exploiting context in hci design for mobile systems,” in in Workshop on Human Computer Interaction with Mobile Devices, 1998. [Online]. Available: <http://eprints.lancs.ac.uk/11619/>.
- [19] R. Hull, P. Neaves, and J. Bedford-Roberts, “Towards situated computing,” in Wearable Computers, 1997. Digest of Papers., First International Symposium on, OCT 1997, pp. 146 –153. [Online]. Available: <http://dx.doi.org/10.1109/ISWC.1997.629931>.
- [20] A. Ward, A. Jones, and A. Hopper, “A new location technique for the active office,” Personal Communications, IEEE, vol. 4, no. 5, pp. 42 –47, oct 1997. [Online]. Available: <http://dx.doi.org/10.1109/98.626982>.
- [21] B. Schilit, N. Adams, and R. Want, “Context-aware computing applications,” in Mobile Computing Systems and Applications, 1994. Proceedings., Workshop on, dec 1994, pp. 85 –90. [Online]. Available: <http://dx.doi.org/10.1109/MCSA.1994.512740>.
- [22] J. Pascoe, “Adding generic contextual capabilities to wearable computers,” in Wearable Computers, 1998. Digest of Papers. Second International Symposium on, oct 1998, pp. 92 –99. [Online]. Available:

<http://dx.doi.org/10.1109/ISWC.1998.729534>.

- [23] A. K. Dey, G. D. Abowd, and D. Salber, "A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications," *Hum.-Comput. Interact.*, vol. 16, pp. 97–166, December 2001. [Online]. Available: <http://dx.doi.org/10.1207/S15327051HCI1623402>.
- [24] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, "Towards a better understanding of context and context-awareness," in *Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, ser. HUC '99. London, UK: Springer-Verlag, 1999, pp. 304–307. [Online]. Available: <http://dl.acm.org/citation.cfm?id=647985.743843>.
- [25] L. Sanchez, J. Lanza, R. Olsen, M. Bauer, and M. Girod-Genet, "A generic context management framework for personal networking environments," in *Mobile and Ubiquitous Systems - Workshops*, 2006. 3rd Annual International Conference on, July 2006, pp. 1–8. [Online]. Available: <http://dx.doi.org/10.1109/MOBISYS.2006.361743>.
- [26] B. Schilit and M. Theimer, "Disseminating active map information to mobile hosts," *Network*, IEEE, vol. 8, no. 5, pp. 22–32, Sep/Oct 1994. [Online]. Available: <http://dx.doi.org/10.1109/65.313011>.
- [27] G. Adomavicius, A. Tuzhilin, "Context-aware recommender systems". In *Proceedings of the 2008 ACM Conference on Recommender Systems*, (RecSys 2008), pages 335–336, New York, NY, USA, 2008. ACM.
- [28] C. Perera, A. Zaslavsky, P. Christen and D. Georgakopoulos "Context aware computing for the Internet of Things: A survey", *IEEE Commun. Surveys Tuts.*, vol. PP, no. 99, pp.1–41 2013.
- [29] S. Pietschmann, A. Mitschick, R. Winkler, and K. Meissner, "Croco: Ontology-based, cross-application context management," in *Semantic Media Adaptation and Personalization*, 2008. SMAP '08. Third International Workshop on, Dec. 2008, pp. 88–93. [Online]. Available: <http://dx.doi.org/10.1109/SMAP.2008.10>.
- [30] J. Indulska and P. Sutton, "Location management in pervasive systems," in *Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003 - Volume 21*, ser. ACSW Frontiers '03. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2003, pp. 143–151. [Online]. Available: <http://dl.acm.org/citation.cfm?id=827987.828003>.
- [31] S. Yanwei, Z. Guangzhou, and P. Haitao, "Research on the context model of intelligent interaction system in the internet of things," in *IT in Medicine and Education (ITME)*, 2011 International Symposium on, vol. 2, Dec. 2011, pp. 379–382. [Online]. Available: <http://dx.doi.org/10.1109/ITIME.2011.6132129>.



- [32] C. Bettini, O. Brdiczka, K. Henriksen, J. Indulska, D. Nicklas, A. Ranganathan, and D. Riboni, "A survey of context modelling and reasoning techniques," *Pervasive Mob. Comput.*, vol. 6, pp. 161–180, April 2010. [Online]. Available: <http://dx.doi.org/10.1016/j.pmcj.2009>.
- [33] G. Chen and D. Kotz, "A survey of context-aware mobile computing research," Department of Computer Science, Dartmouth College, Hanover, NH, USA, Tech. Rep., 2000, <http://www.cs.dartmouth.edu/reports/TR2000-381.pdf>
- [34] T. Strang and C. Linnhoff-Popien, "A context modeling survey," in In: Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing, Nottingham/ England, 2004. [Online]. Available: <http://elib.dlr.de/7444/1/Ubicomp2004ContextWSCameraReadyVersion.pdf>.
- [35] "The Unified Modeling Language". DISQUS. Accessed March 15, 2014. <http://www.uml-diagrams.org/>
- [36] A. Bikakis, T. Patkos, G. Antoniou, , and D. Plexousaki, "A survey of semantics-based approaches for context reasoning in ambient intelligence," in *Ambient Intelligence 2007 Workshops*, M. M, F. A, and A. E, Eds., vol. 11. SPRINGER-VERLAG BERLIN, 2008. [Online]. Available: [http://www.csd.uoc.gr/\\_bikakis/pubs/survey-ami07.pdf](http://www.csd.uoc.gr/_bikakis/pubs/survey-ami07.pdf)
- [37] D. Guan, W. Yuan, S. Lee, and Y.-K. Lee, "Context selection and reasoning in ubiquitous computing," in *Intelligent Pervasive Computing, 2007. IPC. The 2007 International Conference on*, oct. 2007, pp. 184 –187. [Online]. Available: <http://dx.doi.org/10.1109/IPC.2007.102>
- [38] B. Korel and S. Koo, "A survey on context-aware sensing for body sensor networks," *Scientific Research Publishing: Wireless Sensor Network*, vol. 2 (8), pp. 571 –583, 2010. [Online]. Available: <http://www.scirp.org/journal/PaperDownload.aspx?paperID=2345>.
- [39] K. V. Laerhoven, "Combining the self-organizing map and k-means clustering for on-line classification of sensor data," in *Proceedings of the International Conference on Artificial Neural Networks*, ser. ICANN '01. London, UK, UK: Springer-Verlag, 2001, pp. 464–469. [Online]. Available: <http://dl.acm.org/citation.cfm?id=646258.683987>.
- [40] M. Roman, C. Hess, R. Cerqueira, A. Ranganathan, R. H. Campbell, and K. Nahrstedt, "A middleware infrastructure for active spaces," *IEEE Pervasive Computing*, vol. 1, no. 4, pp. 74–83, Oct. 2002. [Online]. Available: <http://dx.doi.org/10.1109/MPRV.2002.1158281>.

- [41] O. Brdiczka, J. Crowley, and P. Reignier, "Learning situation models in a smart home," *Systems, Man, and Cybernetics, Part B: Cybernetics*, IEEE Transactions on, vol. 39, no. 1, pp. 56–63, feb. 2009. [Online]. Available: <http://dx.doi.org/10.1109/TSMCB.2008.923526>.
- [42] L. Baltrunas, F. Ricci, "Context-based splitting of item ratings in collaborative filtering". In *Proceedings of the Third ACM Conference on Recommender Systems*, (RecSys 2009), pages 245-248, New York, NY, USA, 2009. ACM.
- [43] U. Panniello, A. Tuzhilin, M. Gorgoglione, C. Palmisano, and A. Pedone. "Experimental comparison of pre- vs. post-filtering approaches in context-aware recommender systems". In *Proceedings of the third ACM conference on Recommender systems*, (RecSys '09), pages 265-268, New York, NY, USA, 2009. ACM.
- [44] L. Baltrunas, "Exploiting contextual information in recommender systems. In *Proceedings of the 2008 ACM conference on Recommender systems*, (RecSys '08), pages 295-298, New York, NY, USA, 2008. ACM.
- [45] L. Baltrunas, X. Amatriain. "Towards Time-Dependant Recommendation based on Implicit Feedback". In *Context-aware Recommender Systems Workshop at Recsys 2009*, 2009.
- [46] D. Evans, "The Internet of Things: How the Next Evolution of the Internet Is Changing Everything". Cisco Internet Business Solutions Group (IBSG), (April 2011): 2-4. Accessed March 15, 2015.  
[https://www.cisco.com/web/about/ac79/docs/innov/IoT\\_IBSG\\_0411FINAL.pdf](https://www.cisco.com/web/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf)
- [47] K. Ashton, "That 'Internet of Things' thing", *RFID Journal* (2009). Accessed May 10, 2015. <http://www.rfidjournal.com/articles/view?4986>.
- [48] H. Sundmaeker, P. Guillemin, P. Friess, S. Woelfflé, *Vision and challenges for realising the Internet of Things*, Cluster of European Research Projects on the Internet of Things—CERP IoT, 2010.
- [49] M. Weiser, R. Gold, "The origins of ubiquitous computing research at PARC in the late 1980s", *IBM Systems Journal* (1999).
- [50] T. Lu and W. Neng, "Future internet: The internet of things," in *3rd International Conference on Advanced Computer Theory and Engineering (ICACTE)*, vol. 5, August 2010, pp. V5–376–V5–380. [Online]. Available: <http://dx.doi.org/10.1109/ICACTE.2010.5579543>.
- [51] P. Guillemin, P. Friess, "Internet of things strategic research roadmap," *The Cluster of European Research Projects*, Tech. Rep., September 2009, [http://www.internet-of-things-research.eu/pdf/IoT\\_Cluster\\_Strategic\\_Research\\_Agenda\\_2009.pdf](http://www.internet-of-things-research.eu/pdf/IoT_Cluster_Strategic_Research_Agenda_2009.pdf).

- [52] UK Government, The Internet of Things: making the most of the Second Digital Revolution — A report by the UK Government Chief Scientific Adviser, Sir Mark Walport. [https://www.gov.uk/government/uploads/system/uploads/attachment\\_data/file/389315/14-1230-internet-of-things-review.pdf](https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/389315/14-1230-internet-of-things-review.pdf)
- [53] International Data Corporation (IDC), IoT Market Forecast: Worldwide IoT Predictions for 2015, <http://iotinternetofthingsconference.com/2014/12/07/iot-market-forecast-worldwide-iot-predictions-for-2015/>.
- [54] M. Frese, D. Fay, “Personal initiative: An active performance concept for work in the 21st century”. *Research in Organizational Behaviour*, 23, 133-187. (2001).
- [55] D.Tennenhouse, “Proactive computing”. *Communications of the ACM*, 43(5):43-50, 2000.
- [56] A. Salovaara and A. Oulasvirta, “Six modes of proactive resource management”. In *Nordic conference on Human-computer interaction*, pages 57-60, Tampere, Finland, 2004.
- [57] Schafer, J.B., Konstan, J.A., Riedl, J, “E-commerce recommendation applications. *Data Mining and Knowledge Discovery*”. 5(1/2), 115–153 (2001).
- [58] P. Bedi, S.Agarwal, “A situation-aware Proactive Recommender System”, *International Conference on Communication Systems and Network Technologies*, pp 709-713, Katra, India, June (2013), IEEE Computer Society.
- [59] B. Rhodes, “Just-In-Time Information Retrieval”. Phd thesis, MITMedia Lab, 2000.
- [60] D.Billsus, D.Hilbert, D. Aminzade, “Improving proactive information systems”. In *Conference on Intelligent User Interfaces*, pages 159-166, New York, NY, USA, 2005.
- [61] B. Rhodes. “Using physical context for just-in-time information retrieval”. *IEEE Transactions on Computers*, 52(8):1011-1014, 2003.
- [62] Opperman R. and Specht, M. 2000.” A context-sensitive nomadic exhibition guide”. *Proceedings of the 2nd Symposium on Handheld and Ubiquitous Computing*, 127–149.
- [63] I.Podnar, M. Hauswirth, and M.Jazayeri, “Mobile push: delivering content to mobile users”. In *Conference on Distributed Computing Systems Workshops*, pages 563-568, Vienna, Austria, 2002.
- [64] M.Vijayalakshmi, A. Kannan, “Proactive location-based context aware services using agents”. *International Journal of Mobile Communications*, 7(2):232, 2009.
- [65] M. Beigl. ” MemoClip: A location-based remembrance agent. *Personal and Ubiquitous Computing*”, 4(4):230-233, 2000.

- [66] W. Worndl, J. Hubner, R. Bader, and D. Gallego-Vico, “ A model for proactivity in mobile, context-aware recommender systems”. In Conference on Recommender systems, pages 273-276, Chicago,IL, USA, 2011.
- [67] D. Gallego-Vico, W. Woerndl, and R. Bader, “A study on proactive delivery of restaurant recommendations for android smartphones,” in the International Workshop on Personalization in Mobile Applications (PeMA) at 5th ACM International Conference on Recommender Systems, October 2011.
- [68] Fausett L., “Fundamentals of Neural Networks”, Prentice-Hall, 1994. ISBN 0 1042250 9.
- [69] Krenker, J. Bešter, and A. Kos, “Introduction to the Artificial Neural Networks”. Faculty of Electrical Engineering, University of Ljubljana, Slovenia. [Online]. Available: <http://cdn.intechweb.org/pdfs/14881.pdf>
- [70] T. Ayodele , ”Types of Machine Learning Algorithms”, In Tech, University of Portsmouth, United Kingdom, 2011. [Online]. Available: <http://cdn.intechopen.com/pdfs-wm/10694.pdf>.
- [71] C. Gershenson, "Artificial Neural Networks for Beginners" Networks, 2003., vol. cs.NE/0308, pp. 8. [Online]. Available: [https://datajobs.com/data-science-repo/Neural-Net-\[Carlos-Gershenson\].pdf](https://datajobs.com/data-science-repo/Neural-Net-[Carlos-Gershenson].pdf)
- [72] Justinmind Prototyper. 2014. *Justinmind: Interactive wireframes for web and mobile*. [ONLINE] Available at: <http://www.justinmind.com/>. [Accessed 02 September 15].

# APPENDICES

## Appendix A: Matlab Codes

### Ann.m

```
% input - input data.
% target - target data.

x = input;
t = target;

% Choose a Training Function
% For a list of all training functions type: help nntrain
% 'trainlm' is usually fastest.
% 'trainbr' takes longer but may be better for challenging problems.
% 'trainscg' uses less memory. NFTOOL falls back to this in low memory situations.
trainFcn = 'trainscg'; % Scaled Conjugate Gradient

% Create a Fitting Network
hiddenLayerSize = 16;
net = fitnet(hiddenLayerSize,trainFcn);

% Choose Input and Output Pre/Post-Processing Functions
% For a list of all processing functions type: help nnprocess
net.input.processFcns = {'removeconstantrows','mapminmax'};
net.output.processFcns = {'removeconstantrows','mapminmax'};

% Setup Division of Data for Training, Validation, Testing

net.divideFcn = 'dividerand'; % Divide data randomly
net.divideMode = 'sample'; % Divide up every sample
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% Choose a Performance Function
% For a list of all performance functions type: help nnperformance
net.performFcn = 'mse'; % Mean squared error

% Choose Plot Functions
% For a list of all plot functions type: help nnplot
net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
    'plotregression','plotfit'};

% Train the Network
[net,tr] = train(net,x,t);

% Test the Network
y = net(x);
e = gsubtract(t,y);
performance = perform(net,t,y)

% Recalculate Training, Validation and Test Performance
trainTargets = t .* tr.trainMask{1};
valTargets = t .* tr.valMask{1};
testTargets = t .* tr.testMask{1};
trainPerformance = perform(net,trainTargets,y)
valPerformance = perform(net,valTargets,y)
testPerformance = perform(net,testTargets,y)
```

```

% View the Network
view(net)

if (true)
    % Generate MATLAB function for neural network for application deployment
    % in MATLAB scripts or with MATLAB Compiler and Builder tools, or simply
    % to examine the calculations your trained neural network performs.
    genFunction(net,'myNeuralNetworkFunction_new2');
    y = myNeuralNetworkFunction_new2(x);
end

```

---

## test.m

```

function [vector]=test()
n=5;

for i=1:5000

cities=['Ramallah','Jenin','Nablus','Jericho','Nablus','Paris','Rome','Dubai','Amman','Istanbul']
Time=0:23
Gas_Sensor=[0,1]
Car_on=[0,1]
Hungry=[0,1]
out=[0,1]
holiday=[0,1]
temp=0:.5:30

T1=randi(23)
G1=randi(2)
G2=randi(2)
G3=randi(2)
G4=randi(2)
G5=randi(2);
tem=randi(60)
v1=Time(T1);
v2=Gas_Sensor(G1);v3=Car_on(G2);v4=Hungry(G3);v5=out(G4);v6=holiday(G5);v7=temp(tem);

S=[v1,v2,v3,v4,v5,v6]
disp('Time is');disp(v1)

if (v3==1)
    disp('The car is ON')
else
    disp('The car is Off')
end

if (v2==1)
    disp('Gas Tank below threshold')
else
    disp('Gas Tank over threshold')
end
if (v4==1)
    disp('Hungry')
else
    disp('Not Hungry')
end

if (v5==1)
    disp('Outside the country')
else
    disp('Inside the country')
end

if (v6==1)
    disp('It is holiday')
else
    disp('It is not holiday')
end

```

```

end

disp('Temperature is');disp(v7)

ans=myNeuralNetworkFunction_new2(S')
m=max(ans)

push=0;
thresh_res=.5
thresh_gas=.7
thresh_att=.35

if(m>thresh_res || m>thresh_gas || m>thresh_att)
if (m==ans(1)&& m>thresh_res)
    disp('Push a Restuarent Recommendation')
    push=1;
end
    if (m==ans(2)&& m>thresh_gas)
        disp('Push a gas station Recommendation')
        push=1;
    end

    if (m==ans(3)&& m>thresh_att)
        disp('Push an attraction Recommendation')
        push=1;
    end
else
    disp('Dont push any Recommendation')
end

A= [S,ans(1),ans(2),ans(3),push]
p(i,:)=A;

end
p
filename = 'testdata_new14.xlsx';
xlswrite(filename,p);

```

---

## GUI\_final.m

```

function varargout = GUI_final(varargin)

% time1=0;car_on=0;gas_tank=0;hungry=0;outside=0;holiday=0;temp=0;

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn', @GUI_final_OpeningFcn, ...
                  'gui_OutputFcn',  @GUI_final_OutputFcn, ...
                  'gui_LayoutFcn',   [] , ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before GUI final is made visible.
function GUI_final_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
% varargin    command line arguments to GUI_final (see VARARGIN)

```

```

% Choose default command line output for GUI_final
handles.output = hObject;
guidata(hObject, handles);
axes(handles.axes1)
axis off

% Update handles structure

% UIWAIT makes GUI_final wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = GUI_final_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

function edit1_Callback(hObject, eventdata, handles)

% hObject handle to edit1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
time1= str2double(get(hObject,'String')) %returns contents of edit1 as a double
% Save the new volume value
handles.metricdata.time1 = time1;
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, ~)
% hObject handle to edit1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in checkbox1.
function checkbox1_Callback(hObject, eventdata, handles)
% hObject handle to checkbox1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

car_on= get(hObject,'Value') %returns toggle state of checkbox1
handles.metricdata.car_on = car_on;
guidata(hObject,handles)

% --- Executes on button press in checkbox2.
function checkbox2_Callback(hObject, eventdata, handles)
% hObject handle to checkbox2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

gas_tank= get(hObject,'Value') %returns toggle state of checkbox2
handles.metricdata.gas_tank = gas_tank;
guidata(hObject,handles)

% --- Executes on button press in checkbox3.
function checkbox3_Callback(hObject, eventdata, handles)
% hObject handle to checkbox3 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

```



```

hungry= get(hObject,'Value') %returns toggle state of checkbox3
handles.metricdata.hungry = hungry;
guidata(hObject,handles)

% --- Executes on button press in checkbox4.
function checkbox4_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

outside= get(hObject,'Value') %returns toggle state of checkbox4
handles.metricdata.outside = outside;
guidata(hObject,handles)

% --- Executes on button press in checkbox5.
function checkbox5_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

holiday= get(hObject,'Value') %returns toggle state of checkbox5
handles.metricdata.holiday = holiday;
guidata(hObject,handles)

function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
temp= str2double(get(hObject,'String')) % returns contents of edit2 as a double
handles.metricdata.temp = temp;
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
v1=handles.metricdata.time1
v3=handles.metricdata.car_on
%
v2= handles.metricdata.gas_tank
v4=handles.metricdata.hungry
v5=handles.metricdata.outside
v6=handles.metricdata.holiday
v7=handles.metricdata.temp
S=[v1,v2,v3,v4,v5,v6]
ans=myNeuralNetworkFunction_new(S')
m=max(ans)

set(handles.text12,'String',ans(1));
set(handles.text16,'String',ans(2));
set(handles.text17,'String',ans(3));
if(m>.39)
if (m==ans(1))
    disp('Push a Restuarent Recommendation')
    set(handles.text5,'String','Push a Restuarent Recommendation');
    matlabImage = imread('res.jpg');
    image(matlabImage)
    axis off
    axis image

```

```

end
if (m==ans(2))
    disp('Push a gas station Recommendation')
    set(handles.text5,'String','Push a gas station Recommendation');
    matlabImage = imread('gas.jpg');
    image(matlabImage)
    axis off
    axis image
end

if(m==ans(3))
    disp('Push an attraction Recommendation')
    set(handles.text5,'String','Push an attraction Recommendation');
    matlabImage = imread('att.png');
    image(matlabImage)
    axis off
    axis image
end

else
    disp('Dont push any Recommendation')
    set(handles.text5,'String','Dont push any Recommendation');
    matlabImage = imread('no.png');
    image(matlabImage)
    axis off
    axis image
end
end

```

## Appendix B: Survey Form

Please fill this survey after trying ProRec prototype or watching the video in the following link about ProRec:

[https://www.dropbox.com/s/xzxkkwkpu20ougf/ProRec\\_mobile720p\\_clipchamp.com.mp4?dl=0](https://www.dropbox.com/s/xzxkkwkpu20ougf/ProRec_mobile720p_clipchamp.com.mp4?dl=0)

ProRec

1. What is your gender?

☐ Female

☐ Male

2. What is your age?

☐ 18 to 24

☐ 25 to 34

☐ 35 to 44

☐ 45 to 54

☐ 55 to 64

☐ 65 to 74

☐ 75 or older

3. Your major is:

☐ Computer Engineering or Computer Science

☐ Engineering

☐ Other

4. Do you own a smartphone?

☐ Yes

☐ No

5. How often do you check your smartphone?

- ☐ Many times per hour
- ☐ Once every 2-4 hours
- ☐ Many times per day

6. If ProRec is available online, would you like to download and use it?

- ☐ Yes
- ☐ May be
- ☐ No

If answer is no, Why?

7. If there is a system that recommends for you items according to your context, Do you prefer this to be a single-type (as currently available recommender systems) or multi-type (as ProRec)?

- ☐ I prefer single-type recommender
- ☐ I prefer multi-type recommender

8. As overall, how do you evaluate ProRec idea and prototype, where 5 is very good and 1 is very bad?

	5	4	3	2	1
.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

9. Please choose the answer that best represents ProRec after trying it

	Strongly Agree	Agree	No opinion	Disagree	Strongly Disagree
This app will be useful and helpful	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
This app is convenient	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
This app is user friendly	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The app works the way I would want it to work.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I think using the app will be an enjoyable experience.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I think pushing notifications will be annoying.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I think that I would get use of this app frequently	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I like having the app functional on my different devices	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

10. How likely is it that you would recommend ProRec to a friend or a colleague?

	1 (not at all likely)	2	3	4	5 (extremely likely)
.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

11. Do you have any comments or suggestions to enhance ProRec?

# Appendix C: Published Paper

2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications;  
Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing

## *A Proactive Multi-Type Context-Aware Recommender System in the Environment of Internet of Things*

Yasmeen Salman, Abdallatif Abu-Issa, Iyad Tumar, and Yousef Hassouneh

Faculty of Engineering and Technology

Birzeit University

Ramallah, Palestine

{yeld, abuissa, tumar, yhasouneh}@birzeit.edu

**Abstract**—Currently recommender systems are incorporating context and social information of the user, producing context aware recommender systems. In the future, they will use implicit, local and personal information of the user from the Internet of Things, where anyone and anything will be connected at anytime and anywhere. Most recommender systems follow a request-response approach in which the recommendations are provided to the user upon his request. Recently a proactive recommender system - that pushes recommendations to the user when the current situation seems appropriate, without explicit user request - has been introduced in the research area of recommender systems. The fact that the future is for Internet of Things, and the emergence of proactivity concept leads to our system design, in which multi-type rather than one type of recommendations will be recommended proactively to the user in real time. In this paper, a design of a context aware recommender system that recommends different types of items proactively under the Internet of Things paradigm is proposed. A major part of this design is the context aware management system. In this system, we have used a neural network that will do the reasoning of the context to determine whether to push a recommendation or not and what type of items to recommend. The neural network inputs are derived virtually from the Internet of Things, and its outputs are scores for three types of recommendations, they are: gas stations, restaurants and attractions. These scores have been used to decide whether to push a recommendation or not, and what type of recommendations to push among these three types. The results of 5000 random contexts were tested. For an average of 98% of them, our trained neural network generated correct recommendation types in the correct times and contexts.

**Index Terms**—Recommender System, Internet of Things, Context-awareness, Proactivity, Neural Networks

### 1. INTRODUCTION

The vast growth of information on the Internet as well as the increasing number of visitors to websites created a great need to new technologies that can assist us to find resources of interest among the overwhelming items available. One of the most successful such technologies is the Recommender System. A Recommender System (RS) is an intelligent system that suggests items to users that might interest them. Recommender systems apply data mining techniques and prediction algorithms to predict users' interest on information, products and services among the tremendous

amount of available items. A recommender system tries to make its recommendation depending on the user's background and personalization in order to provide affordable, personal, and high-quality recommendations [1]. A common example is Amazon website which recommends books to users depending on what they bought in the past, and what other similar users have bought.

Recommender systems have been developed in parallel with the web. Initially, recommender systems used information from content-based, demographic-based or collaborative filtering techniques or a hybrid between them. Currently, with the development of web 2.0, these systems integrate social information such as followed and followers, friend's lists, posts, blogs, and tags [2].

They also started to integrate contexts in the recommendation process. Where context is any real time known information about a user that helps to give better recommendations, such as time, location, temperature ... etc. This resulted in a new approach for recommendations named as "context-aware recommendation". For example, if the recommender system knows the location of the user at a specific time "from his Smartphone GPS", it will not recommend to him a faraway restaurant, and that will enhance the quality of recommendations.

In the future, recommender systems will use local and personal information from various sensors and devices on the Internet of Things (IoT) [2]. In the IoT, not only people are connected to the internet through their PCs, laptops, and smartphones, but also objects like cars, houses, roads, medical instruments, and many other objects will be connected to the internet. This has already been applied on some countries, partially. But the aim of the IoT is to connect anything, at anytime, from any place, and for anyone [3].

Most of the existing recommender systems deliver their recommendations upon the user's request. But recommendations can be delivered (pushed) to the user without his explicit request. That when the context of the user is known in the real time, the system can know his situation and decide whether this situation needs a recommendation or not. Then recommending a suitable item if that situation is favorable. Later a new term that encapsulates the previous concept has appeared "Pro-activity". It means that the system pushes recommendations to the user when the current situation seems suitable. The

This publication was made possible by NPRP grant # [7-662-2-247] from the Qatar National Research Fund (a member of Qatar Foundation). The statements made herein are solely the responsibility of the author.